



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Identifying context to display correct data for the "interactive document"

**Osmund Chandra  
Maheswaran**

Master of Science in Engineering and ICT

Submission date: June 2013

Supervisor: Ole Ivar Sivertsen, IPM

Norwegian University of Science and Technology  
Department of Engineering Design and Materials



Identifying context to display correct data for the  
”Interactive Document”

Osmund Chandra Maheswaran

June 12th, 2013



# Acknowledgements

While writing this thesis, I have had a lot of help, and I would not have been able to write it without assistance of some key people.

I would like to extend a big "Thank you!" to:

**Geir Iversen**, my contact person at Aker Solutions for helping define the goals of my thesis, and for being very helpful and forthcoming regarding any questions I have had.

**Mozhgan Tavakolifard**, my contact person at NTNU, for providing me with tons of information on context-aware computing, introducing me to Case-Based Reasoning and helping me structure my thesis.

**Mahsa Mehrpour** for the free Visual Studio/C#-consultations. I am sure my work is in good hands if you continue it!

**Ole Ivar Sivertsen**, my supervisor, for being available whenever I have needed it and for providing me with good and fair feedback while always pointing me in the right direction.



# Abstract

## English Abstract

In this master thesis report I have explored the topics of context-aware computing, ontologies, semantic reasoning and case-based reasoning, and how we can apply (and possibly combine) these technologies in an engineering context. I have taken a look at two different formats for visualizing and representing knowledge: Interactive Documents and Knowledge Briefs (K-Briefs). I have investigated how we can combine the aforementioned technologies with these knowledge representation formats. I have also developed a simple application, the K-Brief Recommender, which is meant as a proof-of-concept application. The goal was to prove that we can use Lucene, an open-source search engine, to search in Knowledge Briefs that are stored in rich document formats. This application has been successful, and we can now use this search engine as a baseline for a potential case-based reasoning application, which might be combined with context-aware applications in the future.

## Norsk Sammendrag

I denne masteroppgaven har jeg i hovudsak utforsket noen utvalgte fagfelt, og undersøkt hvordan disse fagfeltene kan bli brukt i en ingeniør+sammenheng. Jeg har også sett på om disse teknologiene lar seg kombinere. Fagfeltene det er snakk om er kontekst-sensitive applikasjoner, ontologier, semantisk resonnering og "case-based reasoning" (CBR). Jeg har også sett på hvordan vi kan kombinere de nevnte teknologiene med to forskjellige format for å visualisere/presentere

kunnskap: Interaktive Dokumenter og Knowledge Briefs (K-Briefs). I tillegg har jeg utviklet en simpel applikasjon, kalt "K-Brief Recommender", som har som formål å bevise at det er mulig å bruke open-source (fritt tilgjengelige, gratis) teknologier til å søke i ofte brukte dokumentformater. Behovet for å bevise dette kommer av at K-Briefs er basert på slike dokumentformater, og det er nødvendig å kunne søke i disse. Hvis vi i tillegg kan oppnå dette ved bruk av open-source teknologier, åpner det for mange spennende muligheter. Applikasjonen har fungert som ønsket, noe som betyr at vi kan bruke den som en basis for en potensiell CBR-implementasjon. Denne CBR-implementasjonen kan potensielt bli videreutviklet slik at den lar seg kombinere med kontekst-sensitiv programvare.



# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Reader's Guide . . . . .	1
1.2 Knowledge Based Engineering . . . . .	2
1.3 AkerSolutions . . . . .	3
1.4 KBeDesign . . . . .	3
1.5 AML - Adaptive Modeling Language . . . . .	3
1.6 Applications of Knowledge Based Engineering . . . . .	4
1.7 LinkedDesign . . . . .	4
<b>2 Problem Description</b>	<b>7</b>
2.1 Knowledge Visualization in Aker Solutions . . . . .	7
2.2 Context-sensitive help . . . . .	8
<b>3 Context-Aware Computing</b>	<b>9</b>
3.1 Introduction . . . . .	9
3.2 Definition of "Context" in Computer Science . . . . .	11
3.2.1 Definition . . . . .	11
3.2.2 Context Categories . . . . .	12
3.3 Shared Contexts . . . . .	14
3.4 Example: Applying the Definition . . . . .	15
3.5 A comment on the Definition of Context from [3] . . . . .	17
3.6 State of the art . . . . .	18
3.6.1 Context Models . . . . .	18

3.6.2	Existing Applications of Context-Aware Computing . . .	19
3.6.3	Discussion . . . . .	21
<b>4</b>	<b>How can we visualize knowledge?</b>	<b>23</b>
4.1	The Interactive Document (Node Example) . . . . .	23
4.1.1	Implementation . . . . .	27
4.1.2	Prototype . . . . .	27
4.1.3	Alternative interpretation of the Interactive Document . .	27
4.2	Knowledge Briefs . . . . .	28
4.2.1	Process Template . . . . .	29
4.2.2	Technical Knowledge Template . . . . .	31
4.2.3	Example K-Brief . . . . .	34
4.2.4	Implementation . . . . .	35
4.3	Addition of Context-Aware Elements . . . . .	35
4.3.1	Interactive Documents . . . . .	36
4.3.2	K-Briefs . . . . .	40
<b>5</b>	<b>Comparison: Interactive Documents and K-briefs</b>	<b>41</b>
5.1	Value Added from the Interactive Document . . . . .	41
5.2	Value Added from K-Briefs . . . . .	42
5.3	User-friendliness . . . . .	43
5.4	Implementation . . . . .	43
5.5	Conclusion . . . . .	44
<b>6</b>	<b>Context Structuring and Modeling</b>	<b>47</b>
6.1	Ontology Crash Course . . . . .	47
6.1.1	The Basics . . . . .	47
6.1.2	Web Ontology Language (OWL) . . . . .	49
6.2	Ontology Models . . . . .	50
6.2.1	Example Ontology-based Context Models . . . . .	53
6.2.2	Context-Driven Information Access in Aker Solutions . . . . .	56
6.3	Identifying Context . . . . .	58
<b>7</b>	<b>Information Location and Mapping</b>	<b>59</b>
7.1	Semantic Reasoning . . . . .	59
7.1.1	What is semantic reasoning? . . . . .	59
7.1.2	Example Application: CONON . . . . .	61
7.1.3	Performance (CONON) . . . . .	64

7.2	Aker Solutions Use Case . . . . .	66
7.2.1	Architecture . . . . .	66
7.2.2	LEAP Architecture . . . . .	67
7.3	Interactive Document Data Sources . . . . .	68
7.3.1	Node Classification . . . . .	68
7.3.2	Calculations . . . . .	69
7.3.3	Node- and beam attributes . . . . .	69
7.3.4	Rules . . . . .	69
7.3.5	Source Code . . . . .	70
7.3.6	AML vs PDMS . . . . .	70
<b>8</b>	<b>Case-Based Reasoning</b>	<b>71</b>
8.1	Introduction . . . . .	71
8.2	The CBR-Cycle . . . . .	73
8.3	Case Representation . . . . .	76
8.4	Similarity Measures (Retrieval Phase) . . . . .	77
8.4.1	Nearest Neighbor Retrieval . . . . .	78
8.4.2	Inductive Approaches . . . . .	78
8.4.3	Knowledge Guided Approaches . . . . .	79
8.4.4	Validated Retrieval . . . . .	79
8.5	When should CBR be used? . . . . .	80
8.6	Why use CBR? . . . . .	81
8.7	State of the Art . . . . .	83
8.7.1	Existing CBR-applications . . . . .	83
8.8	How does CBR apply to Aker Solutions and KBeDesign? . . . . .	83
8.9	Case Representation Using K-Briefs . . . . .	84
<b>9</b>	<b>Related Work</b>	<b>87</b>
<b>10</b>	<b>Prototypes</b>	<b>89</b>
10.1	Interactive Document UI Prototype . . . . .	89
10.2	K-Brief Retriever . . . . .	89
10.2.1	Lucene.Net . . . . .	90
10.2.2	Tika . . . . .	90
10.2.3	IKVM . . . . .	90
10.2.4	How the application works . . . . .	91
10.2.5	Alternative technologies for future research . . . . .	92
10.2.6	Comparison Between Baseline and existing CBR systems . . . . .	93

10.2.7	How does this relate to context-aware computing and/or CBR? . . . . .	93
<b>11</b>	<b>Discussion</b>	<b>95</b>
11.1	KBE and Context-Aware Computing . . . . .	95
11.2	Knowledge Visualization and Representation . . . . .	96
11.3	CBR . . . . .	97
11.4	K-Brief Recommender . . . . .	98
<b>12</b>	<b>Results and Conclusion</b>	<b>99</b>
<b>A</b>	<b>The Interactive Document</b>	<b>103</b>
A.1	Tab 1 - General . . . . .	104
A.2	Tab 2 - Calculations . . . . .	106
A.3	Tab 3 - Node Class . . . . .	112
A.4	Tab 4 - Source Code . . . . .	114
A.5	Tab 5 - Members . . . . .	116
A.6	Tab 6 - Dimensions . . . . .	118
A.7	Project Documentation . . . . .	123
<b>B</b>	<b>Example K-Briefs</b>	<b>125</b>

# Chapter 1

## Introduction

### 1.1 Reader's Guide

In this master thesis report, I have originally been given the task of investigating how one can adapt a concept called the Interactive Document to the LEAP architecture, a software architecture important to LinkedDesign, an ongoing EU research project that Aker Solutions is involved in. However, while writing this thesis, research on the Interactive Document concept has ceased in Aker Solutions and the focus shifted towards another method of displaying knowledge, namely Knowledge Briefs. See chapter 4.1 in LinkedDesign D9.2 for more information about this [10]. There has also been very little documentation available to me regarding the LEAP-architecture. Simple, high-level software architecture figures are available, but there are little details to be found. This is in large part due to the fact that many elements of the LEAP architecture, at least the parts relevant to me *do not exist yet*. At the moment, key parts of the LEAP-architecture only exists in theory. It is therefore very difficult for me to write a complete thesis on something that, presently, only is a principal sketch. Due to these circumstances, I have changed the scope of this report under the guidance of my supervisors and contact persons. Although I will touch upon the subjects of the Interactive Document and the LEAP-architecture, this topic will not be the main focus of this report. In stead, I have focused on other technologies that may be relevant for both Aker Solutions and my immediate academic cir-

cles: context-aware computing, ontologies, semantic reasoning and case-based reasoning, and how we can potentially apply these technologies. I have purposely taken a rather high-level approach when describing these technologies, focusing mostly on the principles, ideas and methodologies behind them, rather than going into a detailed description of how they can be implemented.

The chapters are ordered in the following way: In chapter one, I will describe important topics and stakeholders. In chapter two, I describe the problem that sets the stage for everything I am discussing in this thesis. Chapter three deals with the domain of context-aware computing, which will be important in most of the later chapters. In chapter four, I discuss two ways of visualizing engineering knowledge. This is a part of the solution to the problem discussed in chapter two. In chapter five, I compare these two knowledge visualization methods to each other and discuss pros and cons. Chapter six deals with how we can structure and model context information. In chapter seven, I discuss how the information that is used by the knowledge visualization methods in chapter four is stored. In chapter eight, I discuss a completely new topic, case-based reasoning, and how we can use this technology. In chapter nine I explain prototypes that I have developed to test out certain technologies or principles. Finally, I discuss my findings.

## 1.2 Knowledge Based Engineering

When reading this report, it is important to know about a methodology called Knowledge Based Engineering, or just KBE for short. The main point of this methodology is to reuse knowledge that already exists when engineering new instances of common engineering products, effectively reducing the amount of routine calculations an engineer has to perform by a significant amount. This method can be implemented into computer software, greatly increasing the efficiency of CAD-applications and similar pieces of software that are designed to make an engineer's day-to-day life a little easier. The application of KBE in CAD software simply reduces the amount of models that the user of a given CAD program has to draw, since the software can draw them for him/her. With the help of KBE, the software knows how certain models fit together by referencing a set of rules, and it can figure out how complicated models are put together. These rules are made by gathering knowledge from experienced engineers, who have performed the same designs many times before, before being implemented

into the software. This means that, given a certain minimum of user input, KBE software can create relatively complex models in a short amount of time, instead of the user having to model everything manually.

## 1.3 AkerSolutions

Aker Solutions is a leading global oil services company that provides engineering services, technologies, product solutions and field-life solutions for the oil and gas industry world-wide. The company's knowledge and technologies span from reservoir engineering to production and is applicable throughout the life of an oil field.

## 1.4 KBeDesign

KBeDesign is a department in Aker Solutions' Engineering business area that delivers automated solutions for engineering and reuse of proven product designs. The KBeDesign team creates software to automate routine design activities that can be programmed in a KBE (knowledge-based engineering) system based on the governing standards, best practices and design rules for the designed product.

## 1.5 AML - Adaptive Modeling Language

AML is an acronym for "Adaptive Modeling Language", and is the programming language that KBeDesign has used to write its software. It is an object oriented language based on LISP that also includes features for drawing 3D-models. Simply explained, AML has several already defined classes that represent geometric shapes, which the user can call upon or extend at will. With AML, a user can relatively quickly create 3D-models just by writing a piece of code and running it.

## 1.6 Applications of Knowledge Based Engineering

Aker Solutions engineers a range of products related to oil and gas production. There are several stages to engineering such products; the usual starting point is to conduct some kind of study, followed by a process called front-end engineering and design (FEED). FEED is usually understood as the conceptual planning, programming/schematic design and early project planning that is done in the early stages of projects. It is the process of conceptual development of projects in processing industries such as the oil and gas industry. In practical terms, this often means creating a CAD model of an oil platform (or a module belonging to an oil platform). KBE is especially effective in this phase of a project, since engineers can produce drawings, CAD-models and figures, based on different concepts and ideas, in a relatively short amount of time. Of course, the available applications of knowledge based engineering aren't just FEEDs for the oil and gas industry. KBE can in theory be used in any context where knowledge reuse is helpful, so that engineers don't have to reinvent the wheel for every product they make, or perform the same routine calculations over and over again.

## 1.7 LinkedDesign

LinkedDesign is an EU research project aimed at boosting the European production capabilities. One of the main tools for accomplishing this is developing new collaboration tools for European industries.

From the LinkedDesign website:

*LinkedDesign will boost the productivity of today's engineers by providing an integrated, holistic view on data, persons and processes across the full product lifecycle as a vital resource for the outstanding competitive design of novel products and manufacturing processes.*

This report is somewhat related to LinkedDesign Work Package 9, Deliverable 9.1, which has to do with transparency and traceability in design automation. Here is an excerpt from the report describing LinkedDesign Deliverable 9.1:



*The use of Knowledge Based Engineering in Aker Solutions requires close collaboration between engineers in different domains, handling data in various data formats and a well-structured knowledge acquisition technique followed by transparency and traceability in design automation. The KBeDesign software is used by engineers with different roles in a product's lifecycle. KBeDesign's goal is to visualize product knowledge and lifecycle information in an easily accessible way, enhance data integration and improve collaboration between the engineers involved.*

The key phrase here is “transparency and traceability in design automation”. Here, transparency refers to the ability to see how something is made. It refers to what happens in a process. Traceability refers to the ability to see where something comes from. It refers to why something is what it is. Why we want to improve traceability and transparency is explained in the next chapter.



# Chapter 2

## Problem Description

### 2.1 Knowledge Visualization in Aker Solutions

Picture a structural engineer, making use of KBE software in the future. He is not an expert on the software, nor did he assist in developing it, he is just a normal user. He does not necessarily have much insight in computer science. He has just made some changes to a 3D model, using KBeDesign's KBE-tool to automatically engineer and design new beams and structural nodes. How can he be confident that the new model of the construction is usable? Can he be sure the numbers are accurate? Is the solution presented to him really the best available option? What about the rules and code behind it all? And is the software following the proper rules and procedures for this kind of work?

Today's KBE-tools (used and developed by KBeDesign) provide the user with a 3D model, but not enough information about the rules and standards that determined the model's design. If an engineer does not understand, or does not agree on some aspects of the design, it might be a complicated and time-consuming process to find information that justifies the design.

When an engineer is using software based on knowledge based engineering, he/she will have drawings and models automatically generated for him/her. However, the process behind the model generation isn't necessarily completely clear. The data that are used or generated in this process are not necessarily

self-explaining either.

Because of the problems mentioned above, ways to display and visualize knowledge and rules used in KBE are wanted. In addition to possibly finding ways to improve the user-friendliness of already existing KBE software, issues concerning effective collaboration can also be addressed. In this thesis I will explore ways to better communicate and visualize knowledge contained in KBE software to the user.

To simplify things, we can break the problem down into three subproblems:

- How can we visualize knowledge?
- What knowledge should we visualize, and how do we structure the information?
- Where do we find the information we want, and how do we map it to our desired medium?

## 2.2 Context-sensitive help

Context-sensitive help is a kind of help that is obtained at a specific point in the state of a piece of software, providing help for the situation that is associated with the state. In other words, you get help where you are, when you are there.

Context-sensitive help can be implemented using tooltips, links to topics in a help file, or clicking a "help" button in an application. Another example is to change the pointer to a question mark, and then, after the user clicks in the appropriate place on the screen, the help appears.

This thesis aims to explore how we can provide the best kind of context-sensitive help to users of KBE software. I will discuss different ways to accomplish this, as well as explore the concept of context within computer science further.

## Chapter 3

# Context-Aware Computing

The first technological topic I will discuss is the field of context-aware computing. Earlier I mentioned context-sensitive help. We might have an idea of how context-sensitive help, as most of us has experienced in while using a computer. But what exactly is context? How can we use context to improve the user experience?

### 3.1 Introduction

The field of Context Aware Computing has existed in the academic world since around 1994, when it was introduced by Schilit [1]. In Schilit's article "Context-aware Computing applications", he gives the following definition of context-aware systems:

*Context-aware systems are able to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account. [1]*

Note that by "environmental context" we are not necessarily constraining ourselves to the physical environment, but also social, business and technical envi-

ronments. It is these, often more abstract environments, that will be the most important to this thesis.

In [2], some important general applications of context-aware computing are mentioned. Using context, we can:

- a) Adapt Interfaces
- b) Tailor the set of application-relevant data
- c) Increase the precision of information retrieval
- d) Discover services
- e) Make user-interaction implicit
- f) Build smart environments

Corresponding to the items above, we also have some examples to how these principles could be implemented (also taken from [2]). Imagine that we are in a museum setting. Visitors are given a portable device, that reacts to changes in context by:

- a) Adapting the user interface to the different abilities of the visitor – from low-sighted people to very young children
- b) Providing different information contents based on the different interests/profiles of the visitor (geology, paleontology, scholar, journalist etc.), and on the room he/she is currently in
- c) Learning, from the previous choices performed by the visitor, what information he/she is going to be interested in next
- d) Providing the visitor with appropriate services – to purchase the ticket for a temporary exhibition, or to reserve a seat for the next in-door show on the life of dinosaurs
- e) Deriving location information from sensors which monitor the user environment
- f) Provide active features within the various areas of the museum, which alert visitors with hints and stimuli on what is going on in each particular ambient.

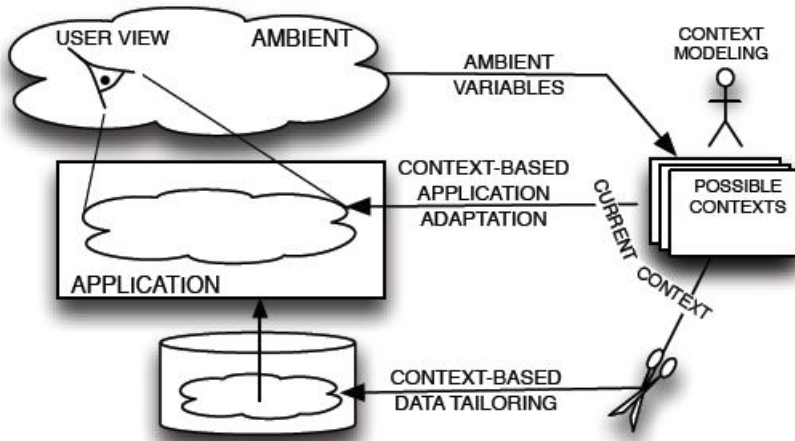


Figure 3.1: A basic example of the architecture of a context-aware application [2]

## 3.2 Definition of "Context" in Computer Science

### 3.2.1 Definition

In everyday speech, the word "context" can be relatively loosely defined, and it is usually rather intuitive to understand what the given context of a situation is. But when it comes to computer science, we need to be a little more specific.

So what do we mean by "context"? Ever since the term *context aware computing* was coined in 1994, this has been a matter of discussion. Many attempts have been made to define the term "context" as precisely as possible, but in the end, most of them fall short. Most definitions up until now have either been incomplete or too general.

In this paper, I will use the definition proposed in [3]: *“Context is any information that can be used to characterize the situation of an entity. Elements for the description of this context information fall into five categories: individuality, activity, location, time and relations.”*

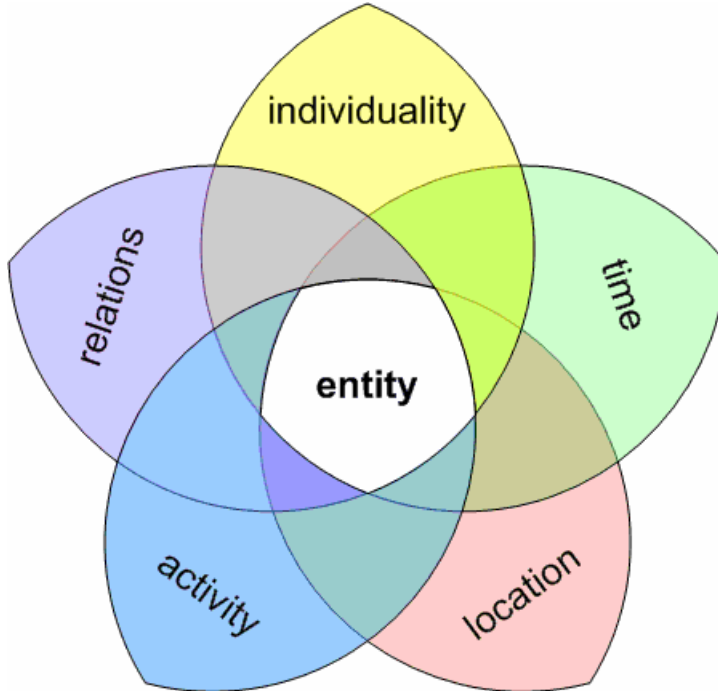


Figure 3.2: Context Categories [3]

### 3.2.2 Context Categories

As mentioned above, context information can be divided into several categories: Individuality, Time, Location, Activity and Relations [3] (see Figure 3.2). Using these categories, it is easier for us to more accurately describe which information that makes up a given context.



**Individuality**

Context information in the *individuality* category is information about the individual entity (or group of entities) that we are working with. It can be virtually any kind of entity: natural entities (entities that represent naturally occurring phenomena), human entities (information that covers the characteristics of human beings, like user profiles), artificial entities (buildings, machines, software, computers, documents) [3].

**Time Context**

*Time* context information is information like the current time, time zones and dates. In this category we can also put time intervals. Time intervals are useful for describing recurring events and user patterns. If we persistently store context information, information in the *Time* category can tell us a lot about usage habits, and we might even be able to predict future usage patterns [3].

**Location Context**

*Location* context information is composed of data that describes an entity's physical or virtual location. A physical location can be described using coordinates, and a virtual location can be described by an IP-address, for example. Location context data also encompasses other spatial information like speed and orientation. Personally, unlike Zimmermann, Lorentz and Oppermann in [3], I would have called this context category "Space" rather than "Location". This is because of these other types of spacial information (speed, orientation etc.).

Location can be absolute or relative to something else. Models for physical locations can be either quantitative (geometric, coordinates), qualitative (symbolic, names of places, buildings or rooms) [3]. In some applications it might be useful to combine both qualitative and quantitative location data to present a fully detailed explanation of where an entity is. One can also use quantitative information to determine qualitative information, and vice versa. For example, given a set of coordinates, an application can determine that the entity in question is inside a specific building which contains the given coordinates. And the other way around: if we know which building we are in, and the application knows the coordinates of this building, we can approximate our own physical coordinates.

### Activity Context

In the *Activity* category of context, we can place information that is relevant for the entity's or entities' tasks. We can ask ourselves: "what does the entity try to achieve, and how?". Activity context information can be expressed as explicit tasks, goals and actions [3]. In [3], we can also find a definition for the term "task" that can be useful: A *task* can be defined as *a goal oriented activity expectation, expressed in a small, executable unit*. Tasks include operation sequences with a determined goal, to which a context aware system can adapt the necessary functions or sequences of functions.

Considering how tasks can be grouped together, it is reasonable to structure tasks into hierarchies, where high-level tasks are composed of sets of low-level tasks. Activity context information can be represented by (domain specific) task models that structure tasks into sub-task hierarchies, which is the most advanced representation of user goals [3].

### Relational Context

*Relational* context information capture the relations an entity has established to other entities (persons, things, services, devices, information). Relations do not need to be static - they may appear and disappear dynamically [3].

Since there are many possible types of relations between entities, it is helpful to categorize relations by the type of entities involved. We can sort relations into three categories: social, functional and compositional relations. Social relations are social connections between two or more people. Functional relations mean relations where entities make use of each other for a certain purpose. Compositional relations are relations between a whole and its parts. The parts will cease to exist if the containing object is destroyed [3].

## 3.3 Shared Contexts

Shared context emerge when two entities are combined and context information overlaps. For two entities to communicate, they need to share the same time and space. After that, actual commonalities are discovered. In [3], an example of two people meeting on a bus is used. For the two people to actually create any relations and discover commonalities, they have to at some time occupy the same space (the bus) and in the same time (even though they are on the same

bus, it doesn't help if one of the persons gets off the bus before the other person gets on it).

### Adjusting Shared Context

Two entities can adjust their shared context to achieve better understanding and communication between each other. For example, a doctor can explain a disease to a patient using language that is a little simpler than he would use if talking to a colleague. If he explains the disease using reference material that is known to both him and the patient, communication between the two will be better. The patient will be more able to understand what is happening and ask questions. This is all possible because the doctor *adjusts* his language to facilitate better communication with his patient [3].

### Exploiting Relations

It is nothing new that the more two people have in common, the more likely they are to understand each other. This is also the case when discussing context. The larger the shared context between two entities is, the easier it is for them to communicate. This is called *exploiting relations* between entities [3].

The concept of exploiting relations is something that is inherently applicable in computer science. The more information two communicating entities have in common, the less effort they have to put into adjusting their context to each other. Interoperability is an example of this. If two applications are created using the same programming language (i.e. two java applications), it is usually easy to integrate them with each other, because they literally speak the same language. Here we can consider each application as an entity. If you want to integrate some java code into your .NET application, however, you have to make some adjustments. Here, we can also draw some immediate parallels to ontologies and semantic web technologies: if two applications share the same ontology, then naturally it is easier for them to work together. More on this later.

## 3.4 Example: Applying the Definition

The point of defining context as precisely as possible is to break the context down into smaller, concrete pieces of information that are far easier to handle.

Let's try to illustrate how we can use the definition of context from [3].

Peter is an engineer working in a big engineering company, which performs many different projects for a range of different clients. Across the company, there is a lot of information available, as well as a lot of competent people. But every time Peter is assigned to work on a new project, it is not always easy to find the information that he needs. Sometimes the information is too inaccurate or irrelevant, and sometimes it is hard to find relevant information at all.

Lucky for Peter, this is all about to change for the better. His company has finally finished a tune-up of the company's CAD/CAE-systems, and added many new context-aware features.

One day, Peter is working on a project, where his job is to help create a 3D-model of a product. The 3D-model is made up from several parts. When Peter has to add a certain type of part to the model, Peter becomes curious. The part looks different from the last time he used it, when he was working on another project. In the new system, there is a "View Documentation"-button available for each part Peter presses this button. Now, the new, context-aware system is given its time to shine. The context-aware system breaks down the current context the following way:

- *Individuality*: The system knows that Peter is the user, since he is logged in with his account. From Peter's profile, the system knows all the important work-related information about him, like which department he is associated with, which project he is currently assigned to (and has been assigned to in the past) and what his areas of expertise are.
- *Location*: The system knows which office location Peter is associated with.
- *Time*: The system notes the time at which Peter decides to search for documentation.
- *Activity*: The system knows which software tool Peter is currently using, which model he is working on, which project this model is associated with, and which part he is examining.
- *Relations*: Here, information about who else is working on the same project and in the same department as Peter is recorded. The part's place in the compositional hierarchy is also noted.

As Peter searches for documentation about the specific part, the systems utilizes this information. Not all of it may be relevant for this particular search, but a lot

of it is. While searching, the system notes that there is an engineer, working on a specific project, looking into information about a specific part. Peter receives the following result:

- Peter is shown several documents. The first is the "core" document regarding the part, showing Peter the initial design of the part. The other documents are each related to their respective projects, where alterations to the part have been made. It turns out that in some projects, the project engineers have decided to change the part slightly. Peter learns that this is usually to satisfy either special engineering conditions related to the given project, or requirements imposed by the customer.
- Although Peter is shown several results, the document which matches his current project is ranked the highest. Peter views this document. Here he learns that the part has been modified slightly, in a way such that the customer can more easily manufacture it after Peter's company has engineered it for them.
- The engineers who made the new design are listed in the document. If Peter wants to, he can reach them via instant messaging or audio/video calls, or he can schedule a meeting with them, all using links in the documentation.

This example also demonstrates some of the appeal of context-aware applications. Instead of starting a lengthy search process himself, Peter can find the information he is looking for in a single click.

### 3.5 A comment on the Definition of Context from [3]

The definition found in [3], and the examples that are used to explain it in the same article, are incredibly general, but still accurate to a satisfying degree. The examples that are used to demonstrate how we can apply this definition, however, are a little ambitious. It is clear that the authors of [3] are not content with only using this definition in computer science. Instead of keeping to technical, more tangible examples where phones, computers and other devices are communicating together (which is the actual application for this kind of research), Zimmermann and his colleagues take a more philosophical and gen-

eral approach. Instead of trying to come up with technical applications of the principles they present, they are using everyday situations to describe things. This is both good and bad. While on one hand, they prove that the definition presented is indeed applicable to most situations. But on the other hand, they deprive themselves of an opportunity for presenting some great examples of how their definition of context could actually be used in any practical way.

For example: instead of explaining how shared contexts work by using an example of two people meeting on a train, I would have liked to see an example where two devices or pieces of software communicate. I have tried my best to supplement with technical and practical examples if none were available in [3].

The only constructive thing I can do with this is of course to view this as a challenge. We have been presented with a very good definition of context. Now, using this definition and the context categories presented, we can explore the technical and practical possibilities ourselves.

Finally, I want to mention that, although the definition in [3] is very useful, any optimal definition and categorization of context information ultimately depends heavily on the application domain and use case. One should not be afraid to make adjustments if it makes for a better result.

## 3.6 State of the art

In this section, I will give some insight into where the field of context-aware computing is today, and what applications currently exist. Since the introduction of the topic in around 1994, context-aware computing has become more widely known. A lot more research papers have been written, and several applications have been developed.

### 3.6.1 Context Models

There are a lot of possible ways to model context information. A context model is needed to define and store context data in a machine processable form [4]. The different available context models are essentially different data structures, and we can use each of these data structures as a "skeleton" for context information.

The basic ways to model context are ([4, 2]):

- **Key-value pairs:** The simplest data structure. Frequently used in various service frameworks, where the key-value pairs are used to describe the capabilities of a service. These key-value pairs are then also used in matching algorithms in service discovery.
- **Markup Scheme Models:** A hierarchical data structure consisting of markup tags with attributes and content (for example RDF/S).
- **Graphical Models:** Context can be modeled using graphical models like UML.
- **Object Oriented Models:** Object-oriented techniques allows developers to use principles like encapsulation, reusability, inheritance and so on. The possibility of encapsulating details of context processing and representation are useful elements of this model. Access to the context and its processing logic is provided by well-defined interfaces.
- **Logic Based Models:** These models have a high degree of formality. Facts, expressions and rules are used to define the context model. Inference/reasoning can be used to derive new facts based on existing rules.
- **Ontology Based Models:** Ontologies represent descriptions of concepts and relationships. Ontologies are inherently well suited to model contextual information due to their high and formal expressiveness and the possibilities for applying ontology reasoning techniques. Various context-aware frameworks use ontologies as underlying context models.

### 3.6.2 Existing Applications of Context-Aware Computing

In this section I will provide a short overview of existing applications within the domain of context-aware computing. It should be noted that, in general, the applications listed are have not been developed with the definition of context from [3] in mind, and therefore have their own way of categorizing context. This is probably due to the fact that [3] is a relatively recent publication.

- **ACTIVITY:** Based on a concept called Activity Theory, which allows description of key aspects influencing human activity. Context categories are *user*, *community* and *rules*, which are used to relate a user to his/her

community. This application/model is still in development, and is deemed too general and holistic to be effective in practice [2].

- *CASS*: A centralized server-based context management framework, meant for small portable devices, offering a high-level abstraction on context sensed by appropriate distributed sensors. It manages both time and space, taking into account the context history, and provides context reasoning; it does not contain user profiling capabilities. Context is described using *location* and *environment* [2, 4].
- *CoBrA*: CoBrA is an acronym for "Context Broker Architecture", an application developed mainly for event/meeting management. The main distinguishing feature of CoBrA is the presence of a central context broker (a "server"), which maintains and manages a shared context model, which a community of agents ("clients") can access. Agents which access the context can be applications in mobile devices, devices in a room or web services. To handle large amounts of data traffic, CoBrA offers the possibility of creating broker federations. CoBrA uses an ontology model written in OWL. The ontology contains many classes, but if we look at the big picture, we can see that the main context categories are *agent*, *location*, *activity* and *time*. In other words, this application uses a context categorization that is relatively close to the one defined in "An Operational Definition of Context" by Zimmerman and his colleagues [2, 3, 4, 5].
- *CoDAMoS*: Context Driven Adaptation of Mobile Services. Aims to introduce context-awareness into mobile devices to offer *personal services*, based on the user's *tasks* and *needs*. Proposes a very general ontology-based context model. Context is categorized into *personal context* (user's wishes, profile and settings), *location* and *device resources* (resource constraints on the device the application is running on) [2, 6].
- *SOCAM*: Service-Oriented Context-Aware Middleware. An architecture for the rapid building and prototyping of context-aware mobile devices. Like CoDAMoS, it uses a very general ontology-based context model, written in OWL, designed to be reused in other applications. Context is categorized into *computer entity*, *location*, *person* and *activity* [2, 4, 7].

In addition to the applications/frameworks I have listed so far, there are plenty of other examples, which I will leave to the reader to explore further if deemed interesting:

- COMANTO [2]



- ConceptualCM [2]
- Context-ADDICT [2]
- CSCP [2]
- EXPDOC [2]
- FAWIS [2]
- GraphicalCM [2]
- HIPS/HyperAudio [2]
- MAIS [2]
- SCOPES [2]
- U-Learn [2]
- Context Management Framework [4]
- Context Toolkit [4]
- CORTEX [4]
- Gaia [4]
- Hydrogen [4]

### 3.6.3 Discussion

A lot of the existing applications and research focus heavily on using physical sensors and the data gathered from them to define context. Current research focuses a lot on *pervasive* context aware applications, where devices and applications are seamlessly integrated into the user's daily life. While this provides for some very interesting possibilities, this might not necessarily be very relevant for the challenges that Aker Solutions and KBeDesign are trying to address. A lot of existing applications and research focuses on using a lot of raw physical data (GPS positioning, temperature, speed, orientation, chemical sensors). This claim is supported in [4]: "Although most authors refer to abstract context sources, the currently mainly used and tested sources are physical sensors. Virtual and logical sensors are capable of providing useful context data as well and should be more incorporated in ongoing research". I think that for Aker Solutions and KBeDesign's goals, we need to focus on more abstract, organizational

and social data to define context. Instead of knowing exactly where someone is, how fast they are moving and the temperature in their immediate surroundings, we want to know which competencies a person has, what their schedule looks like and which projects they are involved in, to name a few examples.

## Chapter 4

# How can we visualize knowledge?

I have mentioned using context-aware applications and context-sensitive help to view relevant documentation in an engineering context. But what is this documentation going to look like? How are we going to present the information? In this chapter, I will address these questions.

### 4.1 The Interactive Document (Node Example)

The Interactive Document is a format inspired by the ideas presented in Linked-Design WP9 D9.1, chapter 6.1.2 [8]. The idea behind the interactive document is to create a format in which we can present the necessary information and rules behind the creation of a CAD-model. It is essentially a “help window” with information about a selected model. By browsing through the Interactive Document, the user should be able to find answers to all questions about why the model looks the way it does. The interactive document is sometimes referred to as the “informal model” [8].

In addition to the answering the “hows” and “whys” concerning 3D CAD model generation in KBE-tools, I have made an effort to explore further, and see what else we can get out of the Interactive Document, since it essentially works as

an information base for the given model. The most important features of the Interactive Document are:

- The possibility to instantly look up rule explanations for the rules that govern a model's dimensions
- Performing calculation checks, verifying that a model is correctly designed according to load conditions
- The ability to view documentation that is relevant to the creation of the model and the project that it belongs to

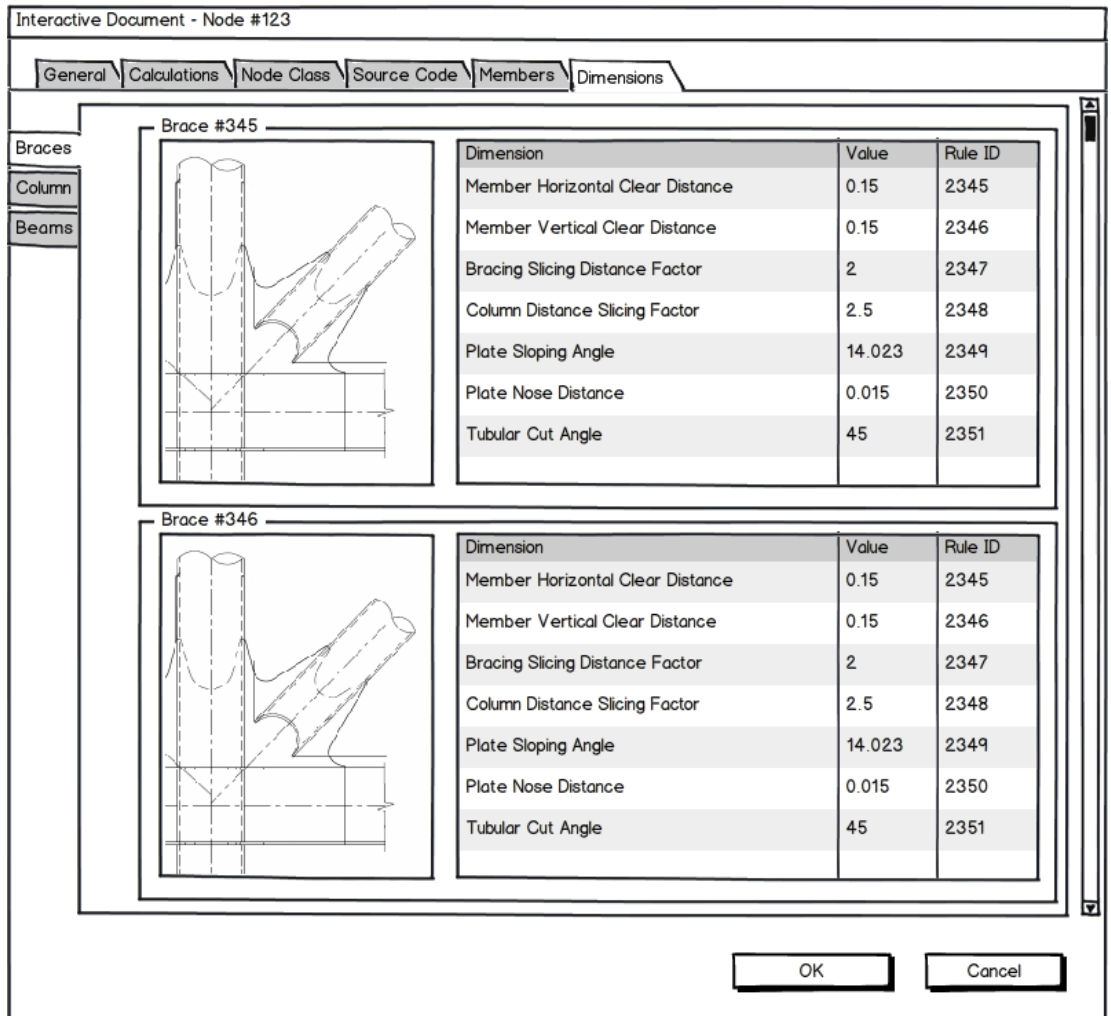


Figure 4.1: The Interactive Document [9]

The format of the Interactive Document is a tab-form window (see Figure 4.1). The different tabs divide the document into different areas of interest. Different types of models will probably have different tabs available. For example, in this

report, I will be working with models of structural nodes/joints. If we were to view an interactive document for, say, an access platform, the Interactive Document, at least in its current version, would look different and contain different tabs.

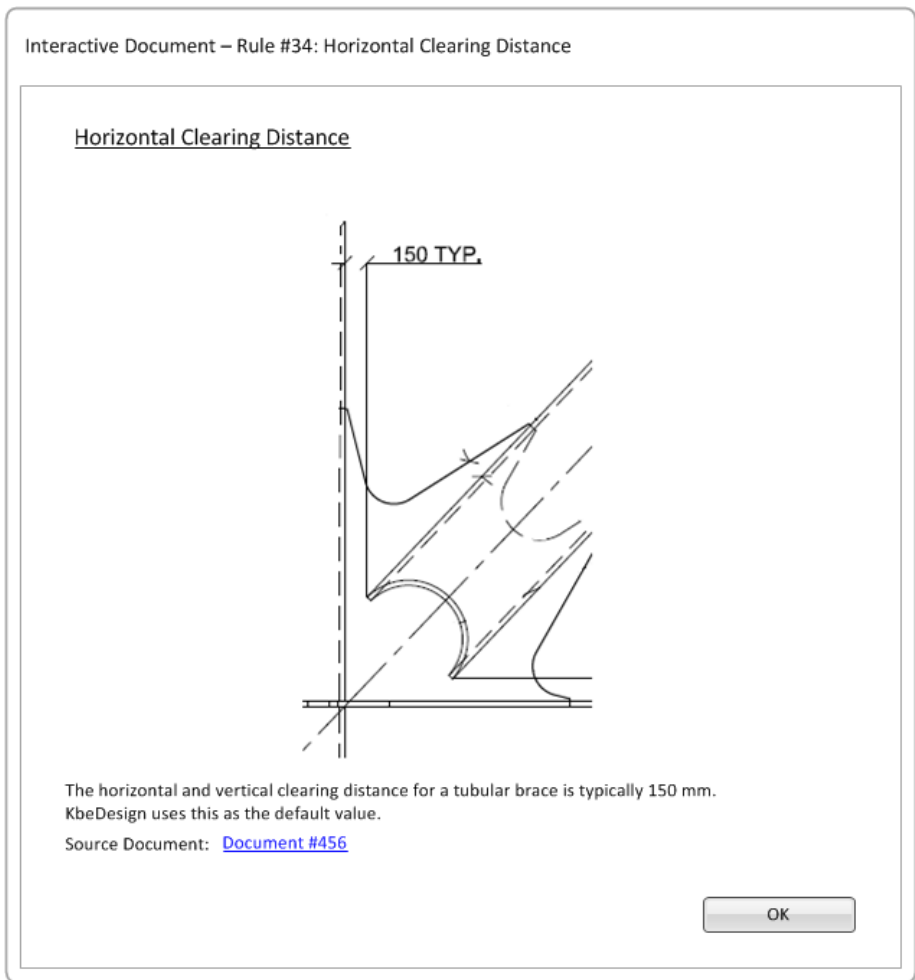


Figure 4.2: Rule Explanation featured in the Interactive Document [9]

I have made some sketches that exemplify how the Interactive Document for nodes/joints could look like. These, along with a detailed description of the Interactive Document can be found in the appendix. I have only added a couple of examples of interactive document features here, like showing dimensions of a model (Figure 4.1) and rule explanations for these dimensions (Figure 4.2).

For more details on the Interactive Document, see the appendix.

### 4.1.1 Implementation

The seemingly best choice for implementing Interactive Documents is to use Microsofts .NET framework. Using .NET, it is possible to integrate AML and PDMS, since PDMS is based on .NET. [9]

The connection between .NET and AML is made possible by using one of KBeDesign's in-house .NET applications that access the AML console. The application is able to input commands into the AML console, and read the output. The mechanisms used for accomplishing this are mainly a couple of classes in .NET called UserControl and CustomControl. Using this application, we can, for example, access the dimensions of a beam. [9]

### 4.1.2 Prototype

As part of my project assignment in 2012, I made a simple GUI prototype in .NET that demonstrates how the Interactive Document could work (see the "Prototypes"-chapter).

### 4.1.3 Alternative interpretation of the Interactive Document

Instead of thinking about the Interactive Document as an application/feature within PDMS (Plant Design Modeling System, CAD-system used by Aker Solutions) and/or AML, one can instead think of it as a collection of features that can be implemented into PDMS or AML where ever they might fit in. Let's consider an example with KBeDesign's AML GUI. If you select a node and click "edit node", you can see all the editable properties of the node. An alternative

solution to increasing the transparency and traceability of the data presented in this screen could be to incorporate the features of the "Dimensions" tab in the Interactive Document into the "edit node" screen. Similar approaches could be taken with respect to the other tabs and features of the Interactive Document.

To put this in another way: instead of the Interactive Document being a separate entity, its features could be integrated into the already existing workflow.

## 4.2 Knowledge Briefs

Knowledge Briefs (K-Briefs) are inspired by the A3-method used in the Lean paradigm, which was originally invented by the Toyota Motor Corporation. The A3-method is named after the A3-size paper format. The idea is that one should be able to present a problem and its proposed solution using only the space available on a single A3 paper sheet.

Now, after the Lean paradigm has become more widely known, A3s are usually called Knowledge Briefs, or K-Briefs for short. In Aker Solutions, the K-Brief was originally intended only for capturing knowledge from meetings and discussions, but further investigation has shown that it can be equally useful for presenting information at a later time.

*"Engineers like visual information. Often during meetings, ideas, geometric relationships, explanations of terminology etc. are sketched on A3 sheets. The resulting sketches are often understandable only to the meeting attendees. Without adequate written documentation of the ideas or knowledge behind these sketches, the understanding behind them, as well as their value, will decrease with time. In order to maintain the knowledge captured during a meeting it is therefore important to have a format for structuring knowledge in meetings."* [10]

The appeal of the K-Brief is its use of visual information. Humans are much more successful in absorbing and retaining information if it is at least partially visually enhanced [10].

The purpose of the K-Brief is to present knowledge in a visual manner, to the extent that it is possible, and to structure this knowledge in a presentable form while it is being captured. Due to the limitations of the format, it is important that a K-brief only contains the most relevant information. If further



information should be necessary, the K-Brief will provide references to other relevant sources of knowledge, like literature, people, reports or other K-Briefs [10].

In engineering, it is useful to split between process knowledge and technical knowledge. These two forms of knowledge have their respective K-brief templates [10].

### 4.2.1 Process Template

Theme	Ownership
Problem statement	Countermeasure selection
Problem analysis	
Goals	Verification method(s)
Alternative evaluation	Implementation and follow-up plan

Figure 4.3: K-Brief Process Template [10]

In Figure 4.3, we can see the K-brief template used for process knowledge. It is called the Plan-do-check-act template [10].

The left-hand side of the K-brief presents the problem and an analysis of the reasons that trigger the problem's occurrence. Goals for the solution of the problem are listed and solution ideas are provided. Through reading the left-hand side of this K-brief, one should understand the situation as it is right now and what the ideal situation should look like after the problem has been solved [10].

On the right-hand side, the best solution is chosen and verification methods of the solution are described. The method of implementing the solution is presented together with a follow-up plan on how to make sure the solution is the most effective [10].



- Important design trade-offs and decisions
- Reusable design elements
- Solutions to critical-to-quality issues
- Solutions to critical-to-cost issues
- Performance curves
- Raw material/component data
- Test results for common design elements
- Reliability/environmental data
- Factory design rules/capability data
- Supplier design rules/capability data
- Frequently used parts/raw materials

Some of these data types might be more useful than others. In addition to the types of data listed above, Knowledge Engineers at KBeDesign have suggested to also include the following data types (this list is also found in [10]):

- Parameter constraints with background: Information about constraints to parameters and explanations of why those constraints have been set
- FAQs to avoid repetitive support requests concerning the same problems
- Links to relevant experts that work within the area of knowledge presented in the K-brief
- Links to further reading (including other K-briefs)
- Links to projects currently working with this knowledge
- Links to conversations started from a K-Brief

Taking all of this into consideration, Aker Soltions have arrived at a new format for the technical K-brief template(see 4.5) :

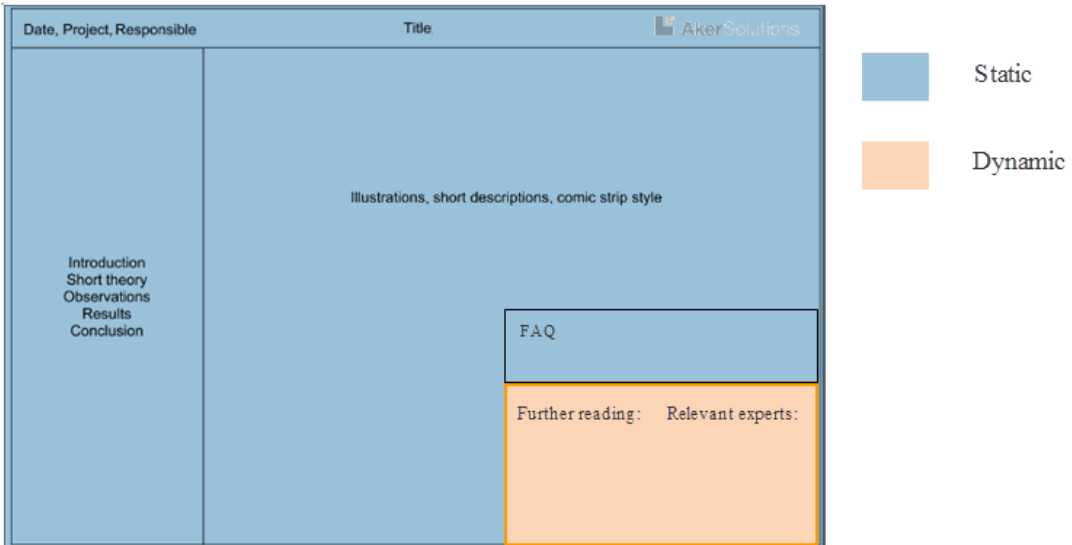


Figure 4.5: K-Brief Technical Knowledge Template, current version [10]

### 4.2.3 Example K-Brief

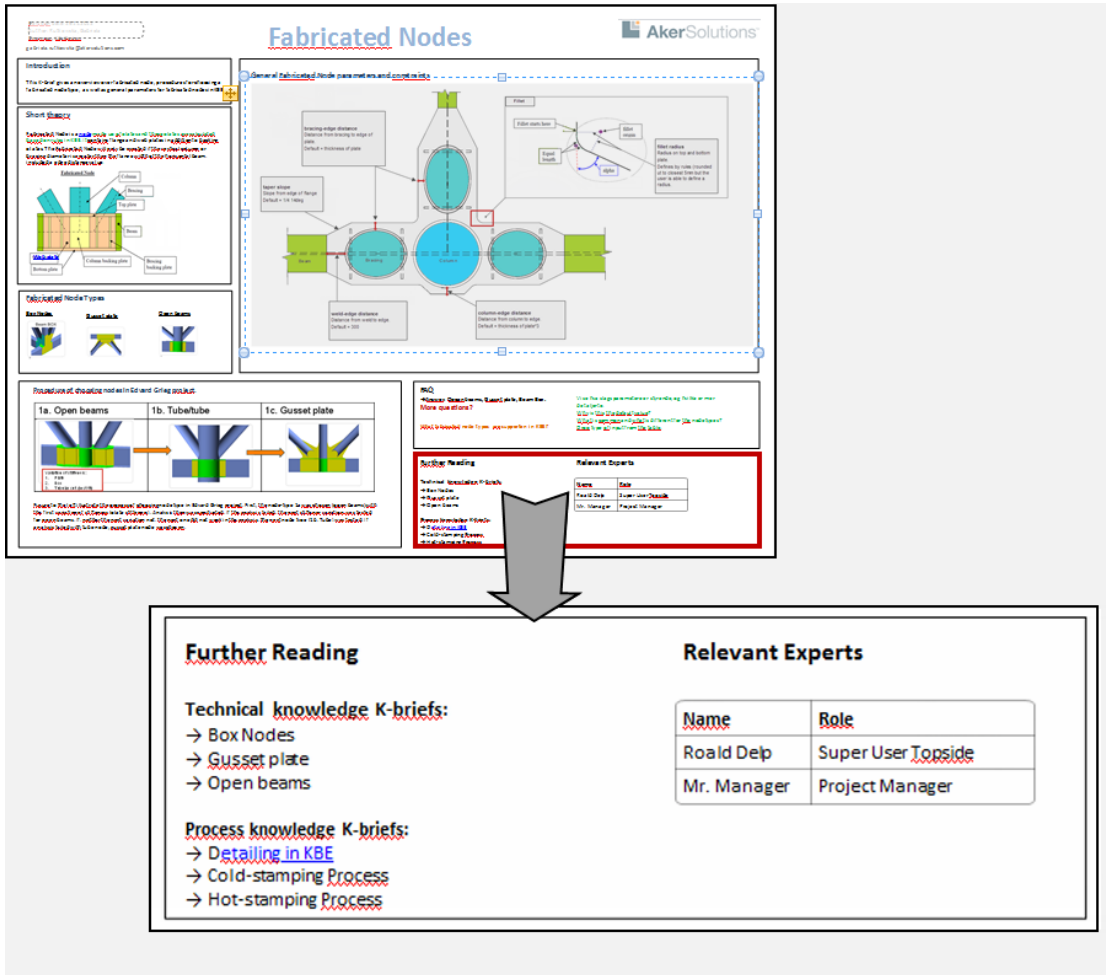


Figure 4.6: K-Brief Technical Knowledge Template, current version [10]

Figure 4.6 shows an example of how a K-brief could look like. As with the Interactive Document example, a structural node is used as an example. The

information contained is mostly figures and explanations, with some supplementary text. Note also the "Further Reading"/"Relevant Experts" section, which guides the reader to further sources of knowledge, should he or she wish to know more about the given topic. For more example K-Briefs, see the appendix.

#### 4.2.4 Implementation

##### Microsoft OneNote and SharePoint

Currently, K-brief prototypes are being implemented by using Microsoft OneNote, part of Microsoft Office. OneNote can be thought of as a digital notebook for collecting notes and information in one place. [10]

OneNote has a search engine and supports sharing of notebooks between users. It is possible to search for words in images as well as in the text. Notebooks can be organized into hierarchies, which is useful for making K-briefs that focus on increasingly specific topics. [10]

OneNote also supports hyperlinks, which can point to other OneNote documents, as well as other files, documents or webpages. [10]

K-briefs can be stored on relevant SharePoint sites. For every new project that starts, K-briefs will be read and stored on the projects SharePoint site if deemed relevant. [10]

##### Alternatives

As the amount of available K-Briefs increases, other ways of storing and managing them may become necessary. Some kind of web-based solution utilizing a relational database and a fast search engine may become attractive if we want to search and navigate in a large volume of K-Briefs.

### 4.3 Addition of Context-Aware Elements

Now we have seen to available ways of visualizing information that is important in a project/engineering context. Now, let's see some examples of how

this context can be further utilized to improve these information visualization methods.

### 4.3.1 Interactive Documents

Interactive documents are inherently context-aware to a certain degree, as they use data from a model's current situation. They keep a log of changes in the current model, and they keep a record of the model's current dimensions, to name a couple of examples. This is a very simple interpretation of context-awareness though. If we want to define interactive documents as what we now think of as a context-aware application, we should include some more features. Note that here, I will reference elements of the Interactive Documents explained in the appendix.

#### General Tab

In the general tab (see appendix and Figure 4.7), we can include additional information of the people who have been involved in creating or editing the model. Aker Solutions use Microsoft Lync for instant messaging, audio and video conversations. If we include the current Lync-status of the people involved with the model, as well as a super user, this allows people to contact each other and ask questions regarding decisions taken when creating/editing the model.



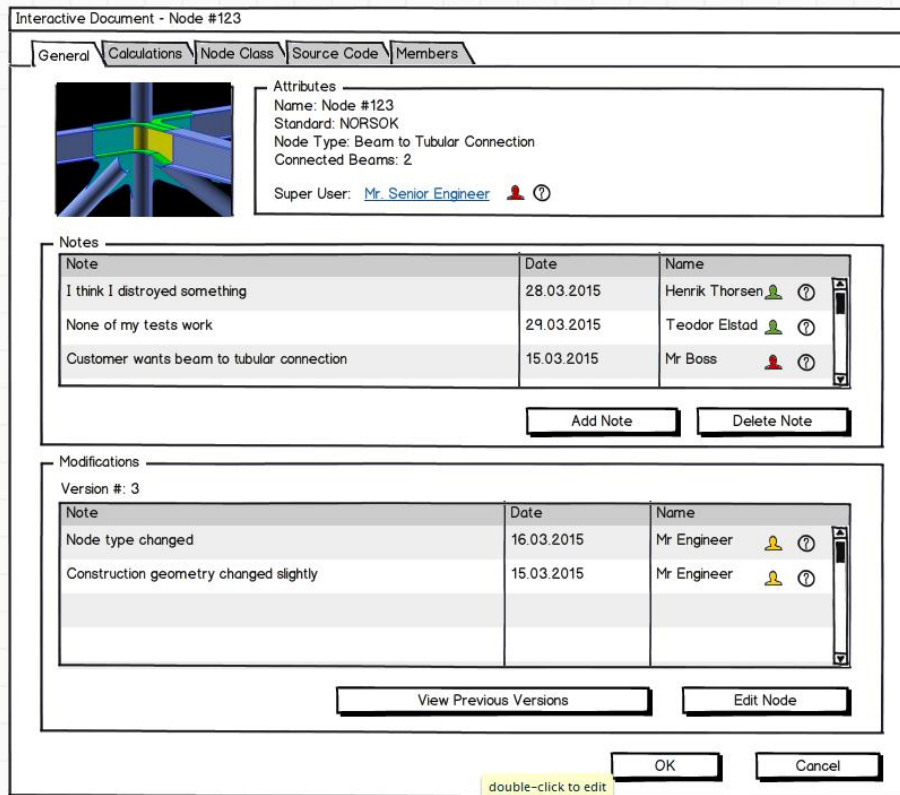


Figure 4.7: Modified "General" Tab [9, 10]

### Calculations Tab

Like in the general tab, we can add information regarding which user has defined important input for the calculations (Figure 4.8).

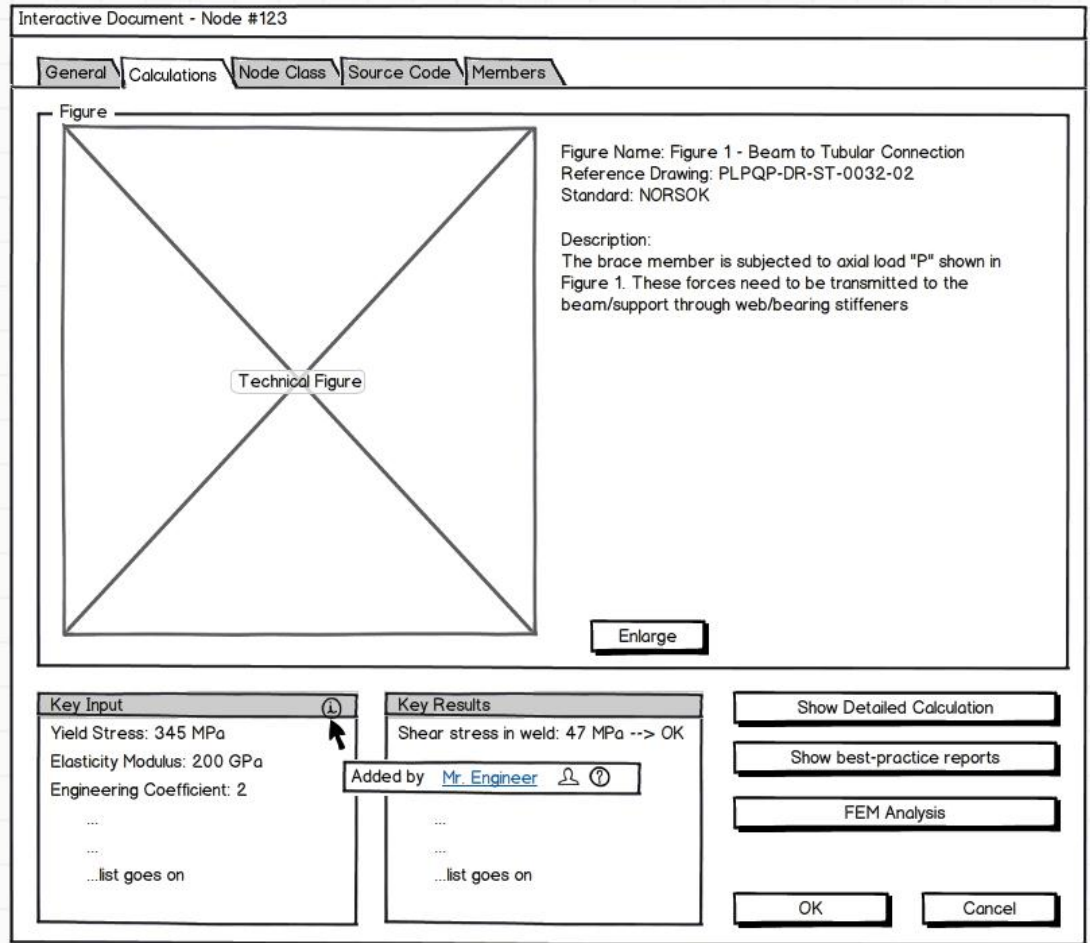


Figure 4.8: Modified "Calculations" Tab [9, 10]

## Rules

Another aspect of interactive documents where we can improve context-awareness is the rule explanation feature. Here we can include data regarding who has added or edited the rule, and their status (Figure 4.9).

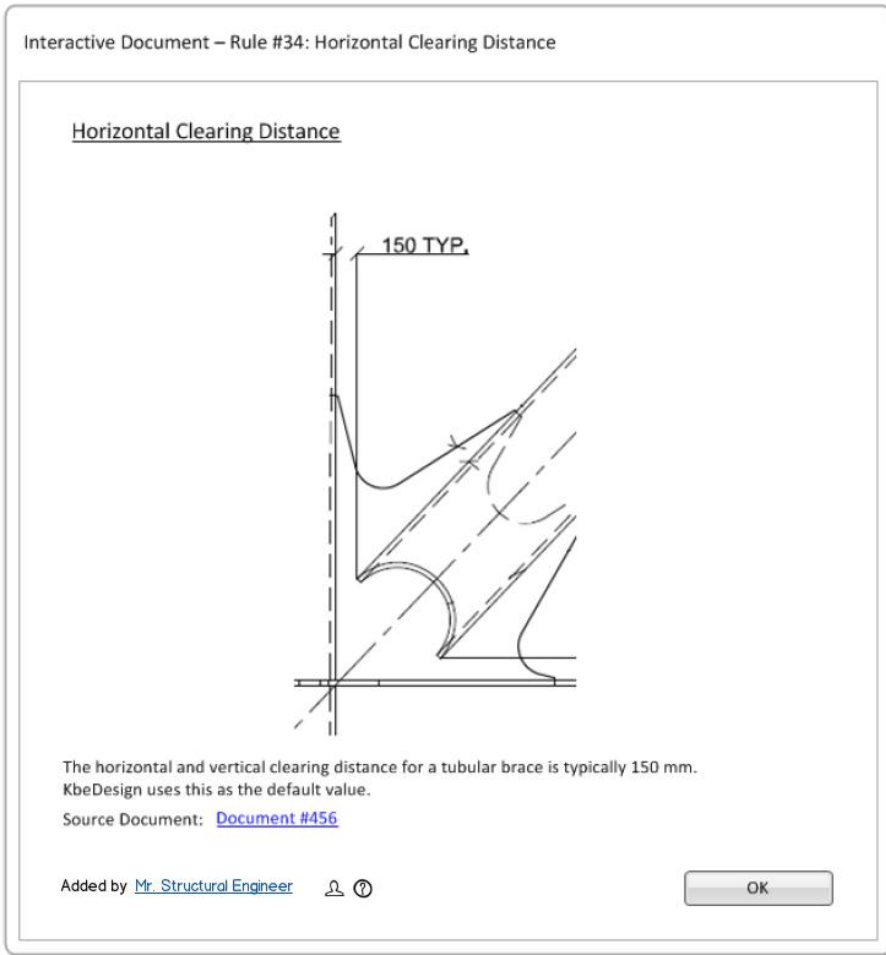


Figure 4.9: Modified Rule Explanation [9, 10]

### Support Requests

If someone has questions or concerns regarding a node, a super user can be contacted for support by clicking the "?"-sign that can be found in the appropriate

tabs. The support request is automatically filled out with the model name and additional context information. The user can also attach a screenshot and select the importance level of the support request (Figure 4.10) [9, 10].

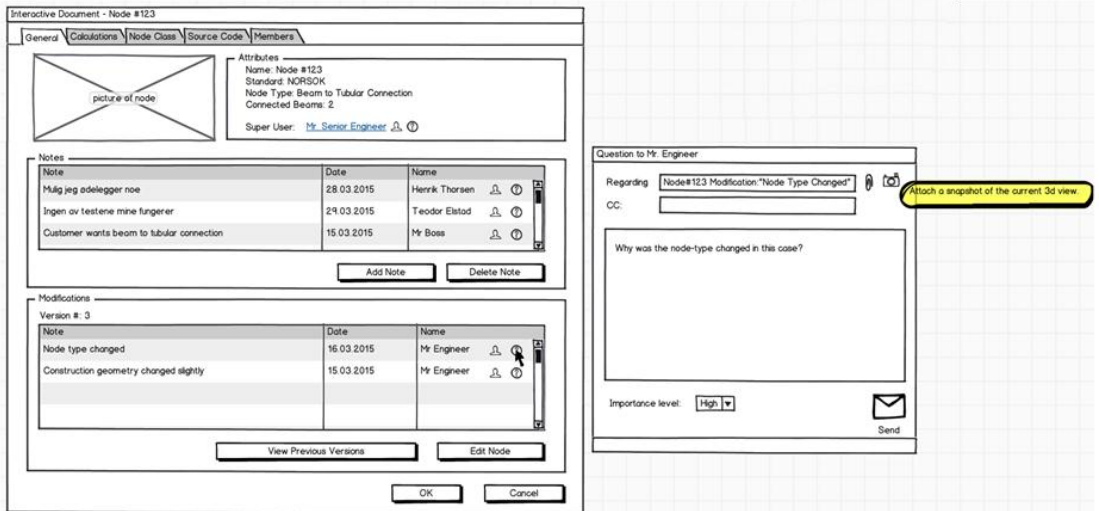


Figure 4.10: Support Request [9, 10]

### 4.3.2 K-Briefs

I will discuss how K-Briefs and context-aware applications can be combined later in this report. See the "Discussion" chapter.

## Chapter 5

# Comparison: Interactive Documents and K-briefs

In the last chapter we looked at two different ways of presenting knowledge. In this chapter I will compare these two formats, and discuss their advantages and disadvantages.

### 5.1 Value Added from the Interactive Document

The Interactive Document is mostly useful in the execution phase of a project. One of the greatest strengths of the Interactive Document is that it is quick, easy to access and it *is in context*. It is able to address specific questions that the user might have about a given model. In other words: the Interactive Documents is meant to be applied to specific *instances* of models used in AML or PDMS. When an engineer is looking at a specific model instance, he/she can find pretty much anything he/she is looking for.

While working with a model, users can easily keep track of the development of the model through the modification history. This enables users working on the same models to be aware of and understand changes that other users might have made. It is also possible to contact other users who have been involved in the work on the model.

An important part of engineering is to *verify* that your product will work under the specified load conditions. The Interactive Document can perform such verification checks via integration with either MathCAD or KBeJOINT (an in-house calculation application used and developed by Aker Solutions).

If the user is curious about the dimensions of the model, he/she can get an explanation with a single click of a button in the "Dimensions" tab. There's no need to search for information or documentation regarding the given model type. In addition to having the rule explained, you also by default get a working example of the rule demonstrated for you - it is easy to check if the dimensions of the model adhere to the rules shown in the explanation (see the appendix for more information).

## 5.2 Value Added from K-Briefs

K-Briefs can be relatively easily integrated into existing workflows, instead of the other way around. They are intended to work as "carriers" of knowledge from development to delivery of a product, and can be used as reference material later, for example when recording lessons learned from a problem or project. In addition to showing knowledge, the K-Brief can be used as a format to record knowledge captured in meetings between knowledge engineers and domain experts [10].

Another useful feature for knowledge capture is that a knowledge engineer can prepare K-briefs in advance (preferably printouts), and bring them to the meetings. During the meeting, the data in the K-Briefs can be discussed with domain experts. A knowledge engineer can even bring blank or semi-complete K-briefs, and fill out the missing pieces with the help of the domain experts. After the meeting, the knowledge engineer can create a complete, electronic version of the K-brief [10].

After knowledge has been captured, it has to be correctly implemented in KBE software. The software construction phase consists of detailed design, coding and testing iterations, and culminates in a releasable product. K-Briefs can be a great source of reference material in this phase [10].

Even though software development has begun, the knowledge that is to be implemented may not be completely clear. Therefore, in addition to be good base reference material for programmers, K-briefs can also be a useful reference

tool when it is necessary to discuss knowledge details with customers or other developers [10].

### 5.3 User-friendliness

If the user has questions about a certain dimension in a model, K-briefs can be considered too general, since they currently only describe *types* of models, not specific *instances*. Currently, a K-brief can't directly explain and demonstrate to a user why a specific dimension of a specific model has a given value. The user would have to cross-reference the dimension values with the principles explained in the K-brief and check the calculations manually. If interactive documents are available, however, checking the validity of a model's dimensions only takes a few clicks.

### 5.4 Implementation

One of the disadvantages of the Interactive Document is that it, at least in its current form, requires a lot of "tailoring". For nodes, an interactive document might contain certain tabs, each with a certain type of content, but if we were to make interactive documents for access platforms, the layout and design of the interactive document might be different from the layout used for constructional nodes. Implementing interactive documents currently requires development of a new layout for each type of model, unless we can come up with a "one size fits all" interactive document layout which is more general and suits most model types.

Another factor to consider when implementing the Interactive Document is data sources. Interactive documents need to get their data from somewhere. This requires databases that contain rules, documents, figures and images to be made. One can argue whether this is just a necessary evil to accomplish a greater cause, or just a disadvantage, but it is definitely something that requires a lot of time and resources to accomplish.

K-briefs, on the other hand, are very simple to implement. Remember, they are basically A3-sheets. In their digital form, they can contain hyperlinks to other

relevant K-Briefs. Hyperlinks are simple to implement in OneNote and other rich document formats.

## 5.5 Conclusion

After comparing these two methods of visualizing knowledge, because of their differing advantages and disadvantages, I am left with the impression that they are not really competing with each other. The goals of each knowledge visualization method are certainly overlapping to some degree, but each of the methods/formats has useful elements that the other lacks. In an ideal world, where time and resources are infinite, I would suggest to implement both the Interactive Document and the K-brief simultaneously. This is probably unrealistic, though. It is more likely that Aker Solutions will have to prioritize.

A possible solution is to take one of the formats, and make it more like the other. For example: interactive documents need tailoring for each model type. A solution to this would be to design a new, more general layout for interactive documents, that can be used for multiple (or all) types of models. A "one size fits all" template for interactive documents, as mentioned earlier. Conversely, K-briefs can be considered too general, since they only contain information about model *classes*, not *instances*. A partial solution to this problem could be to allow some interactivity in K-briefs, where the user could supply some input data into formulas and see the output. This would allow users to experiment with rules and dimensions.

The solution that I would personally recommend, is some combination of the alternative interpretation of the interactive document (mentioned in an earlier section), where interactive document elements are being added to the existing user interfaces in AML and/or PDMS, in addition to implementing K-briefs. This allows for incremental improvements in AML and PDMS features, as well as having K-briefs available. This way we get quick and context-aware help from the interactive document, as well as being able to use K-briefs in more collaborative settings like knowledge acquisition.

For the rest of this thesis, I will keep the main focus K-Briefs. This is due to several reasons, the most important one being that Aker Solutions are toning down research on interactive documents. Since we already know that the K-Brief is going to be used, any results we can find that are relevant for the K-Brief



can be immediately useful.



## Chapter 6

# Context Structuring and Modeling

Now that we have some idea of what kind of information we are interested in displaying, how do we structure this information? In this section I will focus on how to model and structure the information behind the two different visualization methods, rather than the presentation of the information itself. As we shall see in this section, using ontologies to model context seems to be the best choice, and we will explore how this can be done.

### 6.1 Ontology Crash Course

#### 6.1.1 The Basics

While writing this thesis, I have assumed that the reader possesses a fair understanding of ontologies and semantic web technologies. However, should that not be the case, I have tried to create a simple introduction to the topic here.

Originally, "ontology" refers to a major branch of philosophy or metaphysics. It is the philosophical study of *being*, *becoming*, *existence* or *reality*. Ontology deals with questions concerning what entities exist or can be said to exist,

how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences [11]. In computer science and knowledge management, ontology is usually defined as "the shared understanding of a domain, which is often described as a set of entities, relations, functions, axioms and instances".

Since the mid-1970s, researchers in the field of artificial intelligence (AI) have recognized that capturing knowledge is the key to building large and powerful AI systems. AI researchers argued that they could create new ontologies as computational models that enable certain kinds of automated reasoning. In the 1980s, the AI community began to use the term ontology to refer to both a theory of a modeled world and a component of knowledge systems. Some researchers, drawing inspiration from philosophical ontologies, viewed computational ontology as a kind of applied philosophy [12].

Common components of ontologies include (list found in [12]):

- Individuals (instances or objects)
- Classes
- Attributes of classes or instances
- Relations
- Function terms (complex structures formed from certain relations that can be used in place of an individual term in a statement)
- Restrictions (formally stated descriptions of what must be true in order for some assertion to be accepted as input)
- Rules (statements in the form of "if A, then B" that describe logical inferences that can be drawn)
- Axioms (assertions, including rules, in a logical form that together comprise the overall theory that the ontology describes in its domain of applications)
- Events (the changing of attributes or relations)

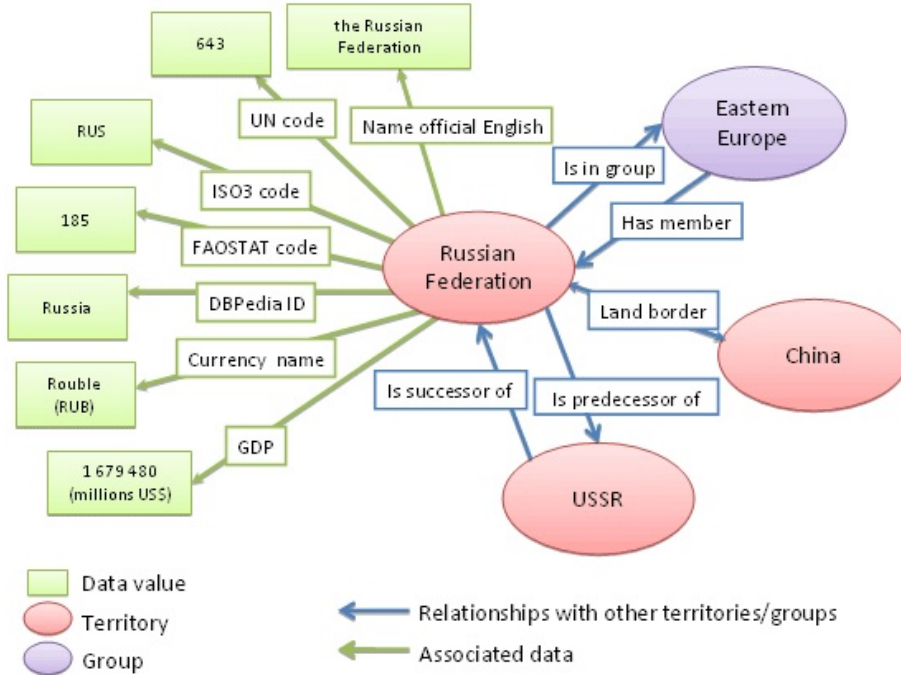


Figure 6.1: A simple ontology example [13].

### 6.1.2 Web Ontology Language (OWL)

There are many available ways to express ontologies. The most commonly used method is the Web Ontology Language, or just "OWL". "OWL" is a so-called *recursive acronym*, which stands for "OWL: Web ontology Language". OWL is a family of knowledge representation languages used for authoring ontologies. OWL is endorsed by the World Wide Web Consortium (W3C) and has attracted academic, medical and commercial interest [14].

OWL is characterised by formal semantics and RDF/XML-based serializations for the semantic web. So when you get to the bottom of it, OWL is basically expressed in XML. This is one of the main reasons for why ontologies written in OWL readily support interoperability and knowledge sharing, since XML is one

of the most widely used methods for representing data structures today.

## 6.2 Ontology Models

Of all these different ways to model context, Ontology Models are generally considered to be the best choice [2, 4, 10]. There are several reasons for this, including:

- *Knowledge Sharing*: Using ontologies enables computational entities to have a common set of concepts, a "common language", while interacting with each other [15].
- *Logic Inference (Semantic Reasoning)*: By using the relations between entities in an ontology, context-aware applications can exploit various reasoning mechanisms to deduce high-level, conceptual context from low-level, raw context. Context-aware applications can also check and resolve inconsistent context information [15].
- *Knowledge Reuse* Once an ontology has been defined, it can be reused by other applications. By allowing our context-aware applications to reuse well-defined ontologies from different domains, we can compose large-scale ontologies without having to start from scratch [15].

When designing ontologies, it is useful to have some guidelines to go after. In [4], we are presented with some requirements and goals that we should seek to satisfy when designing a context ontology:

- *Simplicity*: The used expressions and relations should be as simple as possible to simplify the work of application developers.
- *Flexibility and extensibility*: The ontology should support the simple addition of new context elements and relations.
- *Genericity*: The context should not be limited to special kinds of context atoms, but should rather support different types of context.
- *Expressiveness*: The ontology should allow for describing as many context states as possible in arbitrary detail.

When using ontologies to model context, these requirements should be satisfied.

One final note: though ontologies are widely considered to be the best context modeling technique, one should always take the requirements of the given application into consideration, and consider all suitable options before choosing a context model [2].

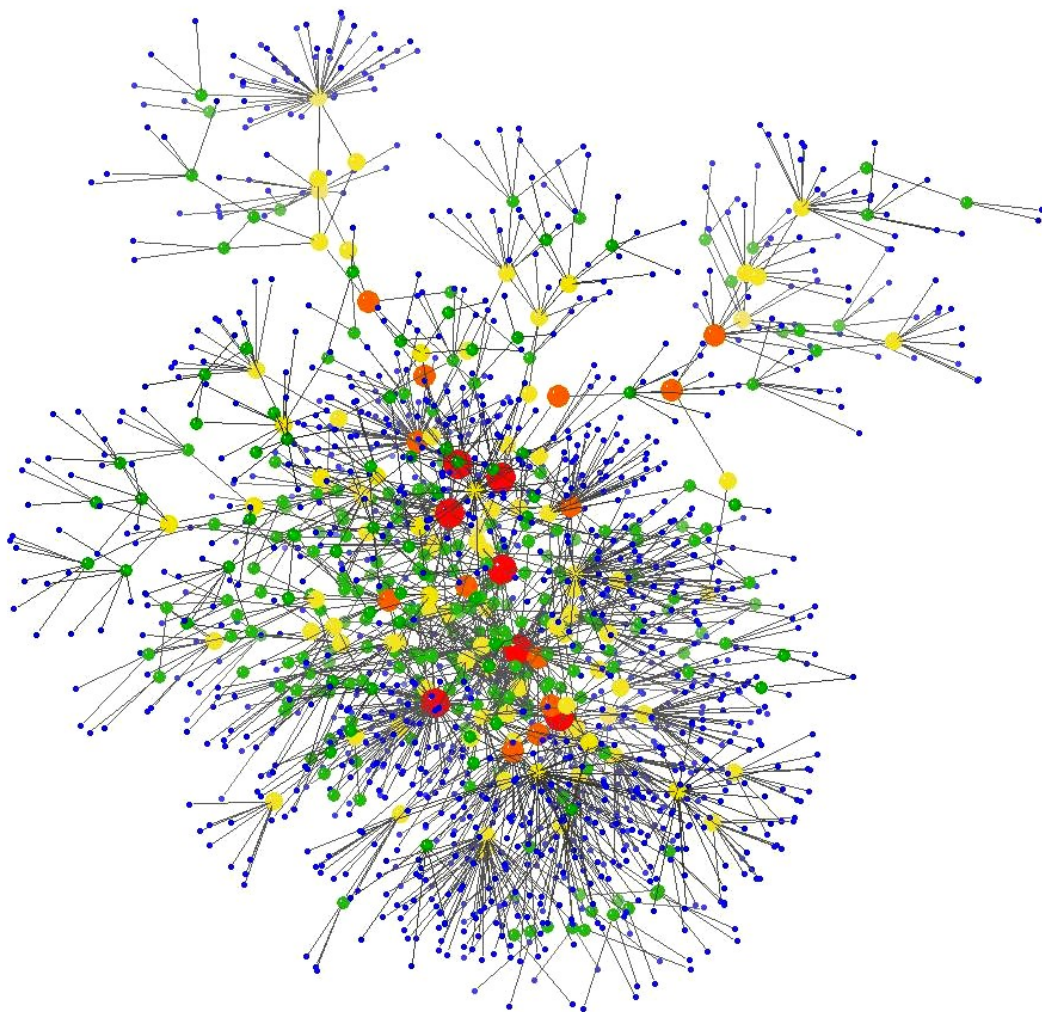


Figure 6.2: A rich ontology [16]



### 6.2.1 Example Ontology-based Context Models

#### CONON

CONON simply stands for Context Ontology. It is presented in [15], where the goal is to address critical issues including formal context representation, knowledge sharing and logic based context reasoning (semantic reasoning).

In [15], before going on to describe CONON, a few other context modeling examples are given. Attribute-value tuples, web-based models, relational database models and first-order predicate models written in DAML+OIL are mentioned, but they are all considered inferior to ontology models, in spite of supporting formal context modeling and some degree of reasoning. What they lack is support for formal knowledge sharing and proof of efficient reasoning capabilities when applied to resource-constrained devices or applications [15].

CONON is intended to model context in *pervasive* (norwegian: "gjennomtremgende") computing environments. In other words, it is not a context model tailored for a specific work environment. The term *pervasive* refers to CONON's goal of expressing context in many types of situations encountered in our daily lives, both work-related and otherwise.

In CONON, context information is grouped into the categories *location*, *user*, *activity* and *computational entity*. These entities help form the upper ontology, which can be extended into domain-specific ontologies. The upper ontology is a high-level ontology which captures general features of basic contextual entities (see Figure 6.3). Domain-specific ontology is a collection of ontology sets which define the details of general concepts and their features in each domain [15]. A domain ontology is depicted in Figure 6.4.

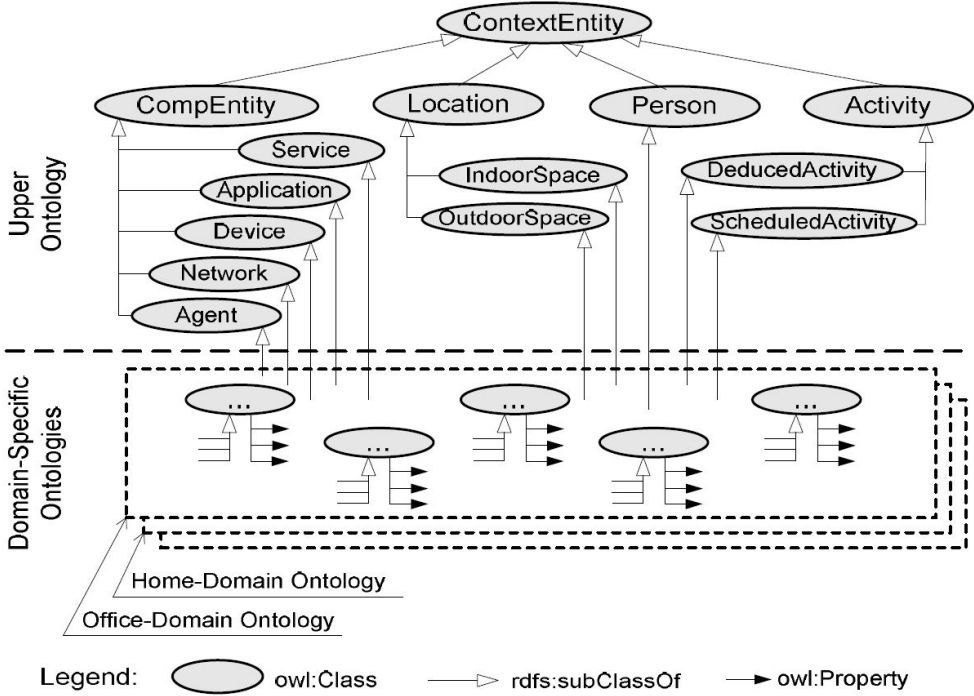


Figure 6.3: Partial definition of the CONON upper ontology [15]

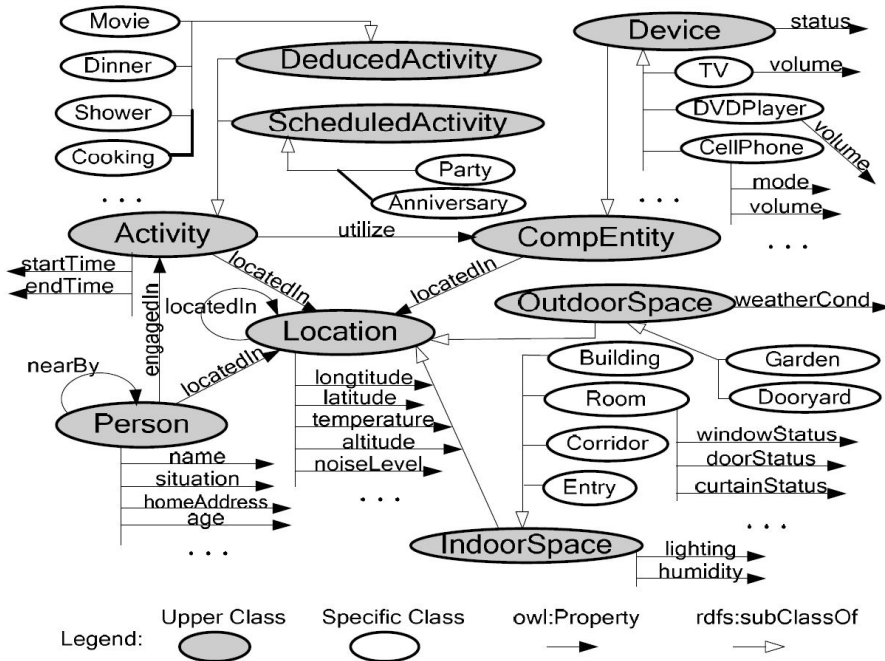


Figure 6.4: Partial definition of a specific ontology for a home domain [15]

The context model is implemented in OWL. The context "categories" are represented by the abstract OWL-entities *Person*, *Activity*, *CompEntity* and *Location* as well as a set of abstract sub-classes. Attributes are represented using *owl:DatatypeProperty*, and relations are represented using *owl:ObjectProperty*. The built-in OWL property *owl:subClassOf* are used to hierarchically structure sub-class entities, and to enable extensions. Figure 6.5 shows how these context categories are represented in OWL.

```

<owl:Class rdf:ID="ContextEntity"/>
<owl:Class rdf:ID="Location">
  <rdfs:subClassOf rdf:resource="#ContextEntity"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="longitude">
  <rdf:type rdf:resource="FunctionalProperty"/>
  <rdfs:domain rdf:resource="Location">
  <rdfs:range rdf:resource="xsd:double">
</owl:ObjectProperty> ...
<owl:Class rdf:ID="IndoorSpace">
  <rdfs:subClassOf rdf:resource="#Location"/>
  <owl:disjointWith rdf:resource="#OutdoorSpace"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="locatedIn">
  <rdf:type="owl:TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Entity"/>
  <rdfs:range rdf:resource="#Location"/>
  <owl:inverseOf rdf:resource="#contains"/>
</owl:ObjectProperty> ...

```

Figure 6.5: Partial OWL-serialization of the upper CONON-ontology [15]

### 6.2.2 Context-Driven Information Access in Aker Solutions

#### Aker Solution's Context Ontology

A problem for many engineers and other workers in large, knowledge-intensive companies is information overload. In [10], the proposed solution for this problem is employ context-driven information access through the use of ontologies.

The idea is to let each user be automatically equipped with all the information required for his/her specific role and task. One approach to accomplishing this is to filter the available information by the user's role and context, in addition to providing different means of access such as contextualized direct search, information spaces, and information analysis means [10].

Important sub-goals to accomplish this are also listed in [10]. I have included a few of them here, since they appear relevant to how we can structure and use context information:

- Integration of informal models (K-Briefs and/or interactive documents) into user contexts
- Definition of relevant contexts for user interaction with the informal model
- Context-based data access
- Role definition and management

How we can solve these subtasks depend heavily on how we choose structure information and model context. Assuming that these tasks are accomplished, what we will be rewarded with is context-specific support for specific tasks in terms of information, reports, tools and easy localization of relevant documents without being forced to initiate a typical search process, which in many companies can be unnecessarily difficult [10, 17].

Aker Solutions has chosen to use an ontology-based context model, written in OWL. As stated earlier, this way of modeling context has many advantages: great expressiveness, support for semantic interoperability so that knowledge can be exchanged and understood across multiple systems and domains, and semantic reasoning to be used by automated processes [10].

In [18], some requirements for a context ontology are stated: exploitation of ontology has to result in an efficient and convenient context-aware source of information regarding the entire business process behind CAD modeling. All the activities, actors and documentation behind a project has to be mapped to the ontology. On the other hand, all rules, sources and constraints employed in automatic CAD modeling have to be transparent and available for interested users.

As we can see from their context ontology, Aker Solutions have used the definition of context from [3]. Context is divided into the categories *activity*, *location*, *time* and *individuality*. The *relations* category has been omitted though.

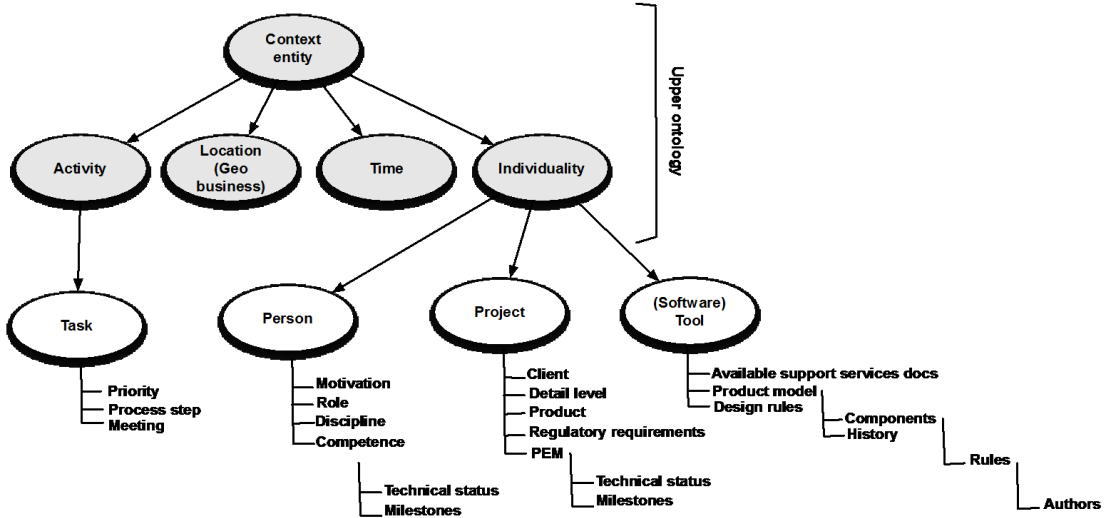


Figure 6.6: Aker Solutions' Context Ontology [10]

The ontology needs to include all entities required to define a given KBE solution and thus to provide the background knowledge to mechanical engineers. Important elements of this knowledge are mentioned in [18]:

- The product structure and a geometrical description
- KBE entities like rules, constraints, formulas etc.
- Knowledge Entities: description, author, attributes, status flag, descriptive figures etc.

### 6.3 Identifying Context

## Chapter 7

# Information Location and Mapping

We now have some suggestions for how to display relevant information in engineering contexts. We also know how we can structure this information. Now we need to figure out how to get our information from our data sources and to the user. Where can we find the information we want, and how do we map the correct information to our applications? For example: how do we know which information to populate our interactive documents with, or which K-Brief to display?

### 7.1 Semantic Reasoning

#### 7.1.1 What is semantic reasoning?

A great deal of the answers that we seek in this chapter lie within semantic reasoning. Semantic reasoning is also a great part of what makes ontology-based context models so appealing. Semantic reasoning and ontologies are so tightly interconnected that one can hardly have one without the other - together they make up a great combination of expressing logical connections and then

utilizing these connections. Using semantic reasoning, we are able to infer logical consequences from a set of asserted facts or axioms. This set of asserted facts/axioms can come from an ontology. If we combine this ontology with a *semantic reasoner* (also known as an *inference engine*) we can accomplish a couple of important things: checking the consistency of context, and deducing high-level, implicit context from low-level, explicit context [15]. Especially the latter will be important to us. This is what allows us to put together every little piece of context information that we can find about a situation, and put it together into a clear image of what a user is doing or trying to do in a given situation, and how the application can accommodate the user in this situation.

Let's use a smart phone scenario to explain the role of semantic reasoning in context-aware computing. We have a smart phone that can adapt to a user's current situation. By defining preference profiles, users can define customized behavior patterns for the phone. For example, when the user is sleeping in his bedroom, incoming calls are forwarded to voicemail. When the user is cooking in the kitchen or watching TV in the living room, the ringing volume is turned up. When the user is having dinner with his family in the dining room, the phone is set to vibrate mode. What we see here are examples of high-level context. This level of context can not be directly acquired from the phone's physical sensors. It has to be logically inferred from low-level data. The phone's sensors provide low-level context data such as physical location, time and environmental information (temperature, ambient noise level) [15]. This logical inference is semantic reasoning.

There are plenty of available semantic reasoners, many of whom support several types of languages and rules. It is common for semantic reasoners to support reasoning with ontologies written in OWL.

Some ontology languages are more suitable for reasoning applications than others. Usually there is a trade-off between expressiveness and reasoning capabilities when choosing an ontology language. Let's consider the OWL-family of languages, for instance: OWL Lite is the simplest language in the OWL-family, but it is also the easiest one to reason with. OWL DL is more expressive, while still retaining the availability to apply practical reasoning algorithms. Lastly, we have OWL Full, which is so expressive that it is undecidable, and no reasoning software is able to perform complete reasoning for it [14].



### 7.1.2 Example Application: CONON

Again, let's look at the CONON-framework to find some examples of how semantic reasoning can be used in a context-aware application. The reasoning tasks in CONON can be grouped into two categories: ontology reasoning using description logic, and user-defined reasoning using first-order logic [15].

A first-order predicate has three fields: a subject, a verb and an object. For example, let's express the physical location context "Wang is located in the bedroom". Expressed as a first-order predicate this becomes "(Wang, locatedIn, Bedroom)" [15].

Description logic (DL) allows specification of a terminological hierarchy using a restricted set of first-order formulas. The equivalence of OWL and description logic allows OWL to exploit the considerable existing body of DL-reasoning to fulfill important logical requirements. These requirements include concept satisfiability, class subsumption, class consistency and instance checking [15].

Transitive-Property	$(?P \text{ rdf:type owl:TransitiveProperty}) \wedge (?A ?P ?B) \wedge (?B ?P ?C) \Rightarrow (?A ?P ?C)$
subClassOf	$(?a \text{ rdfs:subClassOf } ?b) \wedge (?b \text{ rdfs:subClassOf } ?c) \Rightarrow (?a \text{ rdfs:subClassOf } ?c)$
subProperty-Of	$(?a \text{ rdfs:subPropertyOf } ?b) \wedge (?b \text{ rdfs:subPropertyOf } ?c) \Rightarrow (?a \text{ rdfs:subPropertyOf } ?c)$
disjointWith	$(?C \text{ owl:disjointWith } ?D) \wedge (?X \text{ rdf:type } ?C) \wedge (?Y \text{ rdf:type } ?D) \Rightarrow (?X \text{ owl:differentFrom } ?Y)$
inverseOf	$(?P \text{ owl:inverseOf } ?Q) \wedge (?X ?P ?Y) \Rightarrow (?Y ?Q ?X)$

Figure 7.1: OWL ontology property types and how they affect reasoning [15]

A simple example of ontology reasoning is reasoning with physical location. If Wang is currently located in his bedroom, for example, which is in turn part of his home building description logic can be used to conclude that Wang (or at least his phone) is located in his home building. This is possible since the *locatedIn* property is a *transitive* property (see Figures 7.1 and 7.2 [15]).

INPUT	DL Reasoning Rules	$(?P \text{ rdf:type owl:TransitiveProperty}) \wedge$ $(?A ?P ?B) \wedge (?B ?P ?C) \Rightarrow (?A ?P ?C)$ $(?P \text{ owl:inverseOf } ?Q) \wedge (?X ?P ?Y)$ $\Rightarrow (?Y ?Q ?X)$
	Explicit Context	<pre> &lt;owl:ObjectProperty rdf:ID="locatedIn"&gt;   &lt;rdf:type="owl:TransitiveProperty"/&gt;   &lt;owl:inverseOf rdf:resource="#contains"/&gt; &lt;/owl:ObjectProperty&gt; &lt;Person rdf:ID="Wang"&gt;   &lt;locatedIn rdf:resource="#Bedroom"/&gt; &lt;/Person &gt; &lt; Room rdf:ID="Bedroom"&gt;   &lt; locatedIn rdf:resource="#Home"/&gt; &lt;/ Room&gt; </pre>
OUTPUT	Implicit Context	<pre> &lt;Person rdf:ID="Wang"&gt;   &lt;locatedIn rdf:resource="#Home"/&gt; &lt;/Person &gt; &lt;Building rdf:ID="Home"&gt;   &lt; contains rdf:resource="#Bedroom"/&gt;   &lt; contains rdf:resource="#Wang"/&gt; &lt;/Building&gt; &lt;Room rdf:ID="Bedroom"&gt;   &lt; contains rdf:resource="#Wang"/&gt; &lt;/Room&gt; </pre>

Figure 7.2: Using ontology to reason about location.

In addition to ontology reasoning, we can also implement user-defined reasoning. We can create user-defined reasoning rules using first-order logic, which can help the application deduce high-level context such as *what the user is doing* [15]. See Figure 7.4 for some examples.

Situation	Reasoning Rules
Sleeping	$(?u \text{ locatedIn Bedroom}) \wedge (\text{Bedroom lightLevel LOW})$ $\wedge (\text{Bedroom drapeStatus CLOSED})$ $\Rightarrow (?u \text{ situation SLEEPING})$
Showering	$(?u \text{ locatedIn Bathroom})$ $\wedge (\text{WaterHeater locatedIn Bathroom})$ $\wedge (\text{Bathroom doorStatus CLOSED})$ $\wedge (\text{WaterHeater status ON})$ $\Rightarrow (?u \text{ situation SHOWERING})$
Cooking	$(?u \text{ locatedIn Kitchen}) \wedge (\text{ElectricOven locatedIn Kitchen})$ $\wedge (\text{ElectricOven status ON})$ $\Rightarrow (?u \text{ situation COOKING})$
Watching-TV	$(?u \text{ locatedIn LivingRoom})$ $\wedge (\text{TVSet locatedIn LivingRoom})$ $\wedge (\text{TVSet status ON})$ $\Rightarrow (?u \text{ situation WATCHINGTV})$
Having-Dinner	$(?u \text{ locatedIn DiningRoom})$ $\wedge (?v \text{ locatedIn DiningRoom})$ $\wedge (?u \text{ owl:differentFrom } ?v)$ $\Rightarrow (?u \text{ situation HAVINGDINNER})$

Figure 7.3: User-defined context reasoning rules [15].

### 7.1.3 Performance (CONON)

Something else that can be worth mentioning regarding semantic reasoning is *performance*. In [15], we can see some results from a performance experiment (see Figure 7.4). Key takeaways from these experiments are:

- Run-time performance of logic-based reasoning, not surprisingly, depends on three factors: size of the context data set, complexity of the reasoning rules and CPU speed.
- A large difference in performance between tests using different-sized data sets shows that context reasoning is a computationally intensive task
- As long as reasoning tasks are not time-critical, semantic reasoning is feasible with existing CPUs (the reasoning tasks in the experiments took between 0 and 22 seconds). From this result we can also assume that reasoning tasks will become less time consuming in the future, when faster CPUs are available.
- Rule complexity is an important factor. The user-defined reasoner featuring a small rule set greatly outperforms the OWL reasoner with a large Description Logic rule set when applied to the same set of context data.
- For time-critical applications, such as security and navigating systems, the context dataset size and the complexity of the rule set needs to be controlled to ensure that reasoning tasks can finish in an acceptable time.
- To increase performance, we can de-couple context *processing* and context *usage*. This way, context reasoning and processing can be handled by a resource-rich, centralized server, which other applications (clients) can acquire high-level context from.

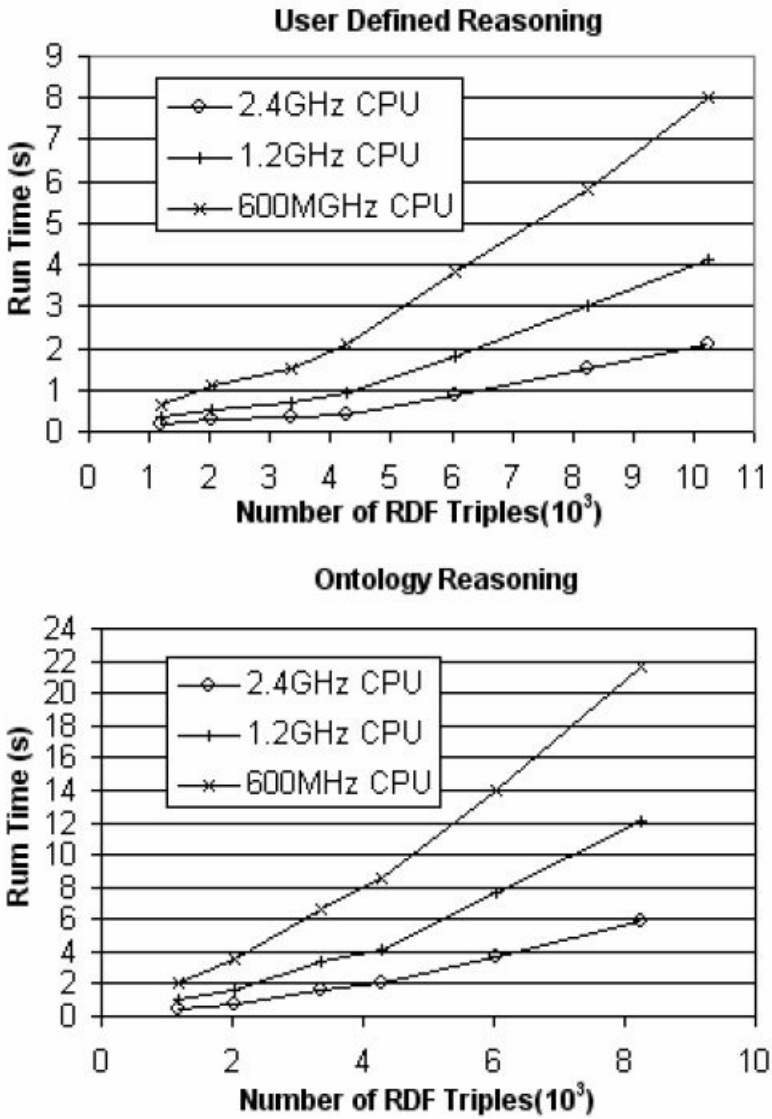


Figure 7.4: CONON Semantic Reasoning Performance. "Number of RDF Triplets" refers to the number of "subject, verb, object" sets [15].

## 7.2 Aker Solutions Use Case

### 7.2.1 Architecture

If we have a look at the Aker Solutions Use Case in [18], we can see a model of how the architecture of a context-aware application would look like.

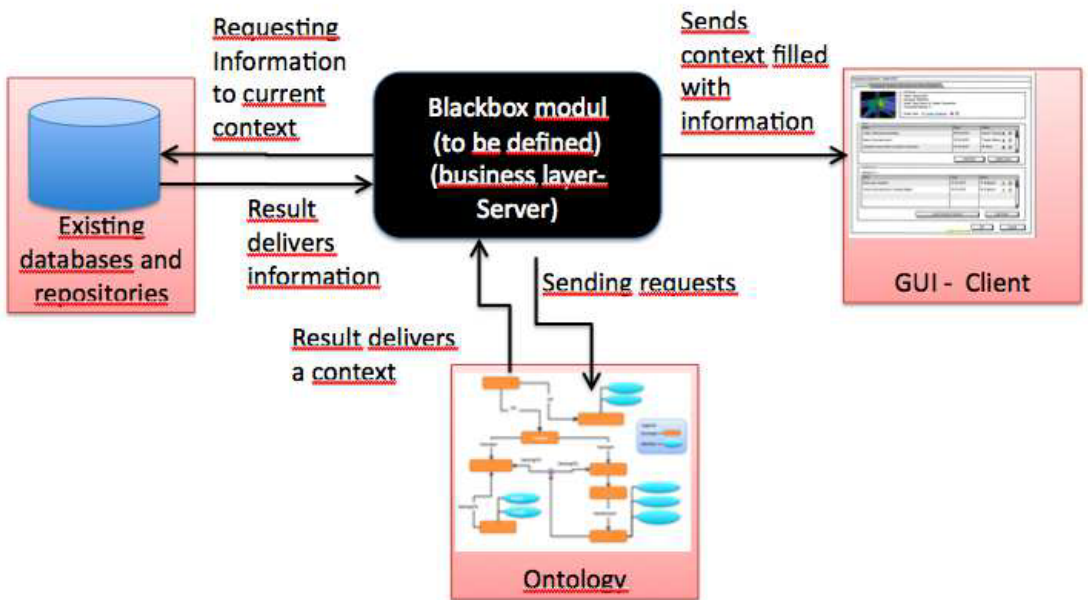


Figure 7.5: Aker Solution's architecture for a context-aware application [10]

As one can see from the figure, the details of the business layer of the architecture has yet to be worked out. Currently, it is only a black box which is assumed to handle business logic correctly. I have interpreted the architecture to work in the following way: depending on what the user is doing, it will correspond to a given context. The user may have a certain role, work on a certain project, and may be trying to accomplish a certain task. The black box collects this information and cross-references it with the ontology. This cross-referencing

allows the black box to determine the context the user is in. Having the context available, the black box queries the appropriate databases and repositories for information that is relevant to the given context. This data is then presented to the user in a suitable format.

### 7.2.2 LEAP Architecture

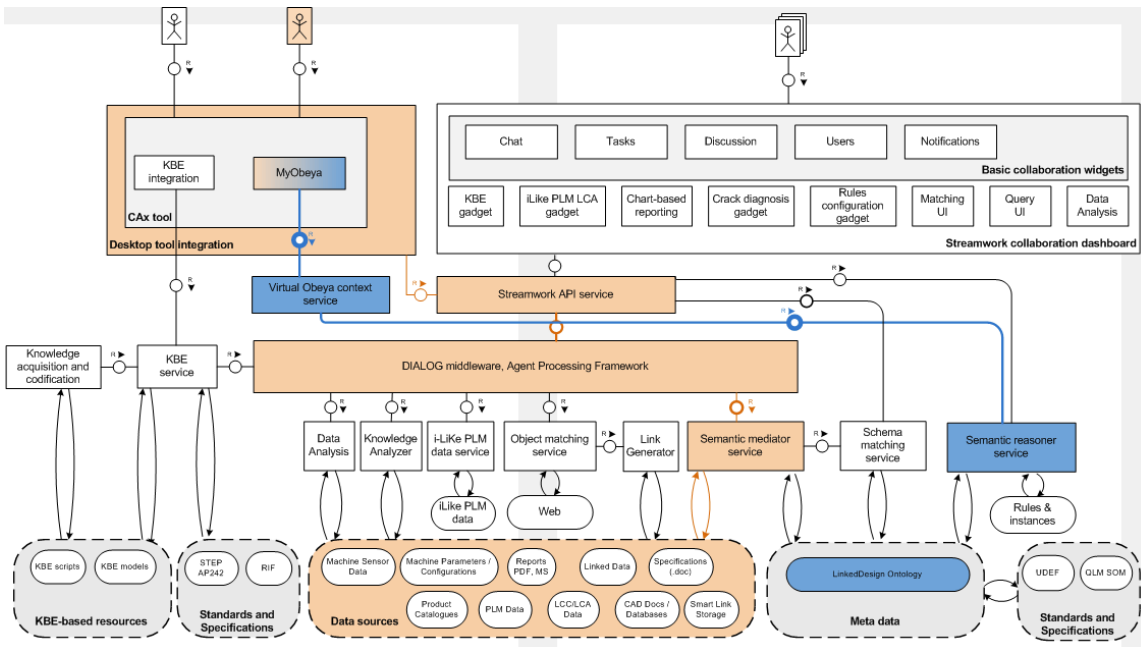


Figure 7.6: The LEAP-architecture [10]

In the previous subsection, we saw a general architecture for a system that utilizes context and semantic reasoning. Let's take a look at a more concrete example: LEAP.

LEAP (Linked Engineering And manufacturing Platform) is meant to be one of the main outputs of the LinkedDesign project. LEAP is meant to be a new IT-platform for engineers which facilitates automated engineering and design,

as well as collaboration across both disciplines and geographical distances. Figure 7.6 shows the LEAP architecture, and how different services and systems are connected in it.

Let's go through how I have interpreted this architecture. What we are going to focus on here are the services marked with blue. Let's start at the top left in the figure, where the user accesses the system. The user is using an application called MyObeya, which is sort of a virtual meeting room and collaboration application (i will not go into further detail regarding MyObeya/Virtual Obeya here). A context service registers the context of the user's interaction with the system. This context service is connected to a semantic reasoner service, which communicates with the LinkedDesign Ontology. Using the context together with the LinkedDesign ontology, the Semantic Reasoner Service is able to determine which data may be relevant to the user.

## 7.3 Interactive Document Data Sources

Implementing interactive documents is going to require some more infrastructure than what exists today. New databases containing all the data referenced in interactive documents will have to be created and maintained.

### 7.3.1 Node Classification

Interactive documents will need to correctly identify the topology of a node to determine exactly which data should be displayed. For example, it is important to know how many braces/columns a node has, and their cross-section, to know exactly which class of node we are talking about, and thereby determine what information should go into an interactive document.

Consider images, for example. Interactive documents use a lot of images and figures, for example when showing principal sketches of the different node types. To display the correct principal sketch, we need to have a database containing data sorted by node classes, each with their respective principal sketches.

One way of classifying nodes is to use data from actual AML models. In AML, each node contains data that tell us which braces/columns are present, out of an available list. Using this list of available braces/columns/beams, and making note of which columns/beams/braces are there or not, we can determine the



node configuration. From here, we can move on to checking the cross-section of each beam/brace/column that is connected to the node. Cross-referencing all of this information will give us the node class.

### 7.3.2 Calculations

Another type of data that is dependent on node classification is data regarding calculations. To display the correct calculations, we need to know the exact topology and geometric dimensions of the node. The former is related to the node class, while the latter is decided by the dimensions of the node and its members (dimension data will be discussed in the next paragraph/subsubsection). For verification calculations on nodes, it is likely that Aker Solutions will use KBeJOINT in the future. If we want calculations available for each node class, we would have to first define and implement these calculations in KBeJOINT, then create a link between interactive documents and KBeJOINT.

### 7.3.3 Node- and beam attributes

In the "Dimensions" and "Members" tabs, various model-specific data is shown. To a large degree, this data can be found in the AML models themselves. When opening an interactive document for a node, this data can be found in the models and the interactive document can be populated upon start-up. This is possible both when accessing the Interactive Document from AML and PDMS (more on this later in this section).

### 7.3.4 Rules

One of the main features of the "Dimensions"-tab, and interactive documents as a whole, is the ability to show explanations of rules behind the dimensions of a model. This would require a database containing these rules and explanations. For each rule, this database would have to contain:

- A Rule ID
- An image that helps explain the rule
- Mathematical expressions

- Supplementary text that aids the explanation

### 7.3.5 Source Code

This can be found in AML, by using the "Inspect"-function that is native to AML.

### 7.3.6 AML vs PDMS

We can access interactive documents both from AML, even though we might have to access data from AML to show the correct data in interactive documents. PDMS is coded in .NET. By using UserControls and CustomCotrols (classes in .NET), custom interfaces can be implemented in PDMS. To complete the line of communication, engineers at KBeDesign have found a way to communicate with AML from .NET by developing an .NET-application that can input commmands into the AML-console and read the resulting output. This should allow retrieval of data from AML-models, even when the user is currently in PDMS. In other words, we can access AML models from PDMS.

# Chapter 8

## Case-Based Reasoning

### 8.1 Introduction

A concept that has similarities with Knowledge Based Engineering is Case-Based Reasoning (CBR). Like Knowledge Based Engineering, Case-Based Reasoning is about reusing knowledge from previous problems to create new solutions or products.

Case-Based Reasoning is actually based on the human mind [20]. Human cognition deals with knowledge in the form of concrete examples. CBR has arisen out of research in the area of Cognitive Science, and is based on work regarding dynamic memory and the role that memories of earlier situations play in problem solving and learning. There are plenty of examples of how humans use past experiences to solve new problems. Doctors recognize symptoms in their patients and are able to make diagnoses based on experiences from other patients. Judges in legal courts use precedents from past trials. Engineers can reuse certain elements in their calculations when parts of a new system is similar to something they have worked with in the past. Figure 8.1 shows the basic principle how CBR works.

In "A tutorial on Case-Based Reasoning", Main et al. define CBR in the following way:

*"A short definition of case-based reasoning is that it is a methodology for solv-*

*ing problems by utilizing previous experiences. It involves retaining a memory of previous problems and their solutions and, by referencing these, solve new problems. Generally, a case-based reasoner will be presented with a problem. It may be presented by either a user or another program or system. The case-based reasoner then searches its memory of past cases (the case base) and attempts to find a case that has the same problem specification as the current case. If the reasoner cannot find an identical case in its case base, it will attempt to find the case or cases in the case base that most closely match the current query case.”* [19]

As mentioned above, cases are organized and stored in a case base. In addition to the case base itself, a CBR-system may include other kinds of data, like models, rules or constraints [20].

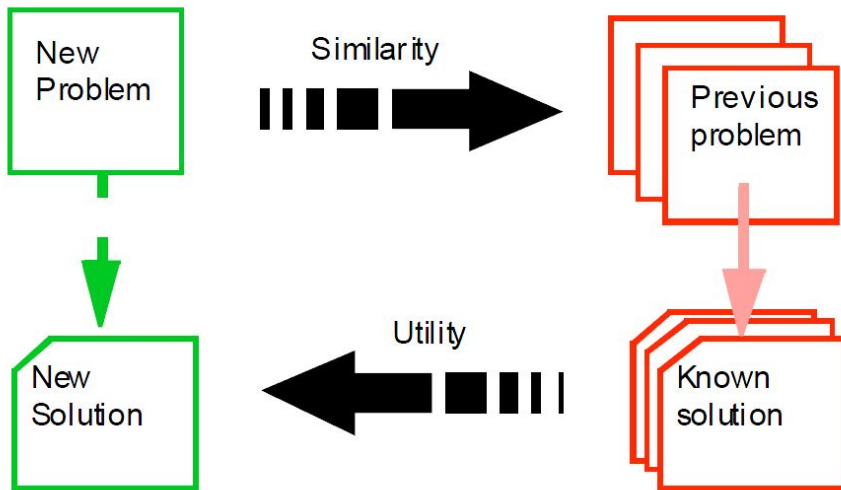


Figure 8.1: Case-Based Reasoning Illustration [20]

The key difference between CBR and KBE lies in the mechanics of how the two methodologies are implemented. In KBE applications, "knowledge" is usually stored as computational models in the form of code used to calculate geometric dimensions of structures and/or other industry products. CBR, however, takes

on a more abstract or general approach. What KBE does is to automatically reuse a predetermined previous solution, given a certain problem. CBR works a little differently. The goal is not necessarily to reuse previous solutions directly, but to compare a new problem, or "case", to previous cases. Similarities between the new and old cases are discovered, and these similarities can be used to construct a new solution. If the new case actually happens to be identical to one or more previous cases, the exact same solution can of course be reused. One of the advantages of a CBR-system is that even though a new case does not have an existing solution, previous cases might be similar enough to help guide the user in the right direction. This is called an *adaptation* phase. When adapting a solution, differences between the current and a previous case are identified, and then solution associated with the previous case is modified to take these differences into account. If the new solution that has been created is acceptable, it is retained and stored in the case base along with all the other previous cases.

CBR can be a very helpful, effective and time-saving tool for solving knowledge-intensive tasks. Due to the general nature of the CBR paradigm, possible applications are virtually limitless. Applications have already been developed in medicine, law, engineering, tech-support, communication networks, manufacturing design, finance, scheduling, language, food and many other fields, even poker [19, 21].

## 8.2 The CBR-Cycle

Case-Based Reasoning can be described as a cyclical process. The process is composed of four stages: retrieve, reuse, revise and retain. In *retrieve* stage, past cases are retrieved and compared to the current problem. In the *reuse* stage, a previous solution is adapted and reused. In the *revise* stage, the solution is tested in the real world (or simulated). Unforeseen problems with the new solution can be discovered, and the new solution is adapted to cope with these new problems. In the *retain* stage, the resulting experience from the new problem and its solution is stored in the case base for future reference [22].

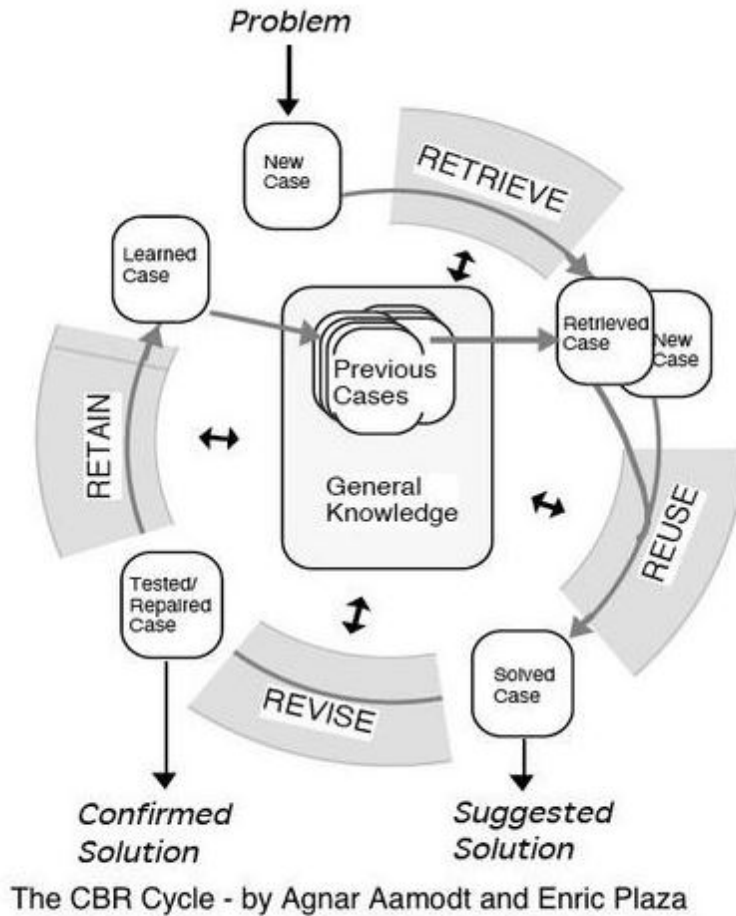


Figure 8.2: Illustration of the CBR cycle [20]

Let's use an example to illustrate this cycle with a cooking example from [23]. Fred wants to make blueberry pancakes but he has never done this before. He has, however, made plain pancakes in the past. Let's compare this process to the CBR-cycle:

- *Retrieve*: Fred retrieves his old recipe for plain pancakes. This serves as a relevant case from the past, which he can reuse.
  
- *Reuse*: Fred reuses the old recipe. The only change he has to make is to add blueberries.
  
- *Revise*: An unexpected result from the new recipe is that the pancake batter has turned blue. Fred solves this problem by making a revision to his solution: he delays the addition of the blueberries until the pancake batter has been ladled into the frying pan.
  
- *Retain*: After successfully making a fresh batch of blueberry pancakes, Fred writes down the new recipe in his cookbook for future reference.

We can also break each stage of the CBR-cycle down into a few key subtasks, shown in Figure 8.3.

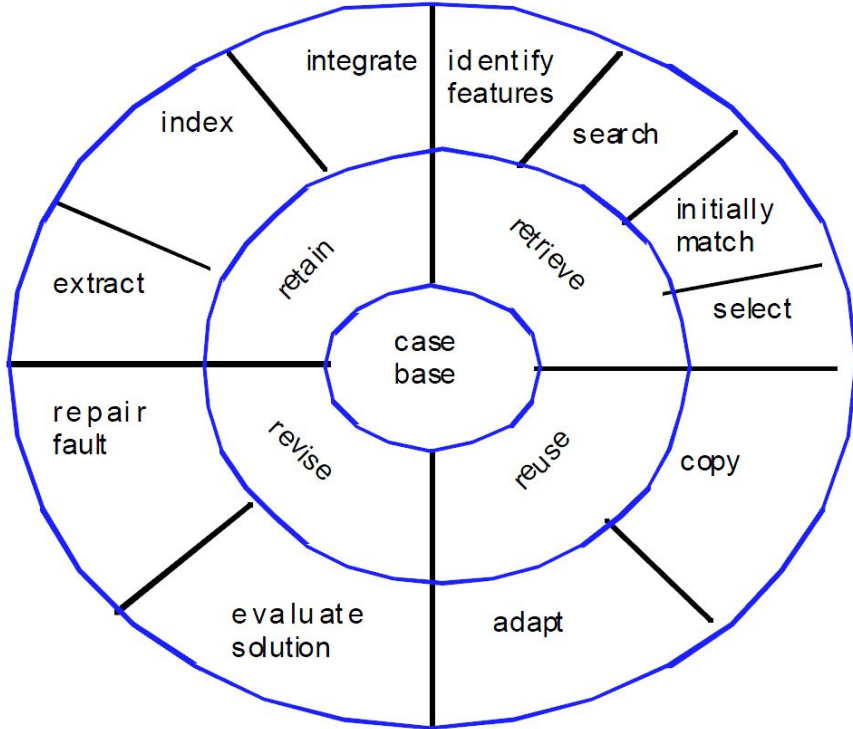


Figure 8.3: Illustration of the CBR cycle [22]

### 8.3 Case Representation

Instead of just storing knowledge as algorithms and computational rules, like in KBE, CBR store knowledge as "cases". How one chooses to define a case is entirely up to the developer. How a case is represented will vary greatly with in form and size, as it is completely dependent on the application domain. Literature on Case-Based Reasoning provide us with some guidelines for representing cases, but there is no strict system that we have to follow.

Cases representation can range from extremely simple case models like vectors,



or more composite and rich models which include several data types. Any format can work, as long as there exists a reasonable way to compare cases and determine the similarity between them. A common approach is to use a set of attributes/values for the case's problem description, and another set of attributes/values for the case's solution.

A case should represent a situation or problem, along with relevant information about the situation. However, the only *required* information in a case is a description of the *situation/problem* and the *solution*. Using only this, we can create a simple CBR-system which compares problems and proposes a fitting solution based on past experiences.

However, the more information you have about a problem and its solution, the better. Relevant information about a case can be, but is not necessarily limited to:

- A clear and concrete description of the situation
- Background information that explains the cause of the problem
- Which solution was selected (if any)
- A rationale for selecting the proposed solution
- Alternative solutions (if any), and why these were not selected
- An objective description of the result after applying the proposed solution
- A qualitative description of the result, to evaluate whether the result was a success or a failure

Note that it can be just as valuable to keep record of unsuccessful solutions as well as successful ones. Sometimes it may be just as valuable to know what *not* to do (and why).

## 8.4 Similarity Measures (Retrieval Phase)

In the retrieval phase, we need to determine which cases in the case base are most similar to the new case. In addition, we need criteria that determine if a case should be retrieved and a mechanism for how the case base is searched [15].

The actual case retrieval process varies greatly from system to system. It depends heavily on the memory model and indexing procedures used in a given case base. Retrieval methods implemented by researchers and implementers of CBR-systems are extremely diverse, ranging from a simple nearest neighbor search to the use of intelligent agents [19]. In this report, I will mention some of the most common methods.

### 8.4.1 Nearest Neighbor Retrieval

In this method, each case is given a weighted score when compared to the new case. Each case has a set of attributes. A case is chosen if it scores higher than most of the other cases in the case base. Cases are scored on their number of matching attributes. For example: if case A matches the new case on 5 attributes, and case B matches the new case on 6 attributes, case B will be considered the most relevant. Attributes may also be weighted according to their relevance. For example, attributes like "author" and "project" may have a weight of 0.25, while attributes like "part name" may be have a weight of 1 if we are working with a new case regarding a specific part. Weighting of each attribute may be defined by the system, but it can also be user defined, varying with each new case. We can express similarity scoring in nearest-neighbor retrieval the following way:

$$S(C_1, C_2) = \sum_{i=1}^n w_i \cdot s_i(C_1, C_2) \quad (8.1)$$

Here,  $S(C_1, C_2)$  is the total similarity score between the two cases  $C_1$  and  $C_2$ ,  $n$  is the number of attributes,  $w_i$  is the weight of attribute  $i$  and  $s_i(C_1, C_2)$  is the local similarity score of attribute  $i$  [20].

### 8.4.2 Inductive Approaches

In Inductive approaches we aim to determine the relative importance of features for discriminating between similar cases, such that we can structure the case base into a hierarchical structure. In other words, we determine which key attributes we can rely on to "filter" the cases in the case base. The resulting hierarchical structure may result in reduced search time [19].

### 8.4.3 Knowledge Guided Approaches

Knowledge guided approaches to case retrieval use domain knowledge to determine which features/attributes of a case will be relevant for future retrieval. In some situations, different attributes/features of a case may be important in the future. As with inductive approaches, knowledge guided indexing may result in a hierarchical case structure which is effective for searching [19].

### 8.4.4 Validated Retrieval

Validated retrieval consists of two phases. First, all cases that satisfy minimum similarity requirement are retrieved. This can be done using simple, cheap similarity measures. In the second phase, the similarity measure is refined, and more expensive similarity measures are applied to determine which of the initially retrieved cases are the most relevant for the new case [19]. Figure 8.4 shows the process of validated retrieval.

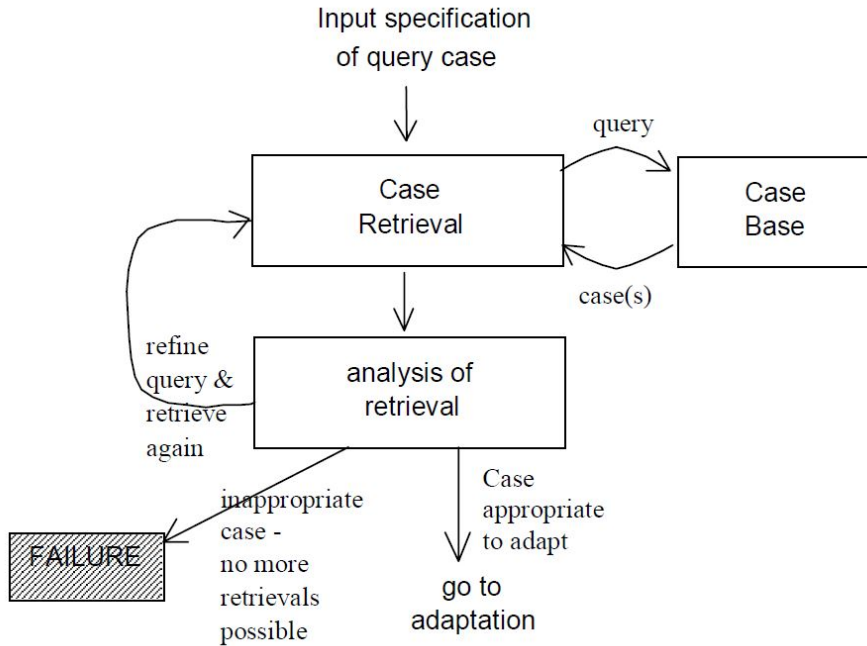


Figure 8.4: Validated Retrieval [19]

## 8.5 When should CBR be used?

CBR is a useful methodology for many types of problems and in different domains, but that doesn't mean it should always be used. In [19], we can find some criteria that should help us decide if using CBR is a good idea:

- *The domain has an underlying model.*  
If a domain does not have an underlying model, or the processes occurring in the domain are random, one cannot capture the factors leading up to success or failure in a case description, and reasoning from past cases is a futile effort.
- *There are exceptional or novel cases.*

If there are no exceptional or novel cases, little reasoning is required, and we can rather model the domain with simple rules.

- *Cases recur.*  
If similar cases are unlikely to recur in the future, there is little value in storing cases.
- *There is significant benefit in adapting past solutions*  
Reusing solutions should provide a significant difference in resources expended when compared to creating new solutions from scratch every time.
- *Relevant previous cases must be obtainable.*  
It must be possible to obtain relevant data that records the characteristics of past cases. It is important that the cases contain enough to explain the problem, its solution and in which context the problem occurred.

If the majority of these criteria are satisfied, it is likely that CBR is applicable and relevant.

The last criterium should be given some extra attention. Even if this criterium is *not* satisfied, it could be a reason to start using the CBR-approach. If cases are not presently being stored and kept for future reference, there is no reason not to start doing this if the rest of the criteria imply that using CBR could be useful. Satisfying the last criterium could be a good first step when implementing CBR.

## 8.6 Why use CBR?

When used in the appropriate settings, there are many advantages to be gained from utilizing CBR. Some of these advantages are mentioned in [19]:

- *CBR reduces the Knowledge Aquisition task.*  
Knowledge Aquisition usually includes the extraction of a model or a set of rules, which is necessary in model/rule-based systems. In CBR, the Knowledge Aquisition-task consists mainly of collecting, representing and storing cases.
- *Past mistakes can be avoided.*  
CBR systems should record failures as well as successes. Preferably, the

reason for these mistakes should be recorded as well. When using CBR, the chance of repeating past mistakes is reduced.

- *Graceful degradation of performance*  
Some model based systems run into real trouble when attempting to solve problems that are on the boundaries of their respective knowledge domains or scopes. CBR, on the other hand, can often have reasonably successful attempts at solving these kinds of problems.
- *CBR-applications are able to reason in domains that have not yet been fully understood, defined or modeled.*  
While insufficient knowledge may exist about a domain to build a satisfactory model or derive rules from it, a case-based reasoner may still function with only a few cases from the given domain. The underlying theory does not need to be quantified.
- *CBR systems may be able to predict a proffered solution's probability of success*  
If information about previous solutions' success or failure rates are stored, this can be used in future cases to predict the chance of success.
- *CBR systems learn over time*  
As more and more cases accumulate in the case base over time, a CBR system will be reasoning within a wider variety of situations, with an increasingly higher degree of refinement and success.
- *Reasoning with incomplete or imprecise data.*  
Even though a new problem doesn't correspond completely to previous cases, we can still reuse parts of old cases where there is a certain degree of similarity.
- *Providing explanation and justification*  
Previous cases and their (successful) solutions can be used to explain and justify a proposed solution to the user. In most domains, it is important for the user to be assured that he/she can trust the solution proposed to him/her.

## 8.7 State of the Art

### 8.7.1 Existing CBR-applications

CBR has been implemented in a variety of ways.

## 8.8 How does CBR apply to Aker Solutions and KBeDesign?

The rationale/justification for including CBR in this report is primarily because of its similarities with KBE. You could say that CBR and KBE are two sides of the same coin. They are two different methods of solving what is essentially the same problem. This fits well into the domain of KBeDesign, who are all about reusing past solutions to recurring problems.

The observant reader will without a doubt have seen the similarities between case representations and K-Briefs. And herein lies an important point: K-Briefs can be used as cases in a CBR process. This CBR-process does not even have to be fully automated - as long as we are able to retrieve K-Briefs that are relevant to a situation that we are currently working with, a lot the work might already be done. After discussing a problem and comparing it to previous, similar cases, people can come up with a solution for the new problem more easily. This is not to say that a proper, mostly automated CBR application isn't desirable either.

While KBeDesign's AML-applications automate the engineering and design in many "low level", routine tasks, combining K-Briefs and CBR could boost effectiveness and efficiency in more high-level problem solving and decision making, for example during the start of a new project. These high-level decisions could be problems like which materials or engineering solutions to use in certain environments or when working with certain customers.

Assuming that K-Briefs become a success within Aker Solutions, then taking the next step and using them as a basis for cases in a CBR-system could be a worthwhile effort.

## 8.9 Case Representation Using K-Briefs

An important factor in a CBR-system is case representation. The first thing we need to decide is *what* a case is. In general, cases can be people, objects, situation, diagnoses, designs, plans or legal rulings. In Aker Solutions, a case should correspond to a K-Brief.

While K-Briefs in Aker Solutions often are about a specific part or assembly, they can also represent other kinds of problems or situations. We need to figure out some key criteria that characterize a specific case/K-Brief (information that identifies the specific case). During one of my discussions with Geir Iversen, my contact person at Aker Solutions, we identified some key data that help identify a K-Brief. These data are:

- Project Number
- Regulations (NORSOK, AISC etc.)
- Customer ID
- People Involved: authors and contributors of a K-Brief and their roles (project leader, engineer, links to their profiles in Knowledge Arena (internal social network in Aker Solutions) and their competencies
- Key competencies that were used in the authoring of the K-Brief
- PEM (Project Execution Model) phase: which phase in the PEM the project is in (concept, FEED or detailing, for example)

In addition to this "identity"-information, we also need some information that describes the actual situation related to the case. If we were to only use the data in the list above when searching for similar cases, we would only find cases that are related to the same project, customer, people etc., without the cases actually being related to the same type of situation. Each case needs data that focuses on the *problem*, not metadata. If we are using K-Briefs as cases, this information will likely have to be textual information, where we will have to use full-text search to compare the similarity between cases. This textual information will be what separates cases that are connected to the same project, customer etc. (which is also represented as textual information in K-Briefs).

So, the initial idea was to represent a case as a K-Brief, which is essentially a rich text document, which in the simplest possible implementation corresponds



to a single text field (in addition to metadata related to the document). We determine similarity between cases by determining similarities between these text fields. Of course, we are not limited to representing the case description as a single text field. We can categorize this information as we please. The only downside is that this adds complexity to the case representation, and thereby makes the CBR-system slightly more complicated.



## Chapter 9

# Related Work



# Chapter 10

## Prototypes

### 10.1 Interactive Document UI Prototype

Based on my work with the Interactive Document, i have made an application that demonstrates how an Interactive Document could look like.

### 10.2 K-Brief Retriever

I have developed a simple application that searches through an set of K-Briefs and returns a set of relevant K-Briefs if the search is successful.

The goal of this application is to provide a baseline for further research regarding CBR. For my thesis, I have focused on the *retrieve* phase of the CBR-cycle. Due to limitations in both scope and time, I have not tried to implement the remaining phases. The idea is to leave the remaining phases for future research by PhD-students at NTNU.

The application is coded in C#, and uses three key open-source technologies to work: Lucene, Tika and IKVM.

### 10.2.1 Lucene.Net

Lucene is a search engine developed by the Apache foundation that does full-text indexing and searching. The user supplies some search terms, and Lucene returns a ranked set of documents that match the given terms. Lucene was originally developed in java, but a .NET-version has also been made, called Lucene.Net.

Lucene is widely considered one of the faster search engines available. It works by indexing a set of documents, and then the index can be rapidly searched through.

I chose Apache Lucene primarily because I was asked to use it by Mozhgan Tavakolifard, one of my main contact persons at NTNU. She is currently doing research on CBR and recommender systems, and was interested in seeing if Lucene could be used for case retrieval in these kinds of applications.

### 10.2.2 Tika

One critical challenge that I had to address is that Lucene can only search in simple text. In the real world, however, K-Briefs and other documents are created and stored using rich text formats like .doc and .pdf. Tika, which like Lucene is also developed by the Apache foundation, can solve this problem for us. Tika is a Java-library which provides us with tools for extracting text from files in rich document formats. This doesn't only include .doc and .pdf-files: many other formats are supported, including HTML, XML, all other Microsoft Office formats (including powerpoint), Rich Text Format, Java class files and archives, and even compressed formats like .zip. It doesn't stop at text formats either: Tika can also extract textual information (like metadata) from several image, audio and video formats [24].

### 10.2.3 IKVM

The observant reader might have observed an apparent contradiction so far in this section. I have written my application in C#, but I am using Tika, a source code library written in Java. This is where IKVM comes in.

IKVM is basically an implementation of Java for the Microsoft .NET framework [25]. It includes the following components:

- A Java Virtual Machine implemented in .NET
- A .NET implementation of the Java class libraries
- Tools that enable Java and .NET interoperability

Although Lucene has been fully translated to .NET, the .NET-implementation of Tika requires IKVM to function properly.

### 10.2.4 How the application works

This application is only intended as a proof-of-concept of Lucene's ability to effectively and efficiently search in large amounts of rich text documents. Because of this, the application is a simple Windows console application, with no fancy bells and whistles.

So how does it work? While the source code is available, I think it is worthwhile to include a simple step-by-step explanation of the application's mechanics here:

- When the application is started, Lucene is used to index all the files in a specified folder. For each file in the folder, Lucene attempts to add it to its index.
- Lucene structures data into Documents and Fields. Each file represents a Document, which contains Fields. For each file in the folder, a Document is created. Then, for each Document, several Fields are added. The most important Field is "text", which contains the text extracted from the file.
- To extract the text properly, we use Tika's text extraction functionality. The text is then stored in the Field called "text".
- Each file is added to the index, which is saved in a previously specified folder.
- Now we can use Lucene's search features to search the index.
- The application asks for our search term(s), which we supply, and then the file names of the most relevant files are shown.

- If we choose to, we can open the highest ranking document from the application.

We have now successfully searched for relevant (rich) documents in a folder of our choosing.

### 10.2.5 Alternative technologies for future research

Lucene is not the only technology that can perform the tasks that the K-Brief-recommender does. While Lucene works perfectly fine for a proof-of-concept console application, researching and developing a web application that performs the same tasks is probably going to be more practical and interesting, at least from a user perspective.

#### Solr

If we want to develop an online K-Brief recommender, Apache Solr seems like a promising piece of technology. Solr is a search server. It's a stand-alone Java application that uses Lucene to provide full-text indexing and searching through an XML/HTTP-interface. This means that it can be used from any platform or language. It can be embedded in Java applications, but this is not how it is primarily meant to be used. Solr is considered to be easier to use than raw Lucene, and provides features commonly used in search applications, like faceted search and hit highlighting. It also handles caching, replication, sharding, and has a web admin interface [26].

#### SolrNet

SolrNet is a library used to communicate with a Solr instance from a .Net-application. It provides an object-oriented interface to Solr's features. It also works as a query-Solr-mapper: query results are mapped to *PONOs* (Plain Old .Net Object), which simply explained is just a simple .Net-object, making the data from search results easily accessible. You can access the search results just like you would access the data in any other object [26].



### 10.2.6 Comparison Between Baseline and existing CBR systems

Now that we have a baseline, the K-Brief Recommender should be compared to other CBR-systems to see if it is actually any better than existing applications. Keep in mind that we can only compare applications with respect to the "retrieve"-stage of the CBR-cycle. The main advantage of the K-Brief Recommender is that it can work directly with rich documents, but we need to know that it retrieves relevant documents.

We basically have two ways of comparing retrieval results: we can use some automatic scoring system, that calculates a set of documents' "similarity score" based on some pre-defined criteria. The K-Brief Recommender will retrieve one set of relevant documents, and the other CBR systems we compare it to will probably retrieve a slightly different set of documents. The application that retrieves the highest scoring set of documents will be considered the most effective. It should be mentioned that both Lucene and existing CBR applications use some kind of similarity measure to retrieve relevant cases, so what we will really be comparing is Lucene's document scoring mechanism versus existing CBR systems' similarity measures.

Another way of scoring sets of documents is to use expert opinion. This can be used both as a replacement or a supplement to automatic scoring. If there is little difference in the document sets, for example, an expert could be used to determine which set is actually the most relevant.

If we compare the K-Brief Recommender with existing CBR-systems and the results are around equal or better, we have some exiting opportunities in front of us. That means we have foundation for a new CBR system ready, based entirely on open-source technology.

### 10.2.7 How does this relate to context-aware computing and/or CBR?

Now that we have an application that can search in rich documents, what can we use it for? How does it relate to the rest of what I have discussed earlier in this thesis?

The next step is to make an application that can intelligently define search

terms based on the current context. These search terms can then be used to search in a large archive of K-Briefs, to quickly find K-Briefs and other documents that are relevant to the current context. What I am describing here is basically the *retrieve*-stage in a CBR-cycle, where we are using context to define our current situation/case. As we saw in the section regarding CBR, having a database of past cases to aid us when making new decisions is without a doubt helpful. So whenever we find ourselves in a context that resembles one or more cases from the past, we can immediately use these past experiences to our advantage.

# Chapter 11

## Discussion

This thesis discusses a lot of different technologies. Each have their own uses and applications. The main point of this report lies with how we, can successfully combine these technologies/methodologies and achieve an even greater, synergetic effect.

### 11.1 KBE and Context-Aware Computing

Knowledge-Based Engineering greatly increases efficiency of routine engineering tasks. Utilizing the principles of context-aware computing and semantic reasoning allows us to extract the most relevant data from a user's interaction with an application, and tailor the interaction depending on the user's context. One of the great strengths of context-aware computing is the ability to logically infer what the user really needs while he/she is attempting to perform a given task. The application can then do its best to cater to these user needs. The immediate user needs in an engineering or CAD/modeling context is to have relevant information and documentation available. This is especially the case when the user is performing some kind of KBE-powered design or modeling, where CAD models are automatically generated from a limited set of user input. This takes us back to the original problem or exposition: currently, engineers may experience difficulties when trying to verify that automatically generated designs are valid. This is usually because of lacking information, or the information may be

hard to find. If we utilize context-aware computing correctly, we can design our software tools and applications such that the application registers what type of model (or general problem) that the engineer is working on, and has the relevant information available automatically if the need for this information should arise.

## 11.2 Knowledge Visualization and Representation

Once we know how to sort out what information the user really needs (via context-awareness), there is the question of how this information should be presented. Currently, we have two available formats: interactive documents and K-Briefs. Although they do seem to complement each other, K-Briefs will probably have to be prioritized, due to their ease of implementation and immediate usefulness. Interactive documents should not be completely disregarded yet, but it will take time and resources to properly implement them. As I have mentioned earlier, I would recommend a "middle way"-approach, where we either make the interactive document templates more general and adaptable to different kinds of models (instead of just nodes), or implement a few of the elements found in interactive documents into the existing KBE-applications, in addition to using K-Briefs. In the immediate future, however, K-Briefs will probably be the only realistic alternative for knowledge representation. This has its pros and cons. While K-briefs are a far easier tool to work with for most people, due to the intuitive information structure (and the fact that it's basically a paper sheet), it only shows *general* information about model *classes*. If we want to inspect the data contained in a specific model *instance*, K-Briefs aren't much help. The genericity of K-Briefs can also be one of its strengths, however. With K-Briefs we are able to represent situations and problems that are too general or high-level to be represented in an interactive document, like high-level decisions that are made in the start-up stage of a project. Examples of such decisions can be what type of node designs to rely on when working with a particular customer or which conditions to be aware of when working in a particular region or climate, just to name a couple of examples.

## 11.3 CBR

The next big topic we need to discuss is Case-Based Reasoning (CBR). CBR fits neatly into everything we have so far talked about because it is relatively easy to use K-Briefs as cases in a case base. If we define a proper case representation such that it is easy to structure and record cases, we can create a large case base consisting of previous situations and problems. Whether or not Aker Solutions has need for a CBR-system can be debated, but I think it seems like a very useful technology/methodology which has many potential applications in any large, knowledge-intensive company where similar situations or problems recur over time. CBR could be a great tool to have on its own, even without combining it with the other mentioned technologies. However, there exist some interesting possibilities if we should try to do this: it certainly seems like an exiting feature to have a context-aware application that that uses context to automatically suggest relevant K-Briefs from a case base.

An argument that can be made against implementing CBR is that it might be redundant to some degree if we already have a context-aware system that understands the user's context and recommends K-Briefs that correspond to this context. We can make a counter-argument to this, though: if we have a context-aware system that recommends relevant K-Briefs, we have already implemented the "retrieve"-phase of the CBR cycle. It is not necessarily such a large task to implement the rest of the CBR-cycle. And consider another scenario, where we are starting to solve a completely new problem. We structure and formulate our new problem such that it fits into our case-representation schema. Would it not be great to automatically discover that there are several cases that show some degree of similarity, and contain partial solutions that we can reuse? Depending on the complexity of the CBR-system, the system might even suggest a solution to our new problem by utilizing these past cases.

Throughout this report, I have made the assumption that if CBR is implemented, K-Briefs should be used as cases. In retrospect, this might seem like a slightly forced effort, just to make CBR fit into the palette of technologies that I have discussed. We should consider the option to implement CBR as a completely new system, where the case representation format does not depend on any existing formats like K-Briefs. This is not to say that we cannot use K-Briefs or A3s as an inspiration for how the knowledge contained in a case could be presented. We should also keep the LEAP-architecture in mind. Utilizing context-aware applications, ontologies and semantic reasoning in combination

with CBR still sounds exiting. It's just that, for CBR, we might use a better case representation than just a K-Brief. We could define a more detailed case representation schema, and relate each case to a K-Brief, for example. We just have to keep in mind the added complexity of this approach.

## 11.4 K-Brief Recommender

Something interesting that I have been able to test out is the ability to search directly in K-Briefs (represented in rich document formats such as pdf, doc or other formats), without having to go the route of creating relational database that keeps track of K-Briefs and the key content contained in them. This way, the search is directly related to the knowledge stored in the K-Briefs. We are therefore not dependent on storing the K-Briefs in an intermediate location, where there is some possibility of K-Briefs/cases being stored with incorrect or insufficient metadata because of human error. So, as long as we have a few important key words, and we know that all K-Briefs are structured in a similar way, containing some key identifiers like authors, project number, model number (in KBE cases) and a description of the topic, we can easily find relevant K-Briefs/cases. If we *want* to use relational databases for K-Briefs, this is of course possible. One of the technologies that I tested (Tika) could read data from K-Briefs and store it in an RDMS-system if that was our wish, but with the current volume of K-Briefs, using Lucene to index and search these K-Briefs yield both fast and relevant results.

## Chapter 12

# Results and Conclusion

While working on my thesis, I have explored several technological domains, and attempted to see if they could be used in an engineering context. In this chapter, I will list what I perceive as the main results of my research:

- Context-aware computing is an exciting field of research, with many possible applications. Most of the research in this field is done with mobile devices and pervasive computing environments in mind, but we can also apply some of the principles of this field into a business or engineering environment. Using context-aware applications and semantic reasoning for engineering tasks allows the application to understand what the user is trying to accomplish, and the application can help make the user's task easier by providing information or help that is relevant to the task.
- I have discussed two different knowledge representation formats: Interactive Documents and K-Briefs. While both of the formats have their advantages, the Interactive Document needs both further adaptations and the establishment of additional infrastructure to be viable. K-Briefs are already being implemented, and they show potential to become great information carriers within large, knowledge-intensive companies like Aker Solutions.
- I have studied Case Based Reasoning, which seems like a very useful methodology for large, knowledge-intensive companies. The problem-solving capabilities of CBR also go beyond simple engineering or design

tasks. While KBE is used to successfully automate routine design operations by using simple rules, CBR can be applied to high-level problem solving and decision making.

- I have successfully made an application, the K-Brief Recommender, that is able to search rich documents, using only open-source technology. This can be used as a baseline for a potential CBR-system (the application implements the *retrieve* phase of the CBR cycle). The idea behind this is that we could use K-Briefs as cases in a CBR-application, and K-Briefs are usually created using rich document formats. This could also potentially be combined with context-aware features, where the application automatically recommends K-Briefs that seem relevant to the user's current context.



# Bibliography

- [1] B. Schilit, N. Adams, and R. Want: *Context-aware Computing applications*
- [2] Reto Krummenacher, Thomas Strang: *Ontology-Based Context Modeling*
- [3] Andreas Zimmermann, Andreas Lorenz, and Reinhard Oppermann: *An Operational Definition of Context*
- [4] Matthias Baldauf, Schahram Dustdar and Florian Rosenberg: *A Survey on Context-Aware Systems*
- [5] Harry Chen, Tim Finin and Anupam Joshi: *An Ontology for Context-Aware Pervasive Computing Environments*
- [6] CoDAMoS Web Page:  
<https://distrinet.cs.kuleuven.be/projects/CoDAMoS/>
- [7] Tao Gu, Hung Keng Pung, Da Quing Zhang: *A Middleware for building Context-Aware Mobile Services*
- [8] Geir Iversen, Gabriela Rutkowska, Kjetil Kristensen: *LinkedDesign D9.1: Definition of Concepts and Requirements*
- [9] Osmund Chandra Maheswaran: *Prototyping Data-, Information- and Knowledge Visualization*
- [10] Geir Iversen, Gabriela Rutkowska, Oluf Tønning, Simone Parotta, Kjetil Kristensen, Mozghan Tavakolifard, Patrick Klein: *LinkedDesign D9.2: Prototypical knowledge integration between CAx and KBE systems*
- [11] Wikipedia article on Ontology:  
<http://en.wikipedia.org/wiki/Ontology>

- [12] Wikipedia article on ontologies in information sciences// [http://en.wikipedia.org/wiki/Ontology\\_\(information\\_science\)](http://en.wikipedia.org/wiki/Ontology_(information_science))
- [13] <http://www.fao.org/countryprofiles/geoinfo/en/>
- [14] Wikipedia article on the Web Ontology Language (OWL): [http://en.wikipedia.org/wiki/Web\\_Ontology\\_Language](http://en.wikipedia.org/wiki/Web_Ontology_Language)
- [15] Xiao Hang Wang, Tao Gu, Da Quing Zhang, Hung Keng Pung:  
*Ontology Based Context Reasoning Using OWL*
- [16] Michel Triana: *Ontology...what?*  
<http://michel triana.com/2012/01/20/ontology-what/>
- [17] Kjetil Kristensen: *Collaboration*
- [18] LinkedDesign Deliverable 3.2: The LinkedDesign Semantic Model
- [19] Julie Main, Tharam Dillon and Simon Shiu: *A Tutorial on Case-Based Reasoning*
- [20] Christiane Gresse von Wangenheim: *Case-Based Reasoning - A Short Introduction*
- [21] Case-Based Reasoning wiki:  
[http://cbrwiki.fdi.ucm.es/mediawiki/index.php/Main\\_Page](http://cbrwiki.fdi.ucm.es/mediawiki/index.php/Main_Page)
- [22] Agnar Aamodt and Enric Plaza: *Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches*
- [23] Wikipedia page on Case-Based Reasoning:  
[http://en.wikipedia.org/wiki/Case-based\\_reasoning](http://en.wikipedia.org/wiki/Case-based_reasoning)
- [24] Apache Tika website  
<https://tika.apache.org/1.3/formats.html>
- [25] IKVM website  
<http://www.ikvm.net>
- [26] Bug squash blog-entry by Mauricio Scheffer:  
<http://bugsquash.blogspot.no/2009/10/untangling-mess-solr-solrnet-nhibernat.html>

## Appendix A

# The Interactive Document

In this section I will show an example of how an Interactive Document could be implemented. The illustrations provided here are simple GUI mock-ups. I will use a constructional node (or "joint") as a case. Simply put, a node is where beams, columns and braces in a construction meet, and forces are transferred.

## A.1 Tab 1 - General

Interactive Document - Node #123

General Calculations Node Class Source Code Members Dimensions

picture of node

Attributes

Name: Node #123  
Standard: NORSOK  
Node Type: Beam to Tubular Connection  
Connected Beams: 2

Notes

Note	Date	Name
This node might be modified later in the project	15.03.2015	Mr Boss

Add Note
Delete Note

Modifications

Version #: 3

Note	Date	Name
Node type changed	16.03.2015	Mr Engineer
Construction geometry changed slightly	15.03.2015	Mr Engineer

View Previous Versions

OK
Cancel

Figure A.1: Caption

The general tab includes some general information about the model. The name of the node is provided, together with the standard used to design it, the node type, as well as the configuration. The general tab contains space for adding notes so that the user can inform other engineers about choices that have been made, about important decisions concerning the design or provide warnings. In addition, links to previous versions of the interactive document are included, so that the user can follow the model's development. A list of changes made is given in a table, together with the date, the name of the engineer and a possibility to "rewind" to a previous version. By clicking on "Edit Node", the user can close the document and make changes to the node [9][10].

## A.2 Tab 2 - Calculations

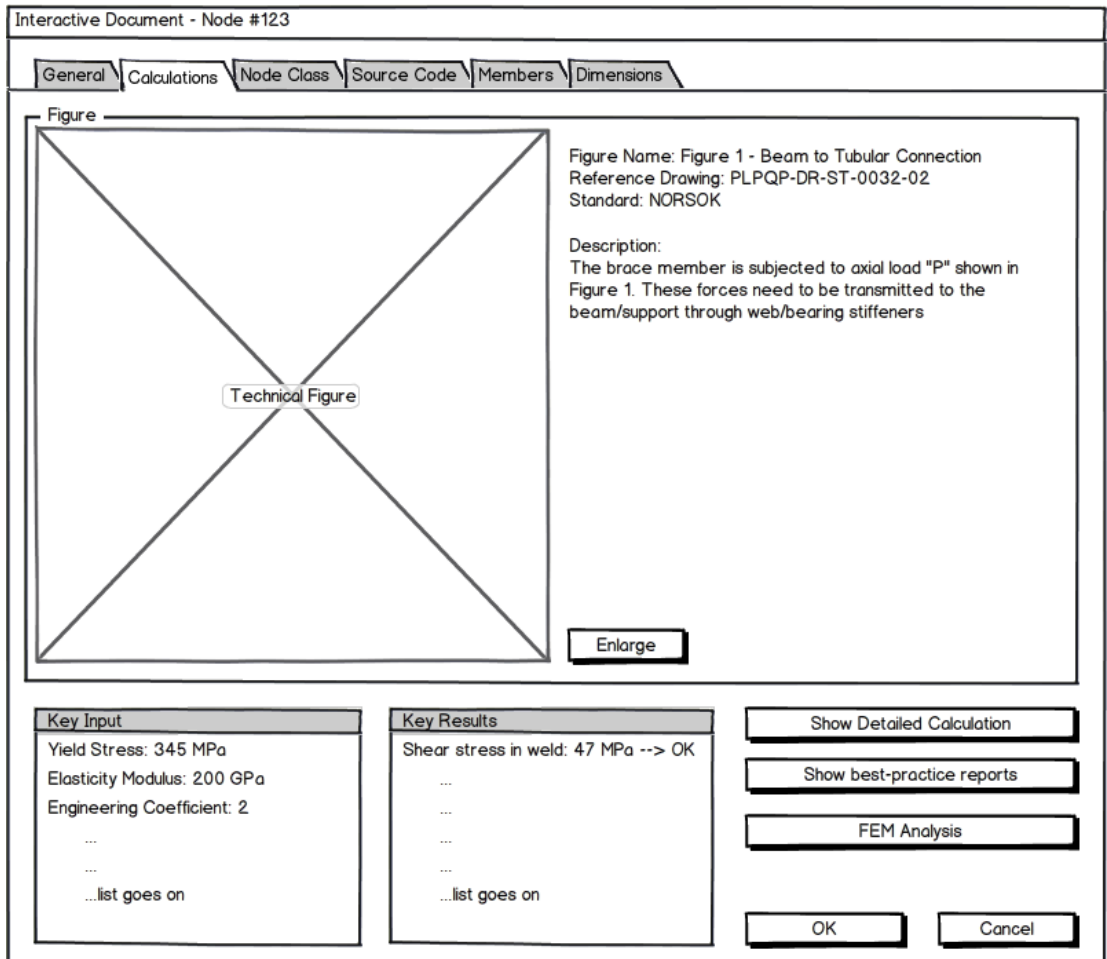


Figure A.2: The "Dimensions" Tab

In the "calculations" tab, the user should be able to find relevant calculations for the node. The foundations for these calculations are presently Mathcad

sheets where key values and geometry from the node is used as input, and an acceptable result is achieved. These inputs and results are shown at the bottom left of the screen. The Mathcad sheet reflects how a structural engineer would do calculations according to a given standard (for example NORSOK). If the user wishes to see how the calculations are done, he/she can press the “Show Detailed Calculations” button to view the entire Mathcad sheet [9].

If an acceptable result is not achieved, the user should be notified, and informed about what went wrong. For example, maybe one or more of the checks in the Mathcad sheet has failed, and the user has to manually modify the geometry of the node and/or beams to make everything work [9].

There are two other features here not yet mentioned. The first is “Show best-practice reports”. This feature could link the user to a set of documents relevant for the entire project, and open the document relevant for this particular type of node (more about this later) [9].

Next is “FEM Analysis”. This feature could create, run and show a FEM-analysis of the node. If a FEM-analysis already has been conducted for the node, the results from this analysis can be fetched and shown to the user. Here it is also important to keep track of whether the FEM-analysis is up to date or not. If a previously conducted FEM-analysis does not exist, the user will have to make a new one. This will have to be done via integrating the Interactive Document with some kind of FEM-analysis software. This is not entirely straight-forward, though. One way of solving this could be by using a “wizard”-approach, where the user is guided through the necessary steps. First you need to properly export the model of the node to a program with FEM-analysis capabilities, like Genie, and do the analysis using this program. Results can of course be studied in Genie, but it should also be possible to save it and review it later without having to open Genie again. To accomplish this, the results of the analysis will have to be saved in a format that can be viewed in the interactive document [9].

Below are a couple of screenshots from the Mathcad sheet for a certain type of node:

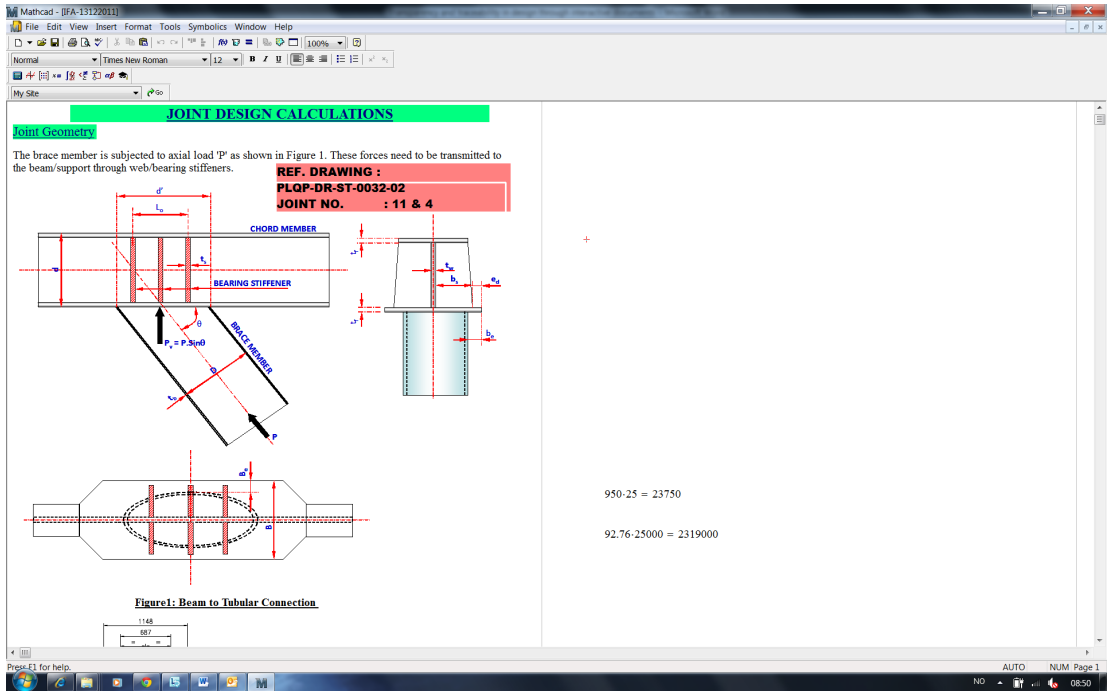


Figure A.3: MathCAD document, technical figure [9]



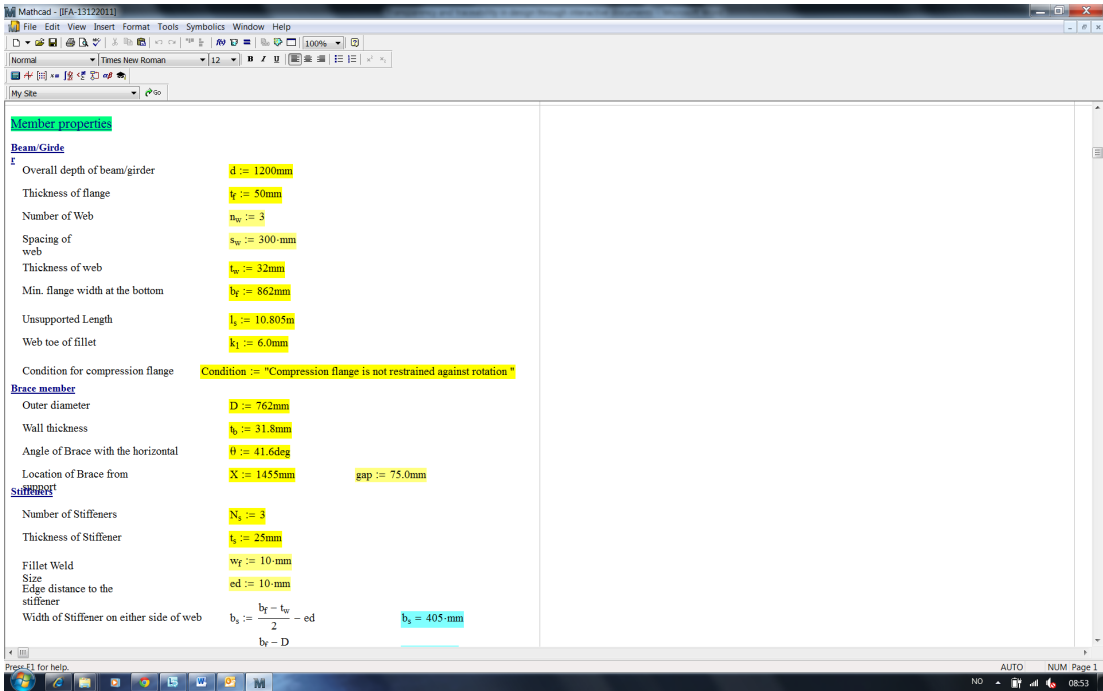


Figure A.4: Input parameters for the MathCAD sheet [9]

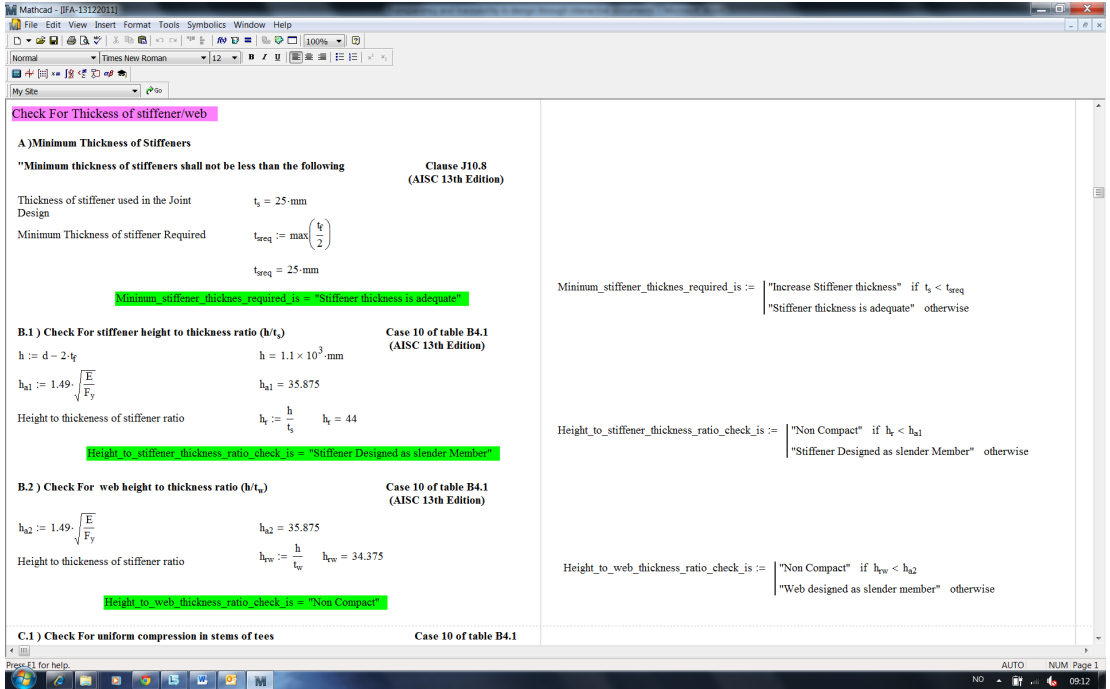


Figure A.5: Some of the calculations performed [9]

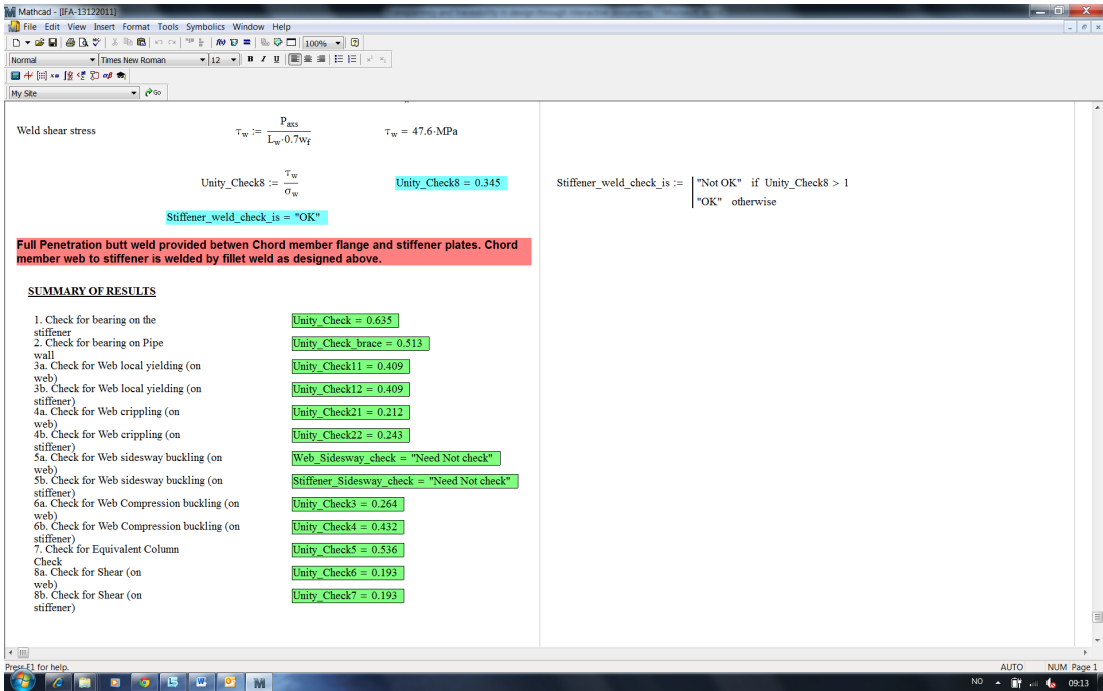


Figure A.6: Summary of key results for the MathCAD calculations [9]

Note: KbeJOINT

### A.3 Tab 3 - Node Class

Interactive Document - Node #123

General | Calculations | **Node Class** | Source Code | Members | Dimensions

Node Class: Beam to Tubular Connection 123

**Node Family**

- Fabricated
- Welded
- Special

**Node Type**

- Box
- Gusset Plate
- Open Beams
- Sketcher

**Configuration**

**Beams**

- IG
- OD
- Box
- HE
- RHS

**Columns**

- IG
- OD
- Box
- HE
- RHS

**Braces**

- IG
- OD
- Box
- HE
- RHS
- None

Image explaining what beams, columns and braces are,

More Info

**Node Constraints:**

Standard #456  
Customer Request #789

Show Constraint Info

OK Cancel

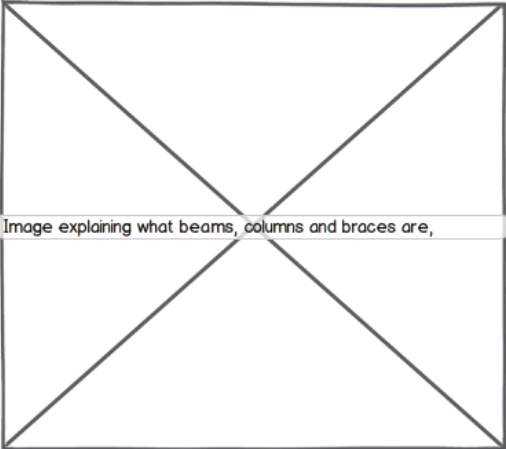


Figure A.7: The "Node Class" Tab [9]

The intention for this screen was for it to display some quick facts about the node class for the selected node, and how it is put together via a certain configuration

(more on this later in the text). Constraints governing available node classes are also shown. The point of all this is to have a quick and easily accessible way of displaying why the node is of one class and not the other. What is shown here are the qualitative properties of the node. The “More Info” and “Show Constraint Info” buttons link to files in the documentation collection mentioned earlier. Using these buttons the user can read more about node types, standards used for the project, or other constraints that might be relevant [9].

## A.4 Tab 4 - Source Code

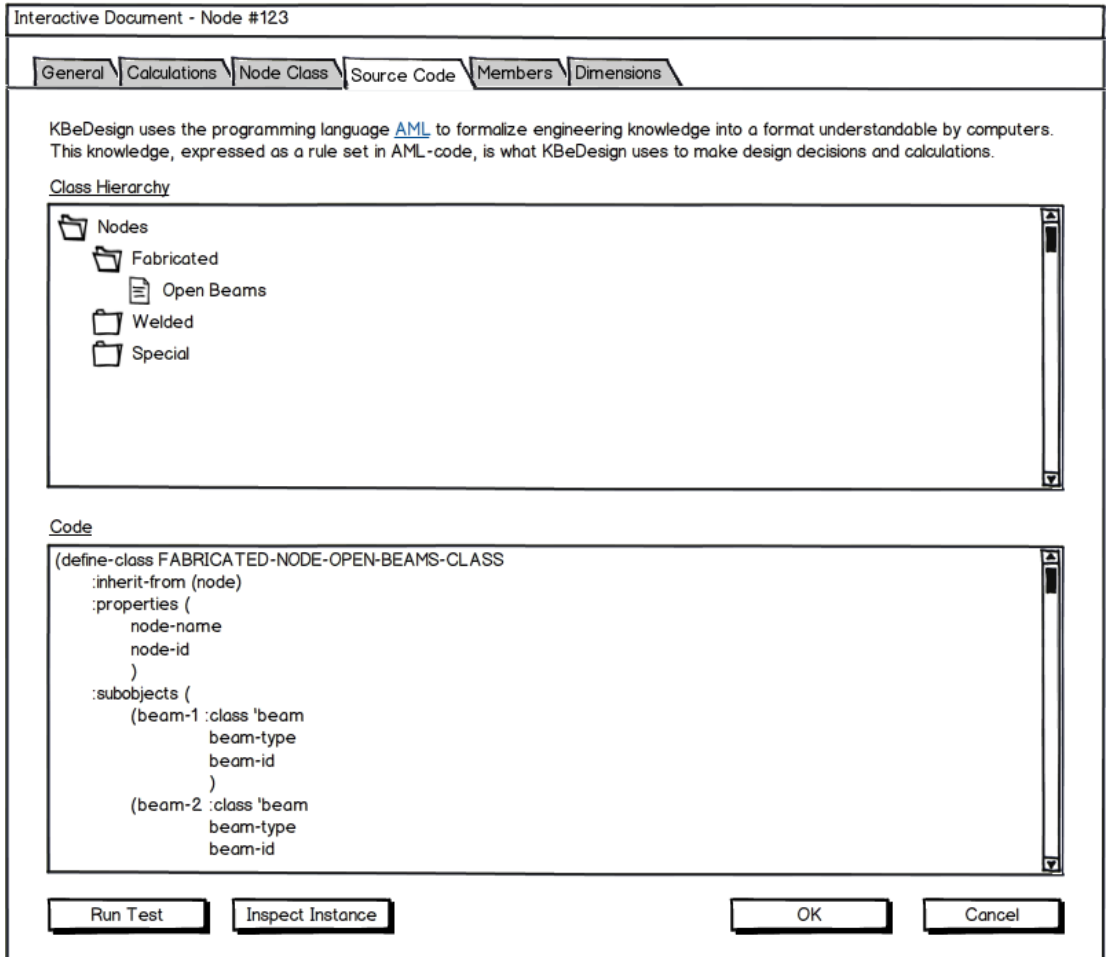


Figure A.8: The "Source Code" Tab [9]

The purpose of this screen is to show how the node is represented in AML. In the top window, the user can see where the class belongs in the class hierarchy.

The user is free to navigate this hierarchy, and view the code for any class. This allows the user to see the code in context, and understand any relevant superclasses or subclasses. The bottom window shows the code for the class selected. Note that what is shown in this window is the code for this particular class, not this particular instance. Viewing the code for a particular instance of a node is already possible in various KBeDesign applications, via the “inspect” feature. However, I didn’t see any problems with also including this feature here, so it could be available via the “Inspect Instance”-button, found at the bottom left of the screen. This will point the user to the code for the particular node he/she selected when he/she opened the interactive document. The last thing that needs to be mentioned here is the “Run Test”-button. This allows the user to run a test using AUnit to verify that the code presented to him/her works as intended [9].

## A.5 Tab 5 - Members

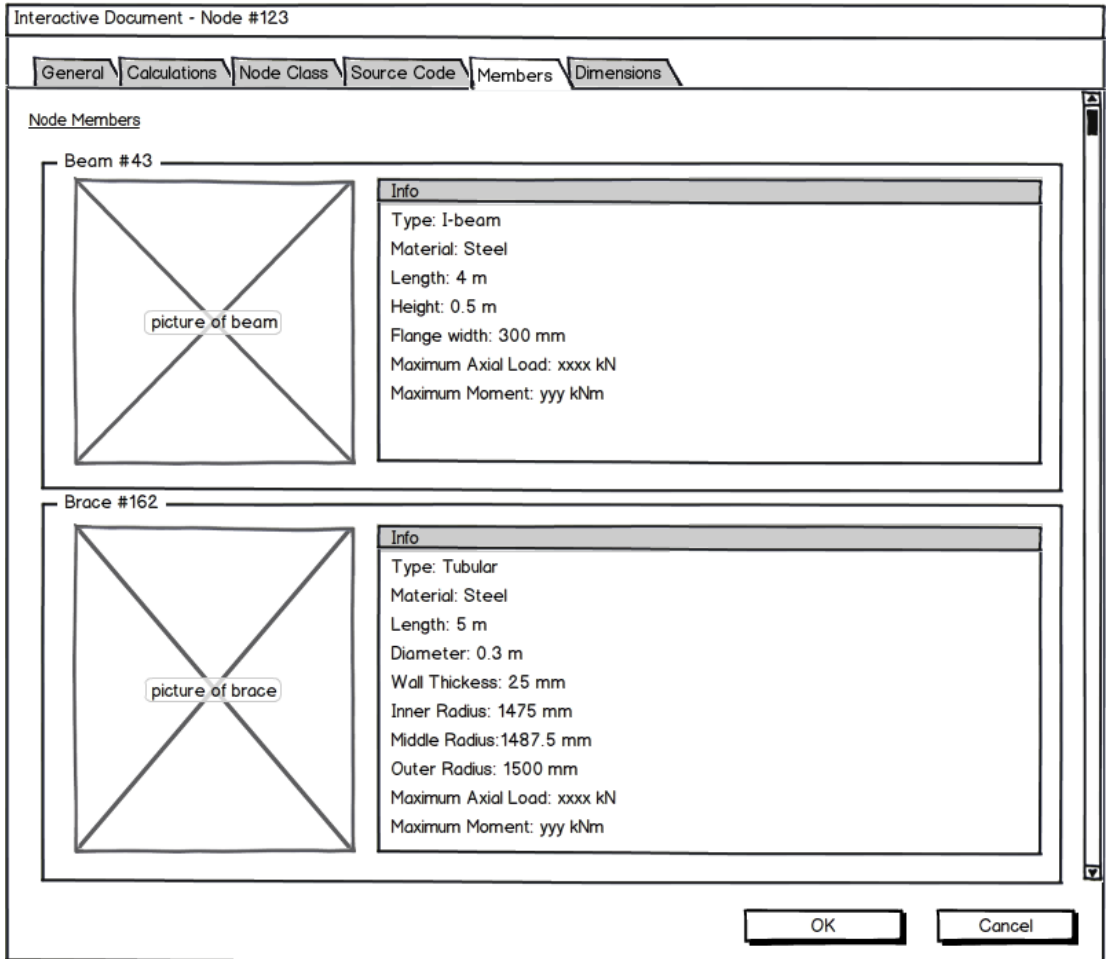


Figure A.9: The "Members" Tab [9]

This screen shows all beams, columns and braces (in other words, all the members) connected to the node. For each member, relevant information and speci-



fications are shown. Information about each beam could give the user an idea of how the node will behave as a whole. It could be useful for navigating through a construction and being able to see how everything is connected [9].

### A.6 Tab 6 - Dimensions

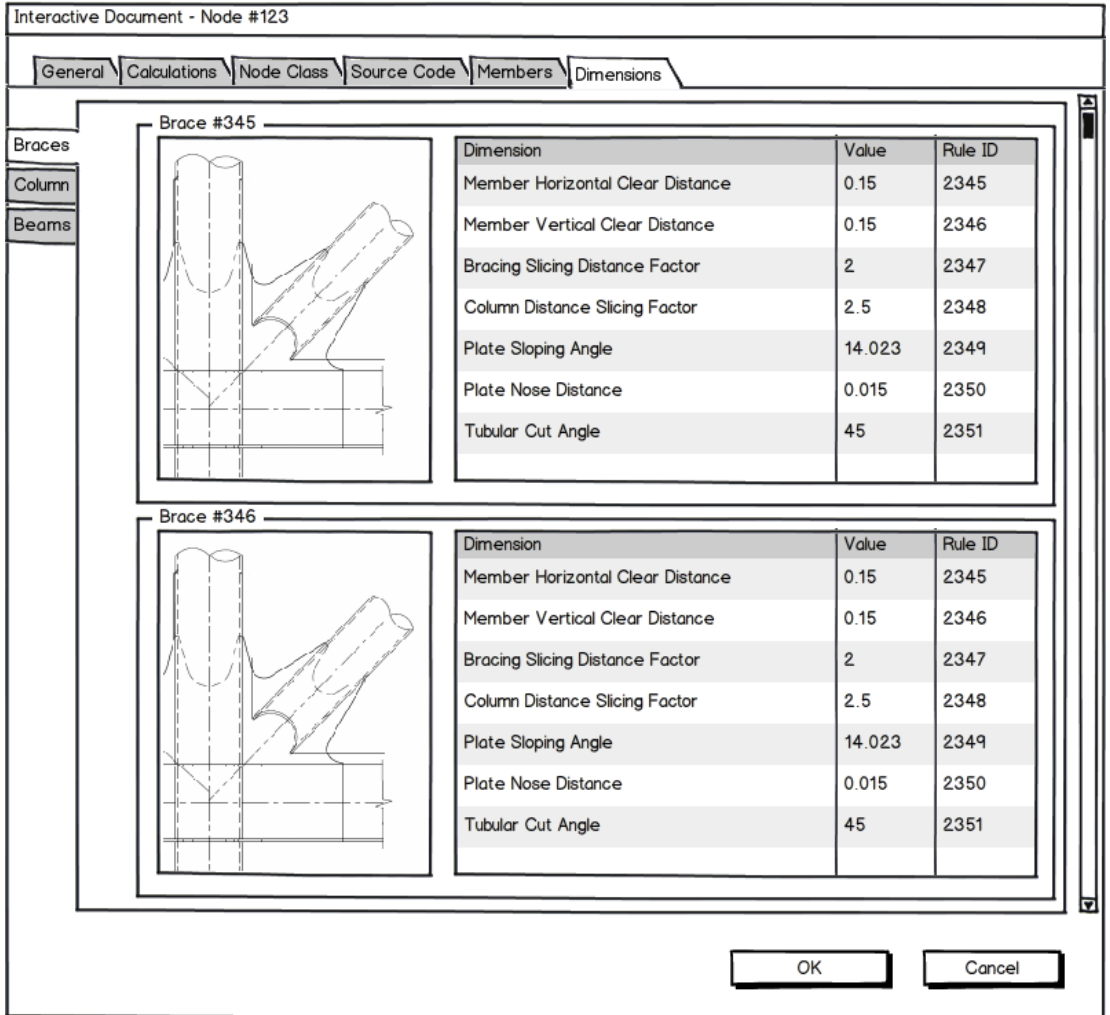


Figure A.10: The "Dimensions" Tab [9]

In the Dimensions tab, the user is able to inspect geometrical aspects of the node and get an explanation for why each dimensional property has a certain value. Selecting a dimension from one of the tables will highlight the dimension in the table's corresponding figure.

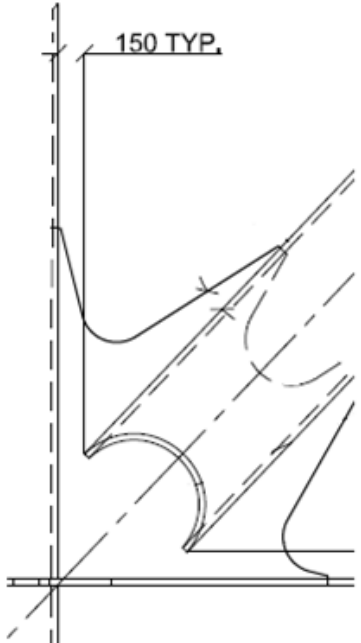
The Dimensions tab will have slightly different layouts depending on the class of the node we are looking at. This is because the number of beams, braces and columns may vary between various node classes, and each of these members may have different sections. Therefore, the kind of information we wish to see in the Dimensions tab will vary between node classes.

Notice that the Dimensions-tab has three sub-tabs. The dimensions of the node will be divided among these sub-tabs.

The Rule ID's are hyperlinks. When we click them, we are shown a description of the given rule. For example, if we click the Rule ID for Member Horizontal Clear Distance, the following window appears:

Interactive Document – Rule #34: Horizontal Clearing Distance

Horizontal Clearing Distance



The diagram illustrates the horizontal and vertical clearing distances for a tubular brace. A vertical dashed line represents the wall. A horizontal dimension line indicates a distance of 150 TYP. from the wall to the centerline of the brace. The brace is shown in a 3D perspective view, with its horizontal and vertical projections clearly visible. The horizontal projection shows the brace extending from the wall, and the vertical projection shows the brace's height and clearance from the floor.

The horizontal and vertical clearing distance for a tubular brace is typically 150 mm. KbeDesign uses this as the default value.

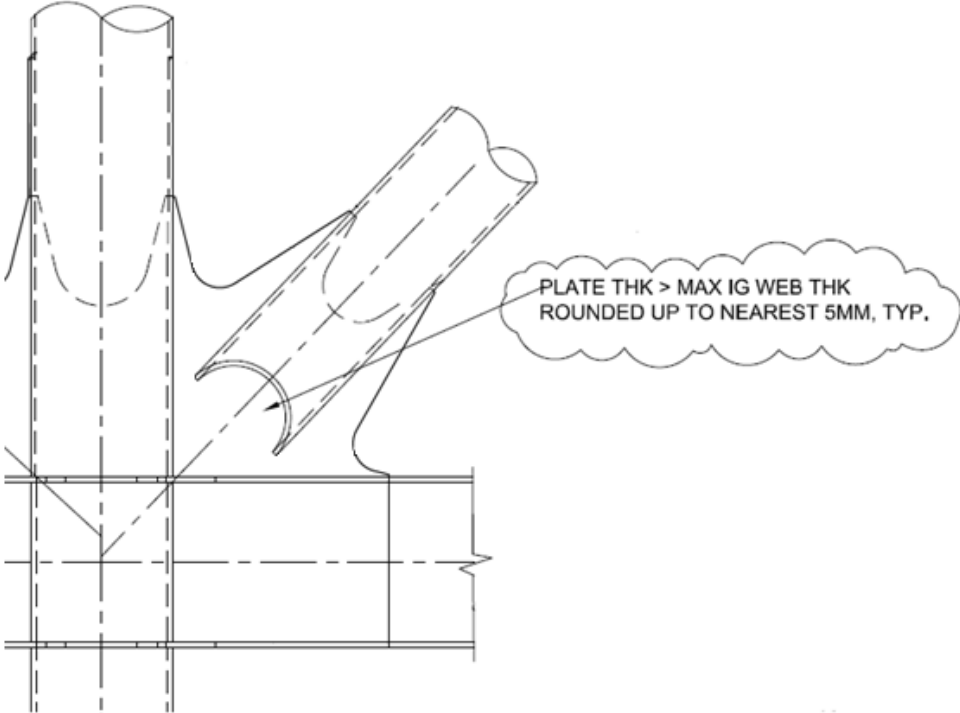
Source Document: [Document #456](#)

OK

Figure A.11: Rule Explanation of "Horizontal Clear Distance" [9]

Similarly, if we click the rule for “Gusset Plate Thickness”, the following window appears:

Interactive Document – Rule #43: Gusset Plate Thickness



The gusset plate thickness has to be greater than the thickness of the thickest incoming beam, and rounded up to the nearest 5mm.

Source Document: [Document #456](#)

OK

Figure A.12: Rule explanation of "Gusset Plate Thickness" [9]

These windows could possibly be incorporated into the Project Documentation mentioned in the next section.

In both of the example screens describing geometric rules used by KBeDesign, there is a link to a “source document”. This is the documentation where these rules are found. The example screens shown here are simplified figures showing only one rule each, so the user doesn’t have to be confused with a huge figure containing loads of information.

## A.7 Project Documentation

I think it would be a nice feature, in any KBeDesign application, to be able to quickly and easily view any relevant document that might answer questions or clarify any issues that the user might be wondering about. Exploring this kind of documentation could also have a preventive or precautionary effect for the user, so that he/she will not make design decisions that are not proper for a given project.

The document collection could look like this:

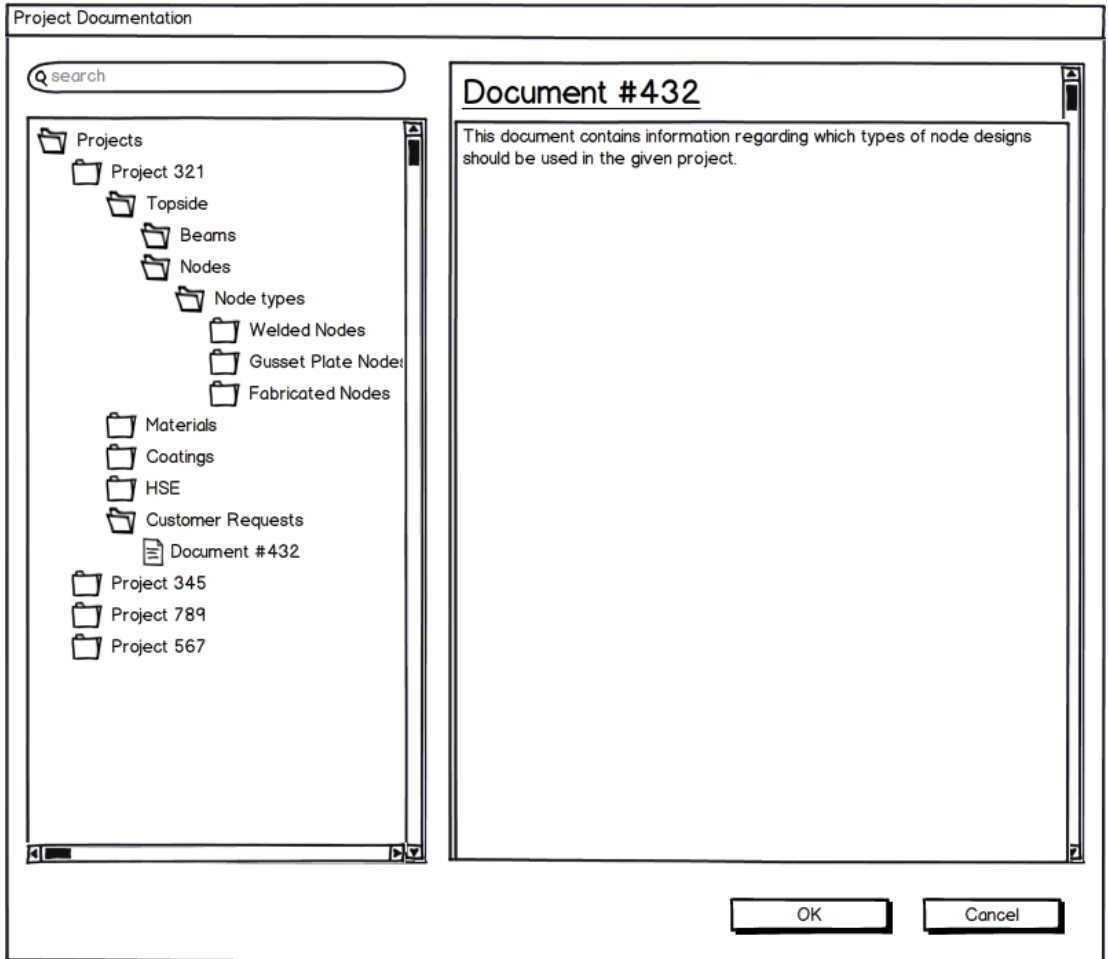


Figure A.13: The "Project Documentation" Tab [9]

It very much resembles any "help"-window, for better or worse. The hierarchical layout helps separating between projects and important topics within each project.



## Appendix B

# Example K-Briefs

Revision date: 29.11.2012  
 Author: Rutkowska, Gabriela  
 Engineer, KBeDesign  
 gabriela.rutkowska@akersolutions.com

# Nodes



## Introduction

This K-brief gives an overview over node types, rules for choosing a node type, the process of creating node models in KBE, as well as general node parameters in KBE.

## Short theory

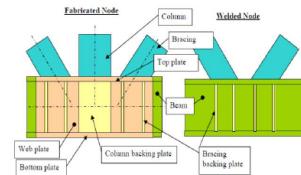
Nodes are connection points between beams and braces in a topside structure.

## Node types

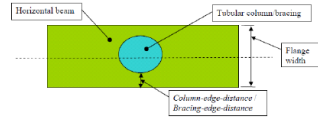
**Welded Node** does not contain any additional plates, except for optionally backing plates. Sections in a welded node are trimmed to each other.

**Fabricated Node** contains flange-, web- and backing plates.

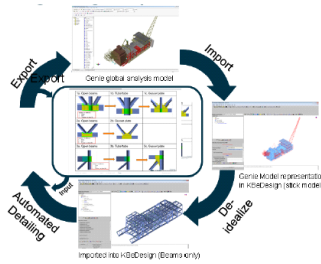
- Open beam
- Box Node
- Gusset Plate



Rules for choosing the node type: The Fabricated Node will only be created if the vertical column or bracing diameter is greater than the flange width of the horizontal beam, included a edge distance value, see figure. A Welded Node will be created when the rule above is not satisfied. Any changes to incoming sections in the node will re-evaluate the rule for creation of Fabricated node, and create a fabricated or welded node.



## Creating node models in KBE



If the purpose is to modify the model design based on a structural analysis, a topside model is imported from an analysis program. The model can be modified manually by the engineers in the analysis program or in PDM. However, involving KBeDesign leads to a faster redesign process as all the relevant rules are implemented and thus the modification process is automated.

As the input is loaded, AMI generates a model topology in form of a "stick-model". KBeDesign continues working on this model to make Structural Detailed Nodes as well as bulkheads and decks. The user can himself define nodes and their properties, e.g. node type and position or choose to use default values. The node's final design is thus decided by the KBE user.

For more information on node detailing in KBE, [click here](#).

Node types that are available in KBE:  
 Best practice for both welded or fabricated nodes is available in KBE. Supported types for fabricated node are open beam, gusset plate and beam box. There is also a possibility to build a node model from scratch in KBE.  
 Bolted nodes are not included in the scope.

## General node parameters in KBE

Input Property name	Description	Type	Type of input:	Default value	Constraints
Horizontal-bracing-gap	Horizontal gap between bracing and column	input	real	50	
Vertical-bracing-gap	Vertical gap between bracing and beam	input	real	100	
Bracing-gap	Select bracing gap type	input	string	Horizontal	Horizontal, Vertical
Add-backing-plates	Allow the user to turn on/off creation of backing plates	input	boolean	true	True, false

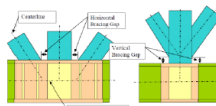


Figure 4-1: Horizontal and vertical bracing gap

## FAQ

Why are only these node types available in KBE?  
 -> Best practice/ engineering has shown that these are the most common node types used in projects.

Why cannot I create fabricated nodes in KBE?  
 -> Check if the incoming beams allow to create fabr. nodes.

Why did my node disappear after I changed the beam in KBE?  
 ->The node's design is dependant on the beam, it's orientation and profile. Kanskje heller si noden forsvinner. Look at rule-logs. Redraw the nodes.

## Further Reading

Technical knowledge K-briefs:  
 -> Box Nodes  
 -> Gusset plate  
 -> Open beams

Process knowledge K-briefs:  
 -> [Detailing in KBE](#)

## Relevant Experts

Name	Role
Road Delp	Super User Topside

Figure B.1: K-Brief, Node [10]

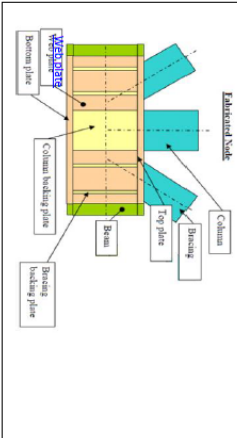
# Fabricated Nodes

### Introduction

This K-brief gives an overview over fabricated node ,procedures for choosing a fabricated node type, as well as general parameters for fabricated nodes in KBE

### Short theory

Fabricated Node is a **node** made up of plates and those plates are calculated based on rules in KBE. It contains flange and web plates in addition to backing plates. The Fabricated Node will only be created if the vertical column or bracing diameter is greater than the flange width of the horizontal beam, including a edge distance value.



Box Nodes

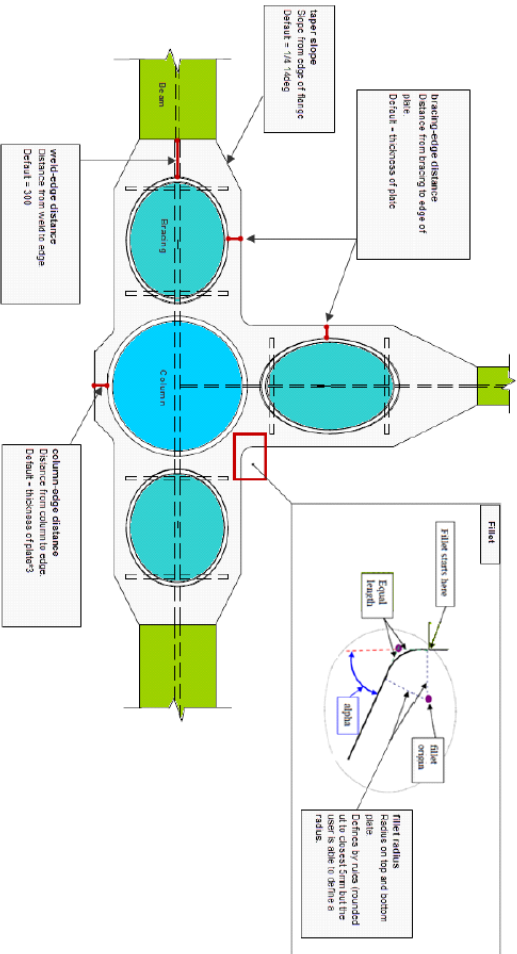


Gussset plate



Open beams

### General Fabricated Node parameters and constraints



### Procedure of choosing nodes in Edward Grieg project.

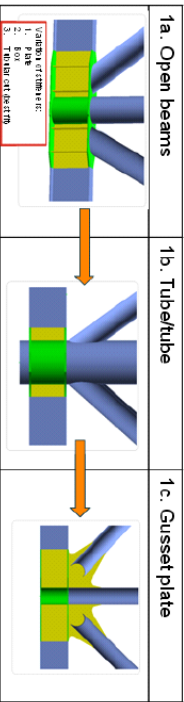


Figure to the left illustrate the process of choosing node type in Edward Grieg project. First, the node-type 1a was chosen (open beams)with the first variation of stiffeners (plate stiffeners). Analysis then was conducted. If the analysis failed, the next stiffener variation was tested for open beams. If, neither the next variation node, the next one did not work in the analysis, the next node type (1b, Tube) was tested. If analysis failed with tube node, gussset plate node was chosen.

### FAQ

What fabricated node types are supported in KBE?

->Answer: Open beams, Gussset plate, Beam Box.

### Further Reading

Technical knowledge K-briefs:

- > Box Nodes
- > Gussset plate
- > Open beams

Process knowledge K-briefs:

-> [DESIGNING IN KBE](#)

### Relevant Experts

Name	Role
Roadr Deip	Super User Topside