Preface

The CEO had a fairly pithy response to the presentation of an information model: "That took you four weeks? It's so clear and obvious, I'd have been able to do it in an afternoon". This statement was probably the highest praise that I have ever received in my work as an information modeller. The effort we invested in collecting the information, the painstaking search for (and sometimes coining of) succinct names as well as the discussions we held to resolve uncertainties and contradictions: None of these things was apparent in the outcome. We had described the company's information universe – as confirmed by the boss – clearly, succinctly and accurately. The boss understood the statements on the diagram.

That's what this book is about: How to create shared understanding across all levels? And how to document something that we have all understood? What is the best form of documentation to ensure that other people have the same understanding of the matter as quickly as possible? This book is not about the technical ins and outs of knowledge (storage, data, presentation). Rather, it focusses on content, the essence and the semantics of information.

This book is intended for everyone involved in the management of data and information, be they IT specialists, business analysts, IT organisers, managers or users from the business departments:

- **IT specialists** will learn the difference between data and information modelling and the benefits they bring to communication with IT novices.
- **People involved in IT organisation** will acquire a methodology and language for communicating concisely and reliably with IT specialists, as well as with prospective users from the business departments.
- **Business analysts** will receive methods and a fitting vocabulary to present the findings of their analysis and modelling

as simply and accessibly as possible, ensuring that everyone involved in the process is in the know.

- **Managers** will quickly acquire a tool-kit providing insight into the requirements and illuminating the solutions. They can then identify the right solution, without getting bogged down in the technical details. Doing so allows them to ask the right questions and to detect and remedy troublesome developments at an early stage.
- **Users** can identify and verify their own contributions to the selected solutions. The simple presentation of knowledge from a user perspective ensures that they feel involved in the project. They can communicate with IT specialists on a level playing field and recognise how their personal perspectives of the informational world are incorporated in future software products.

Stefan Berner, July 2016

Preface to the English edition

Since the German original was published, information modelling has been applied in dozens of projects. I'd like to share some feedback I got from customers:

- "Since we began applying the technique of information modelling, we can discuss in meetings without quarrelling about each term."
- "This model represents the DNA of our enterprise."
- "We were able to solve an issue, that had been bothering us for years, in just one afternoon."

I would like to thank my employer foryouandyourcustomers for the generous support, that made this English version possible.

A special *thank you* goes to Jonathan Möller, Stephan Müller, Christoph Gerber for their input, and to my wife Marie-Theres for her endurance and understanding for my frequent real and mental absences.

Stefan Berner, October 2018

Introduction

Good software

Software crises have been around since the first keystroke of code was written. A variety of studies indicate that between 40 and 80 percent of all IT projects never see the light of day. Although hard to verify, these figures suggest that billions of euros are being tossed out the window on poorly conceived software ventures. Developing proprietary software is risky and generally too expensive. Often, the use of standard software turns out to be more costly than expected, and the additional expenditures associated with rolling it out will ultimately exceed any savings from the lower cost of purchase. Software systems don't fit together; interfaces are complex and buggy. There is no shortage of compelling examples that the quantum leaps in computer sciences refer more to the technology (storage, clock rate, conductivity) than they do to the content or quality.

There are, however, documented, established methods for the development of good software. So why is so much of it poor, although it was developed by specialists using proven techniques? Assisted by business analysts, users describe the requirements and concepts that – from their perspective – reflect their wishes correctly and completely. Highly qualified computer scientists use modern methods and tools to write software that meets these requirements. Yet still the customers are still dissatisfied. Even leaving aside the usual suspects and sources of errors like carelessness, ineptitude, sloppiness, poor work ethic, a haphazardly assembled team and suchlike, it is far from unusual that good people do good work and still produce an unacceptable result.

Software quality rests on the entirety of properties and property values of a software product which influence its ability to satisfy defined or expected requirements[1]. It follows, therefore, that clients perceive software to be *good* software if it fulfils their expectations.

IT specialists generally have a firm grasp of their methods and tools and are good at their jobs. People within the departments and management know what they need. They are familiar with the technical workflows and have wishes or perceptions of how they would like to work. The peripheral systems are also usually well known. So it is less the question of which knowledge exists and more of how it can be translated into future software products. Ignorance does not lead to bad software but the inefficient application of existing knowledge and substandard communication on the interface between the real and abstract worlds. And the problem is merely compounded by the unshakeable belief among all stakeholders that they've understood what everyone else wants.

This book is based on the following proposition:

> Poor software is mainly caused by a lack of shared understanding.

How do misunderstandings occur? Why are people so often at cross purposes, although they speak the same (natural) language? Each environment (companies, departments, countries, cultures, etc.) has terms that are used and understood by everyone. It's the common parlance of everyday life. But frequently the vocabulary is imprecise, and the person using it is prone to assuming that the recipient of the message will interpret the terms exactly as they were intended. How can computer scientists and IT specialists – who frequently come from a different environment than their clients – become familiar with the internal jargon used in a company? Are they even able to understand the specifications and wishes expressed by their clients? What can be done to help them acquire the specific language of an unfamiliar environment?

People often believe they understand things straight away. They assume that other people have the same expectations as their own. So even when everyone at a meeting shares the confident belief that they've understood what was said, it is by no means certain that they in fact did. *Understanding* is always dependent on the perspective, the area of action, the prior knowledge, the environment or – in a nutshell – the context.[1]

1 This for our purposes should be taken to mean a mixture of language, culture, education, experience, attitude, interests, etc.

But software projects frequently involve collaboration between people who do not possess the same contextual knowledge: external consultants, freelance programmers, suppliers, managers, departmental factotums and IT wizards, all of whom brim with different levels and areas of education.

A shared context needs to be created as a matter of urgency to ensure unambiguous communication in heterogeneously assembled groups. This context must be documented in a manner that all stakeholders understand. Clear and unequivocally defined terms, and their clear and unequivocal use, are one of the essential factors, if not *the* essential factor, for fruitful communication and therefore good software. Put succinctly, everyone needs to speak a common language.

> The language will become muddled if the names and terms are out of sync.
> And muddled language leads to chaos and failure.
> Where there is chaos and failure, decency and moral standards will decline.
>
> Confucius (551–479 BC)

Understanding

Allow me to introduce myself using three attribute values from our personnel database:

Stefan	Berner	1955

This information takes me right to the heart of this section. Why do you understand it? Put differently, would you have understood

Martin	Peter	8472

as well? Why not?

In the first example, your grasp of our shared culture and linguistic understanding probably allowed you to recognise the two initial words as a first name and a surname. Your assumption is based on the fact that you're reading this book in English and that you

would recognise words like Stefan, Martin and Peter as first names. The composition of the number, as well as the prior announcement that the author would introduce himself – perhaps a photo of me that you might also have seen – created a context. I assume you concluded it was the year of my birth.

But you are unable to understand the second data-set without additional, explicit knowledge. If you assume that the structure of the first example corresponds to the common practice in English-speaking countries of stating the first name in front of the surname, it is quite probable that you interpreted Martin to be the first name. But you can't be certain. Your interpretation might have been different if you hadn't already seen the first example. So you need structural or contextual information (which is the first name, which the surname?) to be confident that you have understood data values the way that the author intended.

The number in the second example is quite evidently not a year of birth. Given the amount, it could be a monthly salary or a bank balance. In actual fact it is a Swiss postcode. Readers living in Switzerland may have recognised it as such. But that also casts doubt on the interpretation of 1955 as a particular year. The context (same position, same number of numerals) seems to suggest that both numbers have the same underlying meaning. And indeed, 1955 (besides being the year of my birth) is indeed a postcode for Chamoson in the Canton of Valais.

Let's take our interpretation a step further. We know that the information comes from a personnel database, so it is reasonable to assume that the first name and the surname belong to an employee. But the data cannot be understood clearly without knowing the significance of the postcode. What is the link between a town (which in Switzerland is usually associated with a postcode) and an employee? Does he *live* there, *work* there, *grow up* there – or is it perhaps his *birthplace*?

We can use a table (figure 1 on the facing page) to document the contextual knowledge needed to understand the data values. The illustrative data above are added to the context description in the first two rows. Figure 2 on the next page provides a graphic re-

presentation of the contextual knowledge on its own (without data values).[2]

Figure 1: Tabular representation of the initial example.

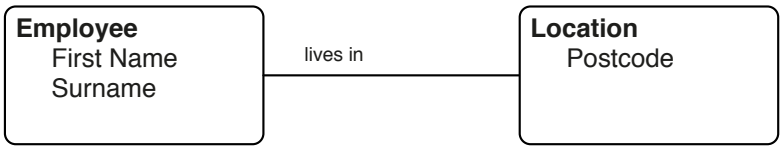

Figure 2: Graphic representation of the initial example.

Let's return again to the data values.

Martin	Peter	8472

You are unable to interpret the three values as the author intended if you are lacking the contextual knowledge. Data (values) on their own are *meaningless*. They are quite literally a jumbled collection of characters. Only after interpretation by the reader do they acquire a significance. It is impossible, based on data values alone, to understand clearly and unequivocally what they might mean. Most readers will recognise *Martin* and *Peter* as male first names. But the fact that they might also be surnames illustrates beautifully that values alone do not produce unambiguous meaning. Let's take the following values as another example: Zurich, Bern, Basle, Geneva. Got it? Are they cities in Switzerland? Or perhaps Cantons? Maybe they are the names of conference rooms in a company? Or the names of railroad engines run by the Swiss Railway?

2 This form of representation will be explained in chapter *Elements of the information model* beginning on page 24.

Every form of communication (verbal, visual, textual) requires context information. We rely on this context in every situation. It is the basis for interpretation and therefore our understanding of what we see and hear. Where it is not provided, explicitly in the form of a model or through syntax and grammar, each person implicitly applies their own personal contextual knowledge to suit the situation. Socialisation, environment, education and personal expertise blend and merge into this personal contextual knowledge. But for two people to interpret the same values in exactly the same way, they must apply an identical context to the data. This can work only if they are both familiar with the shared context and have agreed on the application of precisely this context to the task at hand. Only when these conditions are met will confusion be avoided. Only then will an identical interpretation be possible. Only then will the stakeholders understand each other.

> Understanding describes nothing other than creating a situation in which a variety of people engaging in communication interpret the same data values by applying the same context.

Of course, these principles apply beyond computer science. They are true of all areas in which unambiguous, unequivocal communication is wanted or needed. We do not need to concern ourselves at this point with areas in which clarity is not welcome. Jokes, for instance, acquire much of their poignancy through the introduction of unexpected contexts. And literature would be as dull as dishwater if it didn't leave room for personal interpretation. Personal conversations and artistic renditions often communicate information through body language, melody, colour, form and other techniques. But this book deals exclusively with communication by linguistic and semiotic means, in which clarity for all people in all situations is essential.

The desired, common context allows all project stakeholders to engage in unambiguous communication at the interface between the real and the technical worlds (see figure 3 on the facing page).

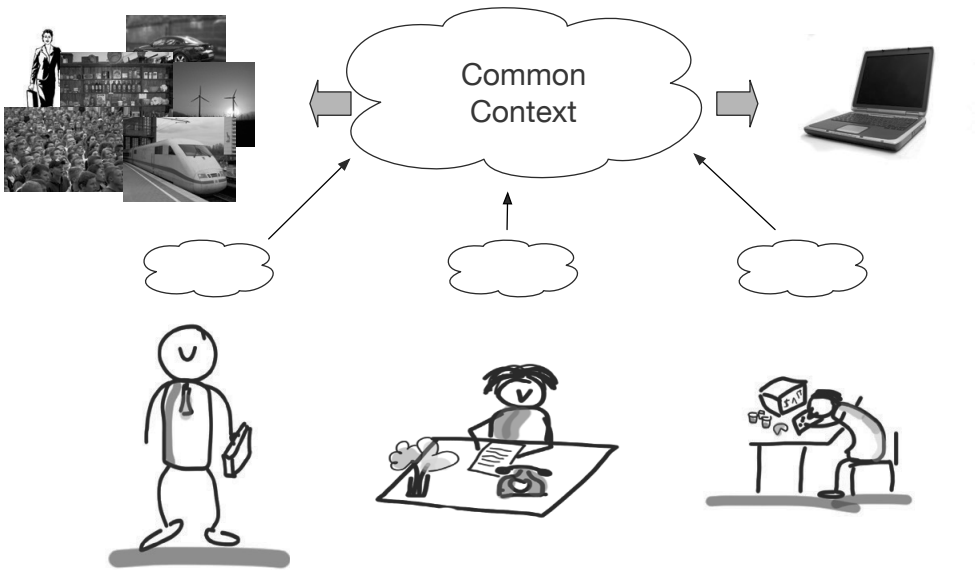

Figure 3: Context as the connecting link between the real world and IT.