The Design and Emergence of a Data/ Information Quality System

Soffi Westin
University of Agder, Norway
soffi.westin@uia.no

Maung K. Sein
University of Agder, Norway and Luleå Teckniska Universitet, Sweden
maung.k.sein@uia.no

Abstract. Poor data and information quality (DQ/IQ) has remained a consistent problem plaguing both the practitioner and academic communities in Information Systems (IS). The consequences of poor DQ/IQ is particularly severe in Construction Engineering, and the field lacks sufficient DQ/IQ assessment frameworks and tools. To address this shortcoming, we applied an action design research (ADR) approach to develop and implement a DQ/IQ assessment tool called Information Quality System (IQS). The multi-year research project took place in a European construction engineering company, and lasted from 2007 to 2012. We drew upon insights from the literature on DQ/IQ assessment and related challenges in construction engineering, as well as practical lessons learned from managing DQ/IQ in the target organization. Through our research, we develop a set of design principles for meeting DQ/IQ challenges.

Key words: data quality, information quality, construction engineering, action design research.

1 Introduction

In the IS literature, the cost of insufficient Data/Information Quality (DQ/IQ)[1] has been found to be very high (Ramaswamy 2006; Strong et al. 1997). This problem has been observed across numerous organizations (Ramaswamy 2006; Wand and Wang 1996). Awareness of these issues has grown rapidly over recent years, and DQ/IQ research has moved on from technical issues,

Accepting editor: Christina Keller

such as query techniques on multiple data sources and data warehouses in the 1980s, to a number of new application areas, such as knowledge management (Madnick et al. 2009) and health care registers (cf. Pipino and Lee 2007; Vician 2011).

To address the issue of the persistence of DQ/IQ problems, IS researchers have called for more investigation in new and different contexts (Madnick et al. 2009). Recently, attention has turned to the context of construction engineering (cf. Lin et al. 2008; Tribelsky and Sacks 2011), which is a field where the consequences of poor DQ/IQ are particularly severe. Even though the body of DQ/IQ literature includes several assessment frameworks (cf. English 1999, 2003; Lee et al. 2002; Pipino et al. 2002), these frameworks and tools cannot be used in construction engineering. This is because their focus is typically on business systems and primarily on accuracy of the data (Neely et al. 2006). This means the 'correct answer' has to be known in order to perform the comparison of inserted data with real world data. This assumption is not valid for construction engineering, where the current paradigm of concurrent engineering is based on parallel processes: tasks that previously were executed in sequences are now executed in parallel. Consequently, engineers have to proceed with incomplete information (Blechinger et al. 2010). Hence, construction engineering is in need of assessment frameworks and tools adapted to its specific context.

The purpose of this paper is to present our solution to this problem. We tell the story of the design, emergence, evaluation and implementation of a tool we call Information Quality System, or IQS for short. Originally framed as a prototypical Design Research project (Hevner et al. 2004), its later stages were conducted as an Action Design Project (Sein et al. 2011). In keeping with the principles of ADR, IQS is targeted at a class of problems, namely, DQ/IQ problems. We carried out our research in a large global construction engineering company head-quartered in Europe (anonymized as European multi-discipline construction engineering company—EU-MEC)

Despite the mentioned shortcomings, existing frameworks can nevertheless provide useful insight on processes and principles related to DQ/IQ assessment in construction engineering. Two such frameworks informed our research. The first framework, the Data Quality Assessment (DQA) framework developed by Pipino et al. (2002) provides guidelines on how to measure data quality in an organization. The framework consists of a list of data dimensions, a set of objective and subjective measures, and a suggestion on analysing and comparing these measures. The methodology distinguishes between *subjective* and *objective* quality metrics. Subjective data quality assessments are tied to the needs and experiences of the different stakeholders, such as collectors, custodians, and consumers. How stakeholders perceive the quality of data influences their behaviour. Objective data quality assessment can be either task dependent or task independent. Task dependent metrics include business rules, regulations and policies, and constraints provided by the database administrator—all of which are specific to the given context. Task independent metrics reflect states of data without any contextual knowledge; e.g.; the extent of missing data, or the extent of inconsistency between records in different databases, and can as such be applied to any data set.

While measuring is essential, it addresses only half the problem of poor DQ/IQ. The measurement also has to be performed on a continuous basis. A framework that focuses on this issue is Total Information Quality Management (TIQM), as proposed by English (1999, 2003). Framed as a methodology for assessing existing quality tools and techniques, it stresses that the key is to view DQ/IQ management and improvement as a continuous process. The framework emphasizes principles, techniques and processes used for total quality management since continuous process improvement sometimes is confused with continuous data cleansing. The latter does not go into correcting the root cause of errors; hence, the process will not improve. Another important aspect of DQ/IQ is measuring data accuracy, which in reality can only be done by comparing the data to the physical object. Use of surrogate sources will only reflect accuracy to the extent that the surrogate source is considered accurate. For construction engineering design, accuracy is almost impossible to measure since the physical object is often not yet produced. Through developing IQS, we aimed at mitigating the consequences of this problem; that is, how to perform useful DQ/IQ assessment on a dataset where the correct data is unknown by the outset.

The rest of the paper is organized as follows. In the next section, we set the scene by narrating the story of how the problem arose and developed in EUMEC. Next, we describe the construction engineering process, specifically situating it in EUMEC to help the reader understand the context of our research. We then interpret and analyse our project through the lenses of design research and action design research approaches and show how our project generated design principles for the class of DQ/IQ problems. Following this analysis, we present the theoretical contributions of our research and end the paper with a reflection on the practical and theoretical implications of our findings.

2 The Problem and the research context

2.1 Problem description

The problem came to light when site managers in EUMEC, started complaining to project managers that drawings which the assembly sites were receiving from the detailed engineering design section contained errors. Since the actual physical construction starts in these sites with assembly of the numerous physical parts (also known as assets) and since the assembly was done based on these drawings, any inaccuracy in the drawings significantly delayed the rest of the project. EUMEC operates in the oil and gas sector and has assembly sites all over the world.

Project managers reacted to these complaints in the time honoured manner: they sent more people to the assembly sites to try to work out the problems. These work hours represented cost overruns, and the problems eventually delayed the handover of the installations to the customers. Major projects undertaken by EUMEC take up to 3 years and cost in the region of 100 million Euros. The impact of delays in a project can be huge.

Alarmed at this trend, EUMEC started a number of discussions at several levels in the period 2007-2008. Many of these meetings were cross-departmental. Information managers (IM) from the IT/IS department attended engineering meetings where these problems were referred to as "problems on site", "problems with equipment interfaces", and "problems with finding the right parts and where they belonged". The first author of this paper was one of the IMs. To reflect this first hand engagement in the project, we will write the rest of the paper in first person plural.

We studied the existing literature on delays and cost overruns in construction engineering projects and found that these problems were quite prevalent. Amongst several reasons indicated were errors and omissions in drawings (Rivas et al. 2010; Toor and Ogunlana 2008). Slowly, it dawned on us that these errors could have something to do with the low data quality in the engineering databases. After all, the drawings in the target organization were generated precisely based on these data. However, we could not yet devote our attention to examining this hunch. Our regular work load was high, and much of it was spent in handling day-to-day problems on ongoing projects. Projects were committed to milestones with deadlines for deliveries, and penalties and bonuses were tied to these milestones. It was thus very important that the projects reached these milestones on the agreed date, fulfilling the agreed delivery requirements. It was quite understandable that management prioritized fire-fighting.

This management priority left little time for us to investigate whether our suspicion about the relationship between data quality and errors in engineering drawings was correct. However, the management mandate to reduce the need for fire-fighting also serendipitously handed us the key to turn the focus back on the data quality issue. We started an investigation on the matter, studied the documentation of requirements, and assessed the level of DQ/IQ in several completed projects. Our findings appeared to support our insight that the main reason for the delays in the projects was in fact related to poor DQ/IQ in the engineering databases. Some of these issues could actually be avoided had we the time to perform some serious assessment of the data quality at an earlier stage of the projects. There was an urgent need to have in place a mechanism or tool to assess the data quality at EUMEC. The company was using a tool, but the IM department found this tool to be inadequate.

At this juncture, the IM who is the first author of this paper entered a doctoral program and the search for an appropriate tool became her dissertation work. The outcome was IQS.

2.2 The nature of construction engineering projects

Large construction engineering projects are characterized by the large number of people involved, the need for different kinds of expertise, and huge costs. The projects are complex and challenging to manage (Franco et al. 2004). In addition, short delivery schedules have resulted in a break in the traditional linear engineering model which has led to the current iterative nature of concurrent engineering (Dobson and Martinez 2007). This causes challenges for DQ/IQ management, for while the processes involved are interdependent, the tasks that previously were executed in sequences are now executed in parallel. The result is that the engineers are forced to proceed with incomplete information (Blechinger et al. 2010).

Another serious challenge is that existing DQ/IQ assessment frameworks and tools are based on the assumption that the correct answer is known by the time of data insertion to a record. This assumption makes these frameworks insufficient for use in construction engineering (Neely et al. 2006). In a review of the construction engineering literature, Westin (2013) found that several challenges related to construction engineering inevitably lead engineers to proceed in performing tasks without knowing the correct answer (or data). Figure 1 shows the links be-

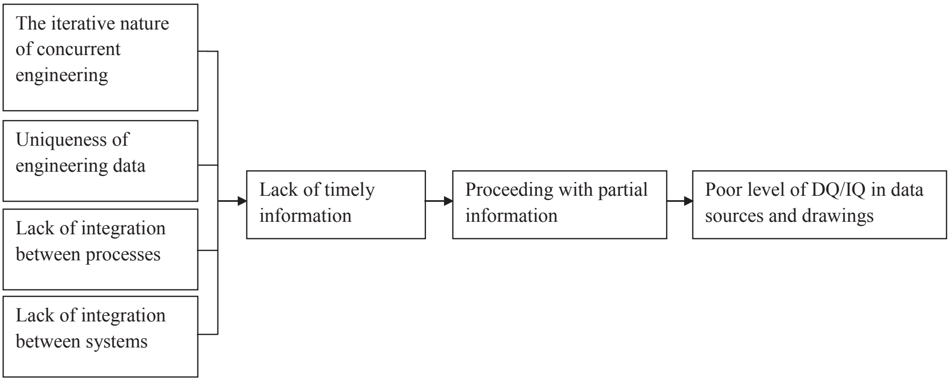
The iterative nature of concurrent engineering
Uniqueness of engineering data
Lack of integration between processes
Lack of integration between systems
Lack of timely information
Proceeding with partial information
Poor level of DQ/IQ in data sources and drawings

Figure 1. Connections between challenges and level of DQ/IQ (Westin, 2013

tween these challenges and levels of DQ/IQ. The four challenges listed on the left side are invariant circumstances in construction engineering. These lead to lack of timely information. Faced with delays, and the pressure of short delivery schedules (Dobson and Martinez 2007), engineers are forced to proceed with incomplete information (Blechinger et al. 2010) which leads to poor DQ/IQ.

2.3 Engineering projects at EUMEC

EUMEC operates in the oil and gas industry, and its core businesses are construction engineering design related to complex offshore installations such as oil rigs and drill ships, and product development within the same area. The company delivers engineering design for construction projects and possesses a significant share of global markets in its product and project domains. Most of its employees are involved in construction engineering projects.

EUMEC's engineering projects are parts of a larger construction project and usually cover design-related activities. Engineering projects have phases of their own that are usually described in some sort of project execution model (Dyrhaug 2002). Figure 2 depicts a model similar to the one used at EUMEC.

In the Feasibility & Concept phase, alternative solutions are identified, and one or two design concepts are selected for further work. At the end of this phase, one concept is chosen and some requirements related to this concept are 'frozen', meaning that no further changes are, or should be, allowed to aspects such as choice of scope/concept and completed detail design. In the System Definition phase, drilling processes and equipment concepts are frozen, together

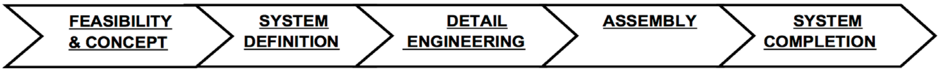

Figure 2. Project phases

with layout and main structure. Towards the end of this phase, global design and equipment design are completed based on the chosen concept. In the Detail Engineering phase the design details are completed. The final deliverables of this phase mainly consist of documents, such as drawings and manuals. The drawings are either from suppliers or extracted from various data sources used in engineering design such as 2D/3D applications and engineering data bases. When the design is complete, these deliverables are handed over to the Assembly phase.

Actual physical work, mainly mechanical in nature, is done in the Assembly phase based on the drawings received from Detail Engineering phase. A test called Factory Acceptance Test (FAT) is performed and used for checking if the equipment meets the requirements and is fully functional. The System Completion phase includes system assembly, commissioning and close-out. 'System' in this context means the entire construction which makes up a system of equipment, pipelines, valves, etc. The most serious problems, especially delays, typically occur in these latter phases. The crux of the challenge for EUMEC, therefore, is to improve the data quality in the drawings handed over from the detail engineering phase to the assembly phase.

Project teams at EUMEC are set up with members from organizational units called 'engineering disciplines' (Table 1).

Several experts from different engineering disciplines are needed to design large, complex, robust and yet delicate constructions. The exact composition of the teams varies depending on the requirements of the individual projects. Table 1 lists the engineering disciplines typically represented in projects at EUMEC. Each engineering discipline is composed of several engineers and has a manager who is responsible for the delivery as a whole. Every engineering discipline depends on input from the other disciplines throughout the project, due to a large number of interfaces and dependencies among the artefacts and systems to be designed and assembled. Tight project schedules means that various activities must be performed in parallel. In such concurrent engineering (Sekine and Arai 1994), quality assurance and possible adjustments are therefore conducted both during the project and then again in the assembly and completion phases.

3 Method

Our research method was Action Design Research (ADR), which we followed implicitly[2] in the earlier stages of the project and explicitly in the later stages. ADR is a Design Science Research (DR) method which also draws on Action Research (AR). In DR, IT artefacts are created and evaluated with the intention to solve organizational problems. The practical implications of the

Disciplines	*Responsibility*
Process	Design of industrial processes; all the facts, sequences and relations in the process and a logical placing of the different items.
Mechanical	Design (choice of equipment and its physical layout and weight).
Piping/Layout	Design of all piping.
Electro	Design and cabling of power distribution for electrical systems: equipment, lights, heat, etc.
Instrument	Design of control systems; i.e.; the control of various valves, machines, the alarm systems, and instrumentation cables for distributing signals.
Telecom	Selection and location of radio and audio systems, alarms etc.
HVAC (Heating, Ventilating and Air Conditioning)	Capacity calculations and layout for ventilation etc.
Safety	Various safety assessments.
Structure (steel)	Design of steel structures, supports, outfitting like hand rails, stairs etc.
Architecture	Interior design.

Table 1. Engineering disciplines in EUMEC's construction projects (Westin and Päivärinta 2011)

designed artefact should impact the evaluation of the scientific research performed (Hevner et al. 2004). However, Sein et al. (2011) criticized DR because it "does not fully recognize the role of organizational context in shaping the design as well as shaping the deployed artefact" (p. 38). To address this shortcoming, they proposed ADR, which draws upon AR to bring in a stronger emphasis on organizational intervention. Hence, ADR is a method for performing IT artefact design that emerges from interaction with an organizational context. The method considers organizational intervention a requirement for development, as it balances what the authors consider "the conflicting demands of (1) addressing a class of problems, and (2) intervening in authentic settings" (p. 39). Organizational intervention is crucial to ADR because of its underlying philosophy that the artefact is not only based on design, but also emerges from interaction with an organizational context. This view is different from existing DR methods, which focus on the technological issues related to the IT artefact and pay less attention to the impact the organizational context may have on the artefact when the two interact. Figure 3 shows the stages and principles of ADR. In the rest of this section, we briefly describe the stages.

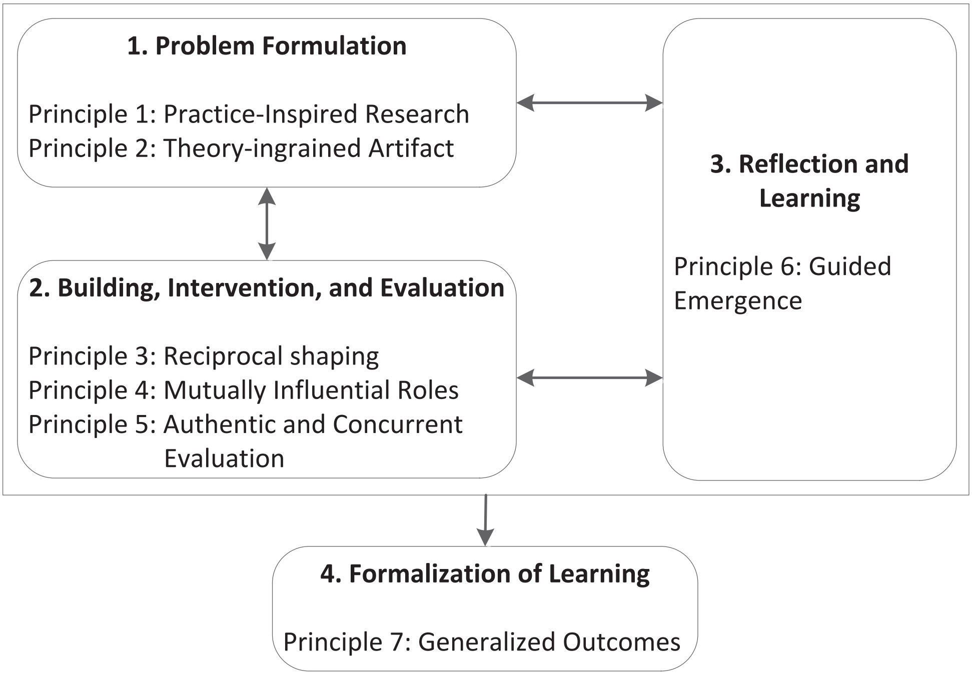
1. Problem Formulation
Principle 1: Practice-Inspired Research
Principle 2: Theory-ingrained Artifact
2. Building, Intervention, and Evaluation
Principle 3: Reciprocal shaping
Principle 4: Mutually Influential Roles
Principle 5: Authentic and Concurrent Evaluation
3. Reflection and Learning
Principle 6: Guided Emergence
4. Formalization of Learning
Principle 7: Generalized Outcomes

Figure 3. ADR Stages and Principles (from Sein et al. 2011)

Problem Formulation: The problem is encountered by a target organization and/or acknowledged by researchers as a possibility to enhance scholarly knowledge. A research question can be defined during an initial empirical investigation of the problem. Casting the specific problem as an instance of a class of problems moves the problem solving effort from a level of mere consulting to a level where new knowledge is generated.

Building, Intervention, and Evaluation (BIE): The initial design of the artefact is based on the premises of stage one. Through an iterative process that requires intervention in the organization, the artefact is further developed (built) and evaluated. Design principles are defined for a class of systems.

Reflection and Learning: This stage runs in parallel with the previous stages and provides the conceptual move from building a solution for a specific instance to applying that learning to a class of problems. This is where it becomes evident that the research process is more than solving a problem. To identify contributions to knowledge, conscious reflection is critical. The learning from these reflections is used to adjust the research process accordingly. The adjustments are performed continuously as the understanding increases.

Formalization of learning: The knowledge generated from designing the artefact is used for generalization by developing general solution concepts for a class of problems. By describing how the artefact design was performed, the accomplishments realized, and the organizational outcome, the researcher formalizes the learning. The design principles embedded in the artefact instance contribute to theory building for the class of systems.

4 The IQS project

In this section, we first describe the chronology of the development of IQS, framing it as an ADR project. The initial phase was not explicitly conducted as ADR but can be retrospectively interpreted as such. ADR was not published at the time when the project was begun in 2007, and so we could not possibly start a project with that method. On reflection though, we realized that we had been implicitly following ADR although not in its complete form. After the publication Sein et al. (2011) on ADR, we were convinced that ADR was the right approach for IQS and decided to follow it in its entirety for the remaining phases.

Figure 4 (next page) shows the timeline of the IQS project with the main events keyed. In narrating the story of the project, we will refer to this figure and specifically the keys. We organize this section around the stages of ADR.

4.1 Problem formulation

Investigating the poor data quality in engineering databases: In section 2, we described how the problem arose (pt. A), what existing literature indicated (pt. B), and how the cause was tracked down to poor DQ/IQ (pt. C). We began our investigation of how poor levels of DQ/IQ affected the engineering drawings by exploring the data values in the main engineering data base (pt. D). This database featured a query tool for detecting missing values. Users could automatically check whether fields in a record contained a value. The intention was to identify blanks in fields that were subject to internal or external (customer) requirements. An example would be to check if the site code field contained a value. Since site codes indicate where an item should be shipped for assembly, a value is required. We decided to take a closer look into the results of these checks because there was more information missing on the drawings than was reported as missing by the query tool. We found that in addition to blanks, there was a widespread use of so-called “straw values”. These were characters and values totally unrelated to the definition of the expected field value. The most frequently used characters were ‘NA’ and ‘-’. When we asked the design engineers why they inserted these straw values, the three most common answers were (these comments were made in Norwegian which we have translated):

> The field is not subjected to any requirements of filling so I inserted one of those values [‘NA’ or ‘-’] to avoid the field from becoming a part of a ‘missing values report’.
>
> I know the requirements state that a value is needed for this field, but I don’t know the correct value yet. I am not sure whether to leave it blank in the meantime so I sometimes insert ‘NA’ or ‘-’.

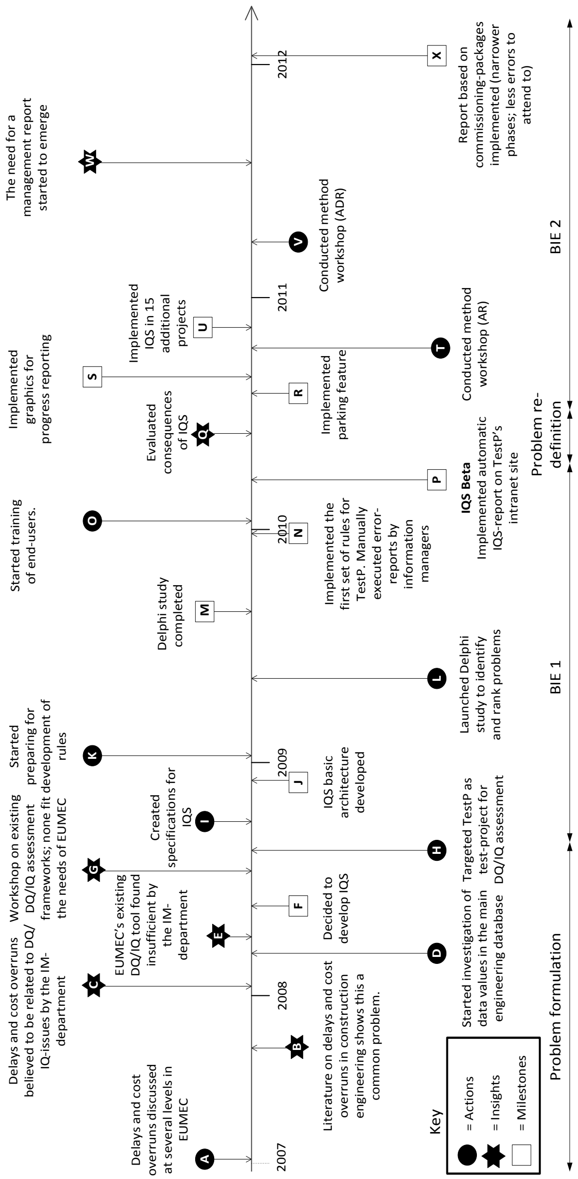
2007
2008
2009
2010
2011
2012
A
Delays and cost overruns discussed at several levels in EUMEC
B
Literature on delays and cost overruns in construction engineering shows this a common problem.
C
Delays and cost overruns believed to be related to DQ/IQ-issues by the IM-department
D
Started investigation of data values in the main engineering database
E
EUMEC's existing DQ/IQ tool found insufficient by the IM-department
F
Decided to develop IQS
G
Workshop on existing DQ/IQ assessment frameworks; none fit the needs of EUMEC
H
Targeted TestP as test-project for DQ/IQ assessment
I
Created specifications for IQS
J
IQS basic architecture developed
K
Started preparing for development of rules
L
Launched Delphi study to identify and rank problems
M
Delphi study completed
N
Implemented the first set of rules for TestP. Manually executed error-reports by information managers
O
Started training of end-users.
P
IQS Beta
Implemented automatic IQS-report on TestP's intranet site
Q
Evaluated consequences of IQS
R
Implemented parking feature
S
Implemented graphics for progress reporting
T
Conducted method workshop (AR)
U
Implemented IQS in 15 additional projects
V
Conducted method workshop (ADR)
W
The need for a management report started to emerge
X
Report based on commissioning-packages implemented (narrower phases; less errors to attend to)
Key
● = Actions
★ = Insights
□ = Milestones
Problem formulation
BIE 1
Problem re-definition
BIE 2

Figure 4. Timeline of the project

> I do not need to know this value to perform my job, so I simply insert 'NA' or '-' to indicate the value is of no interest.

The management's perception of the situation is exemplified by the following quote:

> In the projects there is no clear philosophy for who are responsible for inserting values in the various data fields in the main engineering data base. At the management level we agree that somebody has to be the 'father' of the information to be inserted, but the users responsible for the engineering assets, who we think should be those 'fathers', say they have never been told to insert this information. (Engineering manager)

The outcome of our investigation can be summed up as follows. In situations where a data value had to be inserted in order to proceed, straw values were inserted. These were either a best-guess value or a random temporary value. Data values were also often omitted. In all cases, the intention was to insert the correct values as soon as they were known. On top of incomplete or meaningless straw values, our exploration revealed more problems with data quality. There were inconsistencies between different databases and lack of logical coherence between different fields in the same records of the main engineering data base. An example of the latter was the relation between areas and items. Areas are imaginary 'boxes' that cover the whole construction. Each area is defined by X, Y and Z coordinates, and given a unique area code. All items are also given X, Y and Z coordinates which indicates the placement of the items. In addition, all items are given an area code. Lack of logical coherence would be if the coordinates entered for an item was outside the coordinates belonging to the entered area code.

The number of records to assess for incorrect values was far too large to be handled manually, especially since several data sources were simultaneously in use; e.g.; 3D-models and engineering systems. This, combined with the fact that the existing DQ/IQ tool could only check and flag missing values, made it extremely difficult to detect errors related to consistency or logical coherence.

The resulting missing or incorrect values in the database led to incorrect drawings. These incorrect drawings were sent to the assembly site which created all manners of problems for the site workers. The drawings had to be corrected, which entailed finding the missing or incorrect data first. The ultimate result was delays and cost overruns:

> The errors in the drawings are revealed on site and that leads to a lot of extra work. (Engineering manager)

Having found the nature of errors in the engineering database and their causes, we now turned our attention to the existing DQ/IQ tool in use at EUMEC. Our investigation revealed a number of its inadequacies and shortcomings which are summarized in Table 2, (pt. E).

Decision to develop a DQ/IQ assessment tool: Based on the findings of our investigation described above, a decision was made to develop a tool from scratch (pt. F). The rationale to develop a new system is captured succinctly in the following quote:

> Now the best way to perform data quality checks is manual assessment. But then you have to know exactly what you are looking for, and the problem is that you don't really know. Also, manual checking is very time consuming. We need automatic assessment to

Problem	*DQ/IQ tool in use*
Errors in drawings. The errors are related to completeness, consistency and logical coherence.	Supported only searches for missing values . Supported only assessment of the main engineering database.
Rigid reporting style by allowing only analysis of predefined parameters.	Did not support project specific or changing requirements.
Correct answer not always known by the time of data insertion resulting in insertion of straw values or blanks.	Did not detect or flag straw values as errors.

Table 2. Problems with the existing DQ/IQ assessment tool at EUMEC

> cope with the huge amount of data and to always know the current status of the data quality. (Engineering discipline manager)

We now needed to find an existing DQ/IQ assessment framework that we could use to build the tool. It is here that we ran into a roadblock: none of the existing frameworks met our needs (pt. G). New ground had to be broken. This situation coincided with the start-up of a new engineering project (TestP) which we could use as a test project (pt. H). We identified three main stakeholder groups: the information management (IM) department, engineering projects, and end-users. Together with the researchers, representatives from these three groups formed what we now realize was the ADR team. It was at this stage that the first author of this paper started her doctoral work. The challenges of the innovative nature of the project transformed it into a research project framed as designing an innovative artefact to address a class of problems. As we stated earlier, the project started as a Design Research project and later evolved into an Action Design Research project.

In the following, we describe in detail the project through which we developed the artefact called Information Quality System (IQS). This section described the first stage of ADR, Problem Formulation. The next section describes the second stage, Build-Intervene-Evaluate (BIE). In this stage, the initial design of the artefact is generated and then further shaped by use and in subsequent design cycles (Sein et al. 2011). As we shall see, there were two BIE cycles in our project.

4.2 BIE Cycle 1: Designing for error detection and reporting

An artefact developed through ADR is theory-ingrained: it carries 'traces of theory' that guides its design and development (Sein et al. 2011). The precise nature of the theory remains a matter of debate. We followed Lempinen (2012) and Markus et al. (2002) and used the characteristics of construction engineering (Westin 2013) as the kernel theory.

- Characteristic 1: *The iterative nature of concurrent engineering, which includes distinct interdependent phases conducted concurrently in the individual projects*
- Characteristic 2: *The uniqueness of engineering data*
- Characteristic 3: *Lack of integration between processes*
- Characteristic 4: *Lack of integration between systems*
- Characteristic 5: *Lack of timely information*
- Characteristic 6: *Proceeding with partial information*

We decided that IQS should be rule-based. Rule-based systems are commonly used in data quality assessment (Hayes-Roth 1985; Xu 2012). Knowledge in various forms, for example practitioners' experience or project requirements, can be translated into formal statements (or rules) (Breuker 2013) and then be used for identification of nonconformity in various data sets. Examples of the rules in our case are shown in Table 3 later in this section.

The rationale for using a rule-based approach in IQS was as follows:

- It was possible to capture requirements in rules, which also makes it possible to identify errors.
- It was possible to reuse rules in other projects because some rules could be applied to all projects.
- The rules were fairly easy to program once the requirements were understood by everybody.
- Developing rules would help clarify the project requirements so that it would be possible to agree on a rule set.
- The rules would be easy to change or edit since they would be located outside the system source code. This was important for EUMEC because changes in requirements would be almost inevitable during projects. Adjustments, changes, and additions to the construction design require immediate on-site amendments to the rules throughout a project. Even data sources may be added, or customer systems changed, which could affect the rules.
- The rules could be viewed as business knowledge in a way that makes it possible to create a set of default rules applicable to all projects as well as project specific rules. For EUMEC, this capacity for reuse was important because it would provide the employees with a knowledge base that could be transferred between projects and also used for training.

Requirement Specifications for the System: Based on the kernel theory, we were now ready to define the requirement specifications for the system (pt. I). These are described next.

Requirement Specification 1: IQS should collect needed data from all needed systems.

This was needed to address the lack of integration between systems.

> *Requirement Specification 2:* IQS rules should address three DQ/IQ dimensions: completeness, consistency and logical coherence.

Thus, rules should identify which data could possibly be missing, which data could possibly be inconsistent, and which data could lack logical coherence. 'Quality dimensions' are an important aspect of DQ/IQ assessment and they represent groups of data values with the same qualities; e.g.; accurate or consistent values. Identifying quality dimensions would make it easier to define and discuss issues related to DQ/IQ without referring to specific data values.

> *Requirement Specification 3:* IQS should provide different rules for different phases of the project with increasingly more details to be checked at every subsequent phase.

Due to the huge amount of data involved and the fact that much of the data would not be available at the start of the project, a full check on every requirement from day 1 would result in an unwieldy number of errors. The solution was to define different rules for use at different phases of the project.

These three system requirements could be implemented through a rule-based system. The system could assess whether the project data collected from various systems (specification 1) complied with the required quality dimensions for the project (specification 2) while differentiating the degree of details to be assessed by the rules during the various project phases (specification 3).

By December 2008 the requirement specifications were ready. During the Christmas holidays, the basics of the IQS architecture were developed (pt. J). The programmer stated that "the specifications were so detailed that the programming itself did not take very much time".

Developing rules: In January 2009 we started to prepare for development of rules according to TestP's requirements (pt. K). We were inspired by statements from top management that encouraged cost saving initiatives. A quotation from the CEO illustrates this:

> Cost control on site will be a major focus in the months to come as significant amounts of money can be saved if we use shorter time on site. (CEO of EUMEC)

Spurred by this statement, we launched a Delphi study in June 2009 (for details see: Westin and Päivärinta 2011). The Delphi method is commonly used for identifying and ranking issues for decision making, and has been applied to a broad "variety of situations as a tool for expert problem solving" (Okoli and Pawlowski 2004, p. 16). Our goal was to identify and rank problems that had the most negative impact on the projects' profit margins, and for that, we needed expert opinions from experienced engineers. The heavy workload of these engineers and the fact that they were working in different construction engineering projects made it almost impossible to gather them for group discussion or focus group meetings. The Delphi method allowed us to use the convenience of email. This non-direct confrontation also had another advantage: the experts tend to think more independently and gradually reach a considered opinion without having to defend a stand once taken, which often is the case when using direct confrontation (Dalkey and Helmer 1963). The outcome of the Delphi study was revealing. Of 18 ranked problems, eight, including the top six, were DQ/IQ related. Moreover, five of these DQ/IQ related problems pointed at missing or incorrect information in various types of drawings and/or other documentation. Even the remaining problem was indirectly related to incorrect drawings. The findings of the Delphi study documented and strengthened our argument that DQ/IQ related problems were worth our close attention (pts. L and M).

We developed the rules based on project requirements, operating under the motto "everything that can be formalized can be assessed by rules" (Chief developer). These requirements were of two types: common and project specific. Common requirements applied to every project at EUMEC and reflected internal issues such as organizational policies influencing data values and internal coding manuals. Project specific requirements mainly reflected customer needs and such contingent issues as the type of construction to be designed (types of ship or type of oil rig), and the information systems to be used during design (customers sometimes require systems outside the organization's regular portfolio).

To meet Requirement Specification 3 we had to divide the project into assessment phases. An existing procedure at EUMEC provided us with a basis for doing this. Engineering projects at the company are required to follow an internal Project Execution Model (PEM). PEM provides an overview of all phases in a project, a detailed explanation of requirements for each phase, and shows the milestones for deliveries; e.g.; drawings. We asked expert engineers from the various engineering disciplines to tell us at which phase of a project the correct data values were supposed to be known, and hence when they could be expected to be correct in the database. Based on their responses, we divided the project into three phases aligned with existing PEM phases. In the first phase, we defined rules to check only for the existence of data that should be known by the end of that phase. In the following phases, the rules were progressively 'tightened' to check more and more data fields whose values should be known by then. This evolution of the rules reflected the increasing need for correct detail information as the project progresses. The process culminated in the last phase where everything needed to be correct; i.e.; every data field would be checked against the determined rules.

To illustrate this process of gradual tightening of rules checking, we take the example of pipelines. Larger pipelines are designed and modelled before smaller pipelines. Since the properties of larger pipelines are commonly known before the properties of smaller pipelines, data records belonging to larger pipelines can be expected to contain known and correct values before the records belonging to the smaller pipelines do. Therefore, the larger pipelines' data records can be assessed before the smaller pipelines' data records. Data for smaller pipelines would be inserted into the data records with only an ID field which is needed for referencing. The remaining values could be inserted later when the engineers would have the needed information, such as the type of liquid or gas it would carry. In this phase the smaller pipelines' missing values could be ignored by the rules. This way, the engineers could focus on the larger pipelines and use the rules to identify DQ/IQ errors related to those only. The smaller pipelines could be included in the rule definitions at later stages.

By the end of 2009, an initial set of rules were ready for use in TestP (pt. N). Table 3 lists two examples of rules.

Project requirement	*Rule description*
All main equipment have to be installed at the Yard.	All equipment records are checked to determine if the equipment is main equipment. If so, the rule also checks if the site-code field has a value that indicates Yard (all site-code fields contain a value indicating where to ship the equipment). If the value is not Yard, an error is reported.
All equipment IDs should reflect NORSOK standard.	All equipment IDs are checked against the NORSOK standard. If the value does not comply with NORSOK (there are many other standards) an error is reported.

Table 3. Examples of rules

Implementing and refining the set of rules for TestP: We manually executed the rules (pt. N) and discussed the reported errors with the engineers through emails and in organized meetings. We also started the development of a daily report that would run automatically and would be visible on TestP's home intranet site. From January 2010 we participated in weekly follow-up meetings with focus on training end-users in TestP and providing them with clarifications when they asked (pt. O). These meetings brought about a need for refinement of the rules as the common understanding of TestP's requirements became clearer through these meetings. Below, we provide two examples of exchanges from these meeting:

Example 1: Clarification of requirements.

> Does anybody know which site code to use for HVAC items? (Design engineer)

Explanation: The site code is used to indicate where an item (HVAC in this case) is to be assembled. Each project usually has several assembly sites identified by a unique site code. These sites can be located at different places all over the globe. If the site code for an item is incorrect, the item will be shipped to a wrong location. Shipping items are costly and time consuming to begin with. To return and re-ship items are even more costly. Hence, it is important to insert the correct site code but first one needs to know the code.

Example 2: Unclear system response.

> I have referred to a P&ID for one of my items, but an error is reported: "This P&ID is not referred to in the Line list." What the h.. does that mean?? (Design engineer)

Explanation: P&ID is a Piping and Instrumentation Diagram which shows the piping of the process flow together with equipment and instrumentation. The Line list contains all the P&IDs used in a project. To register an item in a database, engineers need to refer to the P&ID to which the item is connected. To find the relevant P&ID, they consult the Line list. In the Line list all P&IDs used in the project are listed. The error message here was meant to indicate that the P&ID that the engineer was referring to did not exist in the Line list. In this case the error message was not very well formulated. A better formulation that would have made more sense would have been: "The P&ID referred to by this item does not exist in the Line list."

In April 2010 the automatic IQS report was implemented. The aim was to provide the engineers with up-to-date error reports which they could assess whenever they wanted to. This would enable them to focus on errors that they considered most important for immediate handling. This meant the beta version of IQS now was completed and rolled out (pt. P). Several types of activities followed: The training sessions from January continued; another set of weekly meetings were established with a focus on further refining the rules; a third set of meetings focused on clarifying the rules and the concept of DQ/IQ.

During this period, the engineers had several questions about the project requirements related to DQ/IQ. A recurring question from them related to a rule: "How did you know that this was a requirement?" We had to repeatedly explain that we had assessed the requirements for the project in collaboration with discipline managers, the document control manager, and the engineering manager. As time went by and more and more rules were developed and refined, the engineers began to perceive the error report as a working list for correcting errors to meet the requirements. They assumed all DQ/IQ related requirements were covered by the rules.

This assumption had an unfortunate effect every time a new rule was added. A new rule meant more errors reported; hence the total number of errors actually increased gradually even though the engineers had corrected several errors earlier. This surprised them greatly. During this period we also refined a number of aspects of IQS, such as the basic architecture of IQS and the interfaces with internal and external (customer) applications.

The regular meetings held throughout the period of implementation and refinement also served as a forum for evaluating IQS (pt. Q). The ADR team and the engineers evaluated IQS from several perspectives such as: do the principles established result in the anticipated outcome? Is the number of errors manageable? Do the engineers find IQS useful? In the next section, we present the reflections from these evaluative sessions.

4.3 Reflection and learning from BIE Cycle 1

Four design principles were generated through the process of building IQS, intervening by implementing it in TestP, and evaluating its effect.

These principles are shown in the left-hand column of Table 4. Our evaluation revealed both anticipated and unanticipated consequences which are shown in the right-hand column of Table 4.

The first three principles, allow for inconsistency, incompleteness, and lack of logical coherence, had anticipated outcomes. The engineers were generally satisfied with several aspects of the system. They appreciated that IQS assessed not only completeness, but also inconsistency and logical coherence. They also mentioned that they got a better total overview because the assessed data were collected from all relevant data sources.

> Assuming the rules are correctly defined, IQS is a nice tool. (Electro engineer)
>
> If we try to do it manually [assessing data] it takes a lot of time and you also have to know exactly what you are looking for. IQS detects almost everything and fast. (Discipline manager)

Design Principle	*Consequences*
Allow for inconsistency.	Reduced errors. Stopped spending time manually identifying inconsistencies and focused on errors that could be corrected (anticipated).
Allow for incompleteness	Reduced errors. Stopped spending time manually identifying incompleteness and focused on errors that could be corrected (anticipated).
Allow for lack of logical coherence.	Reduced errors. Stopped spending time manually identifying logical incoherence and focused on errors that could be corrected (anticipated).
Phase-based reporting (report only identified errors applicable for current phase).	• The IQS report used as a working list for correction of identified errors (anticipated). • The number of reported errors perceived as still too large. Engineers found it difficult to prioritize errors to correct first, leading to de-motivation (unanticipated). • IQS generated 'false positives' leading to reduced trust in IQS and consequently to de-motivation (unanticipated).

Table 4. Consequences of Building, Intervention, and Evaluation of IQS

> It is so much easier to use the IQS report to identify discrepancies between the main engineering system and the 3D modelling tool, it provides a better overview than if you try to do it manually. (Mechanical engineer)

The principles affected the way the engineers worked: they stopped spending time on manual identification of possible errors and focused on reported errors. Other than that it was business as usual, since the existing engineering design process was anyway based on the insertion of straw values: data that were best-guess values, random temporary values, or blanks. Thus, while these three design principles epitomized the error management paradigm, they only reflected existing engineering design practices.

However, the fourth principle, phase-based reporting, proved to be a challenge. Weekly focus session discussions, participant observation, and semi-structured interviews revealed that the engineers at EUMEC appreciated the intentions of the report and used it as a working list for correcting errors. The report enabled them to focus directly on the errors instead of manually trying to identify them first. However, the implementation also revealed unanticipated consequences. First, although this principle was intended to reduce the number of reported errors, engineers complained that the number was still unmanageably large. They felt that as a result, they found it difficult to prioritize which errors to correct first. The mere number of errors was in some sense demotivating. Second, the system identified some data values as incorrect when they were actually correct. These false positives reduced trust in IQS. Both these unanticipated consequences demotivated the engineers from using the system.

4.4 Problem (re)definition

The unanticipated consequences of the first BIE Cycle led us to take another look at the problem of the large number of false positives in the report. It initiated a thorough review of the rule definitions. We were able to weed out some of them by identifying inaccuracies in the rule definitions. Even after that, we were still left with the majority of the false positives. After several discussions with managers and engineers of each engineering discipline, where we dissected each occurrence of false positive, we realized that most of these errors actually represented exceptions from requirements. The discipline managers had agreed upon these exceptions with the customer.

Identification of exceptions revealed another problem: It was difficult to determine any common denominators of these exceptions that could be captured in a manageable set of rules and thus exclude them from the report. It was of course possible to define a rule for every exception, but the huge number of errors made it a staggering task. Besides, new exceptions would almost certainly arise in the future. We had to find a more manageable way to deal with existing and future exceptions.

The other unanticipated consequence was the unmanageably large number of true errors that IQS was reporting. That led us to think of better ways to group errors for reporting. The goal was to find a solution that would both decrease the amount of errors to manage at the time, and help engineers in prioritizing errors. This redefinition of the problem led to a second cycle of BIE.

4.5 BIE Cycle 2: Designing for error handling

Handling false positives: Our solution for handling false positives was simply to remove those errors from the report and park them in a separate table which we called the 'parking table' (pt. R). The rationale was this: these errors were detected by the project specific rules whose definition was based on a formalized set of requirements. Handling exceptions requires departing from this formalized set, which is a serious action. Hence, to keep track of all these exceptions, they had to be first reported as errors and handled later. The parking table did that. The table contained the errors, the cause of the error and the signature of an authorized person who was either the concerned discipline manager or someone delegated by the manager. At hand-over this information could then be attached as a supplement to the deliverables. This report helped explain why the data deviated from the original set of specifications. We formalized this in a design principle which we call the 'Parking principle'.

Handling the large number of true errors reported: The phase-based principle was aimed at reducing the total number of errors reported by grouping the errors by phases. This was not achieved: the number of reported errors remained large. Our discussions with the engineers and discipline managers revealed that the phases were still too large a unit for grouping errors. We needed a smaller unit. After searching for a unit of optimal size, we settled on 'commissioning packages'. A commissioning package is "a practical scope of work unit within a system or subsystem for commissioning, constituting a functional unit which can be tested by commissioning to confirm its suitability for operation" (NORSOK 1999). The deliverables for all systems belonging to a certain functional unit possess the same delivery date. A project has several such commissioning packages. By grouping all data belonging to a commissioning package, the number of reported errors would be significantly smaller than the number of errors reported for several units in that project phase. In addition, the delivery dates would provide a basis for prioritizing the most urgent packages first. Based on these considerations, we divided the original three report phases into several narrower phases based on commissioning packages and their delivery dates (pt. X). This led us to revise the phase-based principle by adding explicitly that the phases should be optimally narrow.

4.6 Reflection and learning from BIE Cycle 2

The engineers were satisfied with both solutions. Commenting on the parking of errors solution, an electrical engineer said: "It is nice to get rid of these false errors quickly. If we use some time to identify the exceptions in the first place, we will have more time to focus on correcting real errors."

A typical feedback on the solution to report errors by commissioning packages was:

> Now it is possible to prioritize which errors to correct first. The small amounts of errors per package also result in higher motivation for correcting them simply because it feels possible to actually be able to correct everything. (Mechanical engineer)

The evaluation and reflection led us to revise one design principle derived from BIE-1 and add a new one. Table 5 lists the revised set of design principles. The changes from the previous set are shown in italics.

Design Principle	*Description*
Allow for inconsistency	Inconsistencies caused by best guess values, random temporary values, or blanks, should be allowed at appropriate early phases.
Allow for incompleteness	Incompleteness caused by best guess values, random temporary values, or blanks, should be allowed at appropriate early phases.
Allow for lack of logical coherence	Lack of logical coherence caused by best guess values, random temporary values, or blanks, should be allowed at appropriate early phases.
Phase-based reporting	Preliminary incorrect values should not affect the daily report until the appropriate project phase is reached. *The phases should be narrow enough to produce a manageable number of errors and provide means for prioritizing.*
Parking of errors	*False positives occurring as a result of (legitimate) deviation from original (and still applicable) requirements, should be removed from the error report and saved together with an explanation for deviation to be handled later.*

Table 5. Revised set of Design Principles

Other events in BIE Cycle 2: We also made several enhancements to IQS during BIE Cycle 2. We implemented graphics for progress reporting (pt. S) and held workshops on research methods for the ADR team and the engineers. The first was on Action Research (pt. T) and the second one was on ADR (pt. V). Participating in these workshops helped the engineers to appreciate our efforts and to have an insight into their roles in a project that could simultaneously develop a practical solution and generate academic knowledge. However, perhaps the most effective achievement was the spread of the idea of IQS. By then, news about IQS had spread throughout EUMEC, and other projects wanted the tool. From February to November 2010, 15 other projects implemented IQS on their home intranet sites (Pt. U). To make it easier for the project participants in these 15 projects to start using IQS, a default set of rules was developed. The default set contained rules applicable to all projects. Every project would then have to add project specific rules based on project specific requirements. Due to the lack of information management resources and willingness by some projects to budget such resources, different projects received IM support to varying degrees. One of the experiences from TestP was in fact the need for extensive support. We noted that those projects with an internal champion used time to learn and use IQS, while those projects that lacked such champions simply used a default set of rules not fully aligned with the project's requirements for deliverables.

5 Discussion

In this section we first summarize our project by depicting the BIE cycles and listing how our main actions map to the ADR processes and adhere to the ADR principles. We then discuss how data quality issues related to the characteristics; i.e.; challenges, of construction engineering projects were mitigated through the use of IQS.

5.1 Summary of the case

Figure 5 captures the two BIE cycles, each of which addressed specific design challenges. The right-hand side of the figure summarizes the contributions of the IQS project.

The left-hand side of Figure 5 lists the participants in the IQS project and shows how we as researchers actively participated (and intervened) together with practitioners on all levels.

Table 6 summarizes how the project meets the ADR principles, and the main actions listed illustrate the close collaboration between the researcher and the practitioners participating in the project.

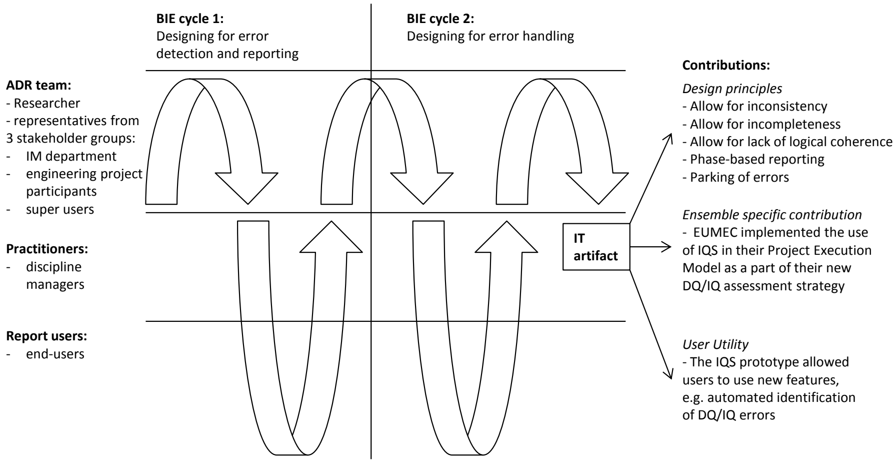
BIE cycle 1:
Designing for error detection and reporting
BIE cycle 2:
Designing for error handling
ADR team:
- Researcher
- representatives from 3 stakeholder groups:
- IM department
- engineering project participants
- super users
Practitioners:
- discipline managers
Report users:
- end-users
IT artifact
Contributions:
Design principles
- Allow for inconsistency
- Allow for incompleteness
- Allow for lack of logical coherence
- Phase-based reporting
- Parking of errors
Ensemble specific contribution
- EUMEC implemented the use of IQS in their Project Execution Model as a part of their new DQ/IQ assessment strategy
User Utility
- The IQS prototype allowed users to use new features, e.g. automated identification of DQ/IQ errors

Figure 5. BIE cycles and contributions of the IQS project

5.2 Mitigating challenges of construction engineering through IQS

Figure 6 provides an overview of how the characteristics (challenges) of construction engineering projects, the project requirements, and the information from the two guiding frameworks were embedded in IQS. The informing perspectives are those pictured at the bottom (Characteristics of Construction Engineering Projects; Construction Engineering Project Requirements; DQA; and TIQM). The IQS entities (Design Principles; Rules; Reports; and PEM are shown inside the oval. Although the arrows point directly from a specific informing perspective that is embedded in a specific IQS entity, all the perspectives in some way informed the overall design of IQS.

The five design principles mitigate the challenges arising out of the very nature of construction engineering which inevitably leads engineers to proceed with only partial or even incorrect information. If not managed properly, this leads to poor DQ/IQ in data sources and drawings. In short, the three first design principles (the 'allows') are there to avoid reporting on missing or incorrect data at an early phase of the project where the correct data are unknown anyway. At all times the data that are supposed to be known in a particular phase are reported for deviations. This is managed by the 'Phase-based reporting' principle. Finally, legitimate exceptions from requirements are parked with an explanation for why this is not an error and authenticated by an 'authorized by' signature. This is the 'Parking of errors' principle.

A rule-based system provided the means for capturing requirements from various sources and formalizing those requirements in executable statements represented as rules. We elaborated on the rationales for using rules in section 4.2. The reports generated based on these rules formed the basis for objective task-dependent assessment, which is one of the assessment types

ADR principles	*The ADR process in the IQS project*	*Main actions*
Principle 1: Practice-Inspired Research	Research was driven by the need for better DQ/IQ assessment tools for construction engineering projects.	Studied documentation. Targeted test-project for DQ/IQ assessment. Conducted Delphi study for identification and ranking of problems. Assessed existing DQ/IQ tool.
Principle 2: Theory-Ingrained Artefact	The theory used was based on existing literature describing unique characteristics of engineering data such as not knowing the correct answer, several disparate data sources, and the large amount of data produced. Two quality assessment frameworks were used to guide the development of IQS: DQA (Pipino, et al., 2002) and TIQM (English 1999, 2003).	Assessed existing DQ/IQ literature in general. Conducted a review of DQ/IQ research in construction engineering. Identified challenges. Selected frameworks for guidance.
Principle 3: Reciprocal Shaping	Errors in data sources and drawings were expected to be an ongoing problem. In collaboration with the practitioners, problems were iteratively addressed and design principles were formulated.	Created requirements for IQS. Developed IQS basic architecture. Developed IQS basic rules.
Principle 4: Mutually Influential Roles	The ADR team included researchers and representatives from three groups of stakeholders: IM department, engineering project, and end-users. One of the designers was an employee from EUMEC who was also a PhD student.	Participated in actions concerning training, support, requirement clarifications, etc. Refined of IQS rules.
Principle 5: Authentic and Concurrent Evaluation	IQS was first evaluated within the ADR team, then in the wider setting of end-users at EUMEC, and finally through a comparison of the level of DQ/IQ in TestP and two other projects not using IQS.	Conducted interviews. Assessed IQS reports.
Principle 6: Guided Emergence	The preliminary design of IQS was continuously reshaped through use and feedback from project participants. During BIE iterations, refinements of IQS were performed based on anticipated and unanticipated consequences.	Developed parking feature. Developed report based on commissioning packages.
Principle 7: Generalized Outcomes	A set of design principles for DQ/IQ assessment systems was articulated (see Table 5). IQS was positioned as an instance of such systems.	Formulated design principles (DPs). Prepared for dissemination of DPs.

Table 6. Mapping IQS project to ADR principles

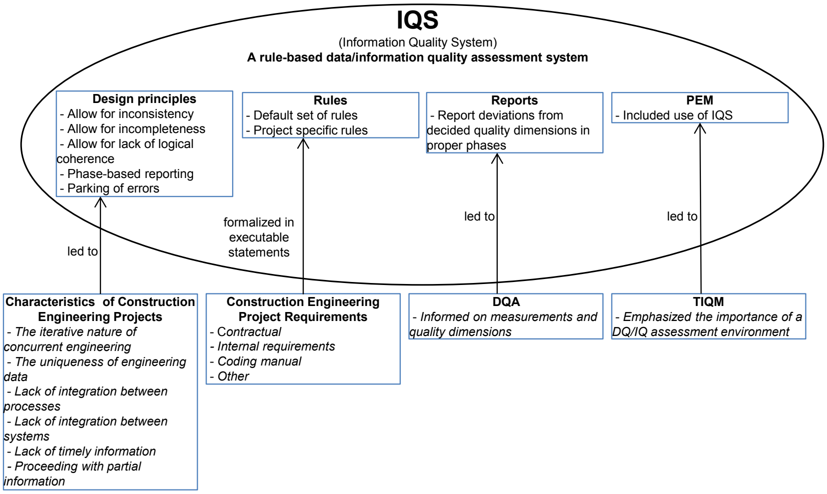
IQS
(Information Quality System)
A rule-based data/information quality assessment system
Design principles
- Allow for inconsistency
- Allow for incompleteness
- Allow for lack of logical coherence
- Phase-based reporting
- Parking of errors
Rules
- Default set of rules
- Project specific rules
Reports
- Report deviations from decided quality dimensions in proper phases
PEM
- Included use of IQS
led to
formalized in executable statements
led to
led to
Characteristics of Construction Engineering Projects
- The iterative nature of concurrent engineering
- The uniqueness of engineering data
- Lack of integration between processes
- Lack of integration between systems
- Lack of timely information
- Proceeding with partial information
Construction Engineering Project Requirements
- Contractual
- Internal requirements
- Coding manual
- Other
DQA
- Informed on measurements and quality dimensions
TIQM
- Emphasized the importance of a DQ/IQ assessment environment

Figure 6. Embedding of informing perspectives in IQS entities

in the DQA framework. Task-dependent assessment is typical when contextual characteristics are a challenge, which was the case here. In addition, the outcome also needed subjective assessment since only experienced engineers could reveal that some 'errors' were indeed legitimate exceptions from requirements. Finally, DQA also suggested defining relevant quality dimensions; in our case these were Completeness, Consistency, and Logical Coherence.

Finally, managing DQ/IQ is a continuous process. The premise for embodying this aspect came from the TIQM framework which emphasizes that the key to achieving a satisfactory level of DQ/IQ is to view DQ/IQ management and improvement as a continuous process, and establishing an information quality environment. In our case we established a methodology for evaluating and refining IQS, including necessary alignments of various processes. The activities related to IQS were included in the organization's Project Execution Model (PEM), and Information Management roles were included in projects.

5.3 Contributions

ADR studies are expected to produce generalized outcomes of the problem instance and the solution instance, and to derive design principles (Sein et al. 2011). Below, we elaborate how in developing IQS, our study met these expectations.

The class of problems addressed by our study was DQ/IQ problems. The specific context was construction engineering, which can be also understood as the area of concern (Mathiassen et al. 2012). Through this study, we answered calls to IS researchers to extend DQ/IQ research to new contexts (Madnick et al. 2009), and to provide practitioners with tools that can assess the level of DQ/IQ (Pipino et al. 2002). In addition, we identified DQ/IQ challenges unique to the context of large construction engineering projects.

The class of solutions addressed by our study was the extant approaches used to solve DQ/IQ problems. These approaches were mainly represented by the frameworks that we used to guide the development of IQS, namely DQA (Pipino et al. 2002) and TIQM (English 1999, 2003). These approaches were built on the assumption that the correct data value to insert in a record was known by the time of insertion. Thus, the extant paradigm was *error detection and correction*. Our approach was shifting the paradigm to *error management*. We elaborate further on this later in this section. Our study specifically demonstrated how a rule-based approach was suitable for capturing project requirements that needed quality assessment.

The development of IQS resulted in a set of *design principles* (see Table 5) that (a) mitigate the problem of not knowing the correct data values to insert at the time of insertion to a record, and, (b) handle the reporting of an inevitably huge number of identified errors by introducing phase-based reporting and the parking of identified errors that for some reason are not to be considered errors. An example of such false positive is a legitimate deviation from the original, but still applicable, requirements

In addition, we identified quality dimensions that are relevant for construction engineering projects: completeness, consistency, and logical coherence. In the rest of this section, we elaborate on these contributions to knowledge and reflect on the IQS project in a holistic manner.

Essentially, IQS helps to balance two equally important requirements that act in opposite directions. The first is that the projects need to proceed swiftly and with minimum delay. The second is that data should be accurate so that drawings can be accurate. This second requires a thorough check of the data to detect errors, but that would delay the project and thus go against the first requirement. An error detecting tool would first have to correctly identify errors and then report them. The detection of errors is not straightforward. Due to the concurrent nature of construction engineering, data cannot be complete, or accurate or even coherent at the start of the project. So in order to meet the first requirement—keep the project flowing—short cuts have to be taken in inserting data values. This is done knowingly, and if managed properly, will not harm the project's progress. That is what IQS does. IQS also speeds up the process of error detection and brings consistency and uniformity to the process by using rules which automatically detects errors.

IQS thus represents a shift in paradigm on how DQ/IQ errors are defined and handled in construction engineering by bringing in the context—namely the challenges faced by engineers. All that is reported as error is not error. Some are 'required errors' because no one knows the correct data at that time. Others are not really errors but are exceptions. Once the users—the engineers—understand this, errors become manageable. IQS does not improve DQ/IQ by correcting errors. It simply detects errors and reports them in a manageable way. The actual correction of the errors is still the responsibility of the users—engineers in this case.

Taken together, the five design principles that emerged from the development of IQS underscore the error management paradigm. The first three principles (the three 'allows') keep the project flowing by allowing short cuts and insisting only on the known correct data. The next one, phased based reporting, reduces the number of errors reported so that engineers can prioritize the errors that need to be corrected. The last one, parking of errors, keeps the project flowing by handling exceptions to requirements that would be detected as errors. IQS recognizes this. However, it also recognizes that as long as there is a record somewhere that marks these as exceptions, and identifies the person who has indicated that these are exceptions, the project continues to flow without harm.

In addition, our study also *contributed to practice* by solving an actual problem. EUMEC's projects were plagued by delays and cost overruns. We tracked the cause of the problem to the low level of DQ/IQ, and our solution was developing IQS. By comparing a project that used IQS with two projects that did not, we demonstrated that the system alleviated DQ/IQ problems (see Westin and Sein 2013).

Finally, our study also contributed to the method space. We had implicitly followed ADR as our research method at the outset and explicitly at later stages of the project after the publication of the Sein et al. (2011) paper. Nevertheless, we can suggest a modification to the method. In ADR, Stage 4 is framed as the final stage where outcomes are formalized when the project is completed. In our study, the formalized outcome was fed back to Stage 1 and informed the problem formulation for the next iteration of BIE. For example, we initially used the successive phases of PEM as a basis for implementing stricter rules for error checking. This led us to formulate one of the design principles, phase-based reporting. When this was implemented in the BIE stage, we realized the phases were too long: the number of errors was still too large, making it difficult for the engineers to manage and correct the errors. This led us back to the problem formulation stage and we reformulated the problem as: what is the optimal phase to make the reported number of errors manageable? The answer was to report per commissioning package based on delivery dates of the packages. It is worth considering whether a feedback path from Stage 4 to Stage 1 should be added to the ADR process to capture unanticipated consequences that were vital for new iterations of BIE.

6 Limitations and future research directions

As with any other empirical study, our study has its limitations, but these limitations also offer opportunities for further research. First, at the time of writing, the ADR project was still ongoing. Thus, the design principles that we are reporting may not have reached their final form yet. Refinement of the principles may occur and new principles may be added. It is likely that when IQS has been in use for an extended period and in several projects, it will mutate. Consequently, changes and additions to the design principles will emerge. Following this development at EUMEC over this extended period would be interesting and insightful. In addition, it would be of interest to extend the research to other organizations. The findings from this study could be useful to other researchers as a starting point for such extended research.

As shown in the time line (Figure 4), the need for a management report was about to emerge at the time of writing. The need for such management reports that communicate the level of DQ/IQ and provide comparative assessment over time has been articulated in the literature (Pipino et al. 2002). Not including such a report can be seen as a limitation of this study. This issue is related to the limitation described next.

A limitation of IQS is that the errors it reports are not weighted. Some errors may have more severe consequences than others on delays and cost overruns. To be able to provide a single aggregated measure of the level of DQ/IQ, weighting of the various variables (the rules identifying errors in this case) is needed (Pipino et al. 2002). If the organization has a good understanding of the importance of each rule defined, then a subjective weighting can be used (ibid). Another alternative would be to empirically determine the degree of delay caused by each error type. This can be done by gathering data at assembly sites of several projects. Such an investigation could provide the information needed to weight the rules. Either way, we can get more accurate information on which of the DQ/IQ errors are more important to weed out in construction engineering. This could also enhance the value of a management report.

Another limitation of this study is that we did not demonstrate that an increased level of DQ/IQ also improved overall project performance. This would be an essential extension of projects such as this. Future studies may focus on the impacts of DQ/IQ assessment systems in construction engineering, more specifically those related to the correlation between DQ/IQ on the one side, and delays and cost overruns on the other.

Finally, we are left with an intriguing question. An artefact such as IQS includes features that make it possible to detect errors that would not have been possible through manual searches. The question is whether the time consuming work of capturing project requirements in assessment rules outweighs the time used at the assembly site for correction of errors. More research is needed to investigate these issues.

Notes

1. Data/Information Quality (DQ/IQ) has been defined in various ways. Data Quality often refers to technical issues, while Information Quality usually refers to non-technical issues (Madnick et al. 2009). For our purpose we do not distinguish, but adopt the “fitness of use” perspective (Wang and Strong 1996) which is based on the intended purposes of the users of information.
2. The description of ADR was first published in Sein et al. (2011) as described in section 4.

References

Blechinger, J., Lauterwald, F., and Lenz, R., (2010). Supporting the production of high-quality data in concurrent plant engineering using a metadatarepository. *Proceedings of the Americas Conference on Information Systems,* August 12–15, Lima, Peru.

Breuker, J., (2013). A cognitive science perspective on knowledge acquisition. *International Journal of Human-Computer Studies*, (71: 2): 177–183.

Dalkey, N., and Helmer, O., (1963). An experimental application of the Delphi method to the use of experts. *Management Science*, (9: 3): 458–467.

Dobson, D., and Martinez, R., (2007). Integrated engineering. *ABB Review*, (SPEC. REP.): 48–52.

Dyrhaug, Q., (2002). *A Generalized Critical Success Factor Process Model for Managing Offshore Development Projects in Norway*. Doctoral thesis, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

English, L. P., (1999). *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits*, John Wiley and Sons, New York, New York, USA.

English, L. P., (2003). Total information quality management: a complete methodology for IQ management. *Dm Review* (Now: *Information Management*), (9): 1–7.

Franco, L. A., Cushman, M., and Rosenhead, J., (2004). Project review and learning in the construction industry: embedding a problem structuring method within a partnership context. *European Journal of Operational Research*, (152: 3): 586–601.

Hayes-Roth, F., (1985). Rule-Based Systems. *Communications of the ACM*, (29: 9): 921–932.

Hevner, A. R., March, S. T., Jinsoo, P., and Ram, S., (2004). Design Science in Information Systems Research. *MIS Quarterly*, (28: 1): 75–105.

Lee, Y. W., Strong, D. M., Kahn, B. K., and Wang, R. Y., (2002). AIMQ: A methodology for information quality assessment. *Information & Management*, (40: 2): 133–146.

Lempinen, H., (2012). Constructing a design framework for performance dashboards. In: *Nordic Contributions in IS Research,* C. Keller, M. Wiberg, P. J. Ågerfalk and J. S. Z, Eriksson Lundström (eds.), Springer, Heidelberg, pp. 109–130.

Lin, S., Gao, J., and Koronios, A., (2008). A data quality framework for engineering asset management. *Australian Journal of Mechanical Engineering*, (5: 2): 209–219.

Madnick, S. E., Wang, R. Y., Lee, Y. W., and Zhu, H., (2009). Overview and framework for data and information quality research. *Journal of Data and Information Quality*, (1: 1): 1–22.

Markus, M. L., Majchrzak, A., and Gasser, L., (2002). A design theory for systems that support emergent knowledge processes. *MIS Quarterly*, (26: 3): 179–212.

Mathiassen, L., Chiasson, M., and Germonprez, M., (2012). Style composition in action research publication. *MIS Quarterly*, (36: 2): 347–363.

Neely, M. P., Lin, S., Gao, J., and Koronios, A., (2006). The deficiencies of current data quality tools in the realm of engineering asset management. *Proceedings of the Americas Conference on Information Systems*, August 4–6, Acapulco, México.

NORSOK., (1999). Z-007 Mechanical Completion and Commissioning. http://www.standard.no/no/Fagomrader/Petroleum/NORSOK-Standard-Categories/Z-MC--Preservation/Z-007/.

Okoli, C., and Pawlowski, S. D., (2004). The Delphi method as a research tool : An example, design considerations and applications. *Information and Management*, (42: 1): 15–29.

Pipino, L. L., and Lee, Y. W., (2007). Applying IT to healthcare: Humans, errors, and a data quality perspective. *Proceedings of the Americas Conference on Information Systems*, August 10–12, Keystone, Colorado.

Pipino, L. L., Lee, Y. W., and Wang, R. Y., (2002). Data quality assessment. *Communications of the ACM*, (45: 4): 211–218.

Ramaswamy, M., (2006). On the phenomenon of information dilution. *Issues in Information Systems*, (VII: 2): 289–292.

Rivas, R. A., Borcherding, J. D., González, V., and Alarcón, L. F., (2010). Analysis of factors influencing productivity using craftsmen questionnaires: Case study in a Chilean construction company. *Journal of Construction Engineering and Management*, (137: 4): 312–320.

Sein, M. K., Henfridsson, O., Purao, S., Rossi, M., and Lindgren, R., (2011). Action Design Research. *MIS Quarterly*, (35: 1): 37–56.

Sekine, K., and Arai, K., (1994). *Design Team Revolution: How to Cut Lead Times in Half and Double Your Productivity*, Productivity Press, Portland, OR.

Strong, D. M., Lee, Y. W., and Wang, R. Y., (1997). Data quality in context. *Communications of the ACM*, (40: 5): 103–110.

Toor, S.-U.-R., and Ogunlana, S. O., (2008). Problems causing delays in major construction projects in Thailand. *Construction Management & Economics*, (26: 4): 395–408.

Tribelsky, E., and Sacks, R., (2011). An empirical study of information flows in multidisciplinary civil engineering design teams using lean measures. *Architectural Engineering and Design Management*, (7: 2): 85–101.

Vician, C., (2011). A resource review for health care quality execution: Business intelligence strategy. *Proceedings of the Americas Conference on Information Systems*, August 5–7, Detroit, Michigan.

Wand, Y., and Wang, R. Y., (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, (39: 11): 86–95.

Wang, R. Y., and Strong, D. M., (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, (12: 4): 5–33.

Westin, S., (2013). Data and information quality research in engineering construction projects: A review of literature. In: *Building Sustainable Information Systems*, H. Linger, J. Fisher, A. Barnden, C. Barry, M. Lang and C. Schneider (eds.), Springer, Heidelberg, pp. 583–594.

Westin, S., and Päivärinta, T., (2011). Information quality in large engineering and construction projects: A Delphi case study. *Proceedings of the European Conference on Information Systems*, June 9–11, Helsinki, Finland.

Westin, S., and Sein, M. K., (2013). Improving data quality in construction engineering projects: An action design research approach. *Journal of Management in Engineering*, (30: 3).

Xu, J., (2012) Rule-based automatic software performance diagnosis and improvement. *Performance Evaluation*, (69: 11): 525–550.