ANALYSIS OF THE USE OF A TOOL TO PERFORM AUDITS ON COINS CONTAINERS

The development of a new automated method for performing audits in BIM processes at Rijkswaterstaat

Technische Universiteit Eindhoven
Master Construction Management & Engineering
Faculty of Built Environment

R.E.L. (Ruben) van der Heijden
July 2016

Analysis of the use of a tool to perform audits on BIM stored in COINS containers:

The development of a new method for performing audits at Rijkswaterstaat.

Author:	R.E.L. (Ruben) van der Heijden
Student number:	0874537
Date:	July 1th, 2016
E-mail:	rubenvanderheijden@live.nl

University:

University:	Eindhoven University of Technology
Faculty:	Faculty of the Built Environment
Master course:	Construction Management & Engineering (CME)

In collaboration with:

Company:	Rijkswaterstaat
Department:	Building Information Management

Graduation committee:

Chairman (TU/e):	prof.dr.ir. B. (Bauke) de Vries
Graduation supervisor (TU/e):	dr.Dipl.-Ing. J. (Jakob) Beetz
External supervisor Rijkswaterstaat:	Ir. H. (Henk) Schaap

TU/e Technische Universiteit Eindhoven University of Technology

Rijkswaterstaat *Ministerie van Infrastructuur en Milieu*

Preface

I hereby present to you this thesis, the result of my graduation project which was completed in collaboration with the Eindhoven University of Technology and Rijkswaterstaat. This thesis represents the finishing research conducted to finalize my master study Construction Management & Engineering.

The research was conducted in the field of Building Information Modelling and the Semantic Web technologies in the Architecture, Engineering, Construction and Facility Management industry (AEC/FM). This field of research proved to be a challenging field for me as a student with a bachelors' degree in Civil Engineering and a student of the master Construction Management and Engineering but with only limited education on the topics behind the semantic web and without any experience with programming languages. As a result, this thesis turned out to be an opportunity for a great learning experience for which I am very grateful.

This thesis is for developers and researchers who want to improve the interoperability of BIM tools through the integration of Semantic Web technologies. The method proposed and developed in this thesis increases the operability of BIM during audits at Rijkswaterstaat through the development of a tool that relies on Semantic Web technology. I hope that this research is used to further develop and research tools and methods to improve the interoperability of BIM technology within the AEC/FM industry.

The people who are experienced with the field of computer programming and the semantic web turned out to be a generous community when it comes to help and support. Many people helped me in personal conversations as well as through online forums whenever my research hit a dead end or when I struggled with the hardship of writing code. First of all, I would like to thank my supervisor Jakob Beetz (TU/e) for his guidance and support in both technical aspects related to the script as well as in the guidance and support to understand the foundations of the theory behind the Semantic Web throughout the entire graduation project. In addition, I would like to thank all employees from the BIM department of Rijkswaterstaat, who generously took the time to help me whenever needed. In special, I would like to thank Frans van Dam, Wouter Pronk and Henk Schaap, who monitored the progression of the project closely and provided me with deep insight in all aspects of the thesis in valuable discussions. More people who I have met and who helped me in various stages of the project and who I would like to mention are: Léon van Berlo, Peter Wilems, Chi Zhang, and the attendees of the BIM colloquium #2 in 2015 and the CIB W78 conference 2015. Finally, I would like to thank my family and friends for all their help, support and patience throughout the project.

I hope you will enjoy reading this thesis and that it will spark more interest in the proposed method for future research and development.

R.E.L. (Ruben) van der Heijden,
Maastricht, May 2016

Summary

The architecture, engineering, construction (AEC) industry can be regarded as a highly collaboration environment with multiple disciplines that require repeated interdisciplinary iterative data exchanges and communications. Continuous effort is being made in the development of new tools to create an interconnected environment in form of Building information Models (BIM) in order to facilitate these data exchanges and types of communication. Currently, BIM is more and more being implemented at Rijkswaterstaat in order to take advantage of the benefits which BIM offers. However, BIM is considered to be not yet fully implemented within all processes at Rijkswaterstaat. Amongst these processes are the verification and validation processes at Rijkswaterstaat, which are known as the 'SCB processes'.

Working according to the principles of the SCB processes in a Design-, Build-, Finance- and Maintain (DBFM) construction project implicates that the responsibility of managing the projects quality control system lie with the contractor. Therefore, a contractor has to verify his own products and processes. In order to check the contractors' work, the contractor is required to deliver proof of the validation and verification of their products and processes to Rijkswaterstaat in a standard prescribed BIM format. The BIM format prescribed by Rijkswaterstaat are COINS containers. COINS containers are a means of communication for object related content and associated documents between various users and contributors of BIM data.

At present, no automated tool exists at Rijkswaterstaat which allows the auditor to perform planned verification and validation tests of COINS containers. Therefore, the auditor has to perform the tests manually and is required to have an extensive knowledge on the structure of COINS containers in order to be able to extract the required data from the container and verify and validate the data based on requirements stored in another database. The aim of this research is to optimize the SCB process by automating the audit process for the auditors at Rijkswaterstaat in order to further optimize the working with BIM and hereby gain more of the benefits of working with BIM. A new method for the SCB process is proposed in this thesis. This method uses a script to automate the data retrieval and the requirements check. In order to improve the user-friendliness of the tool, a graphical user interface is created which allows an auditor to easily perform the check without having to possess technical know-how on how the data is stored.

The BIM data is provided though COINS containers. These COINS containers contain data stored according to the principles of the Semantic Web. The Semantic Web is a knowledge structure used to formally represent and share information through the modelling and creation of a framework of relevant concepts and the semantic relations between the concepts. Semantic data is data that is stored based on the Resource Description Framework (RDF), which is therefore part of the Semantic Web. The Web Ontology Language (OWL) is an extension of RDF, which allows to explicitly represent the meaning of terms in vocabularies and the relationships between those terms, hereby further extending the possibilities of the Semantic Web. Data that is stored using the RDF and an OWL can be queried for by using the SparQL query language. This language is interoperable with the programming language

Python, through a Python module. The tool that is used in the proposed method is therefore constructed using Python as programming language and SparQL as query language.

Three use case studies are performed in order to identify to method requirements. These requirements are used to create a user path for the auditor, which needs to be automated to suit the proposed method. The three use case scenarios also provide insight in the potential benefits of the proposed method. These benefits include: Speeding up the process of the audits and reducing the labour intensity, making the process more accessible for new (BIM) users, standardising the process in such a way that errors are less likely to occur, and the method allows users to have a better understanding of what data the user is assessing via a 3D GML viewer, hereby reducing risks of selection errors.

The proposed method for audits is applicable in the current SCB processes at Rijkswaterstaat due to the fact that it allows an auditor to perform the audit while hereby attaining multiple new benefits when compared to the current method. The current version of the proposed method is considered to be better than the current method as a direct result of the new benefits, ease of operation and user friendliness. In interviews at Rijkswaterstaat is concluded that the proposed method is better than the current method and that it is applicable in the current processes at Rijkswaterstaat as well as applicable in future developments of processes at Rijkswaterstaat.

In this research it has been proven that audit processes at Rijkswaterstaat can be improved by extending the implementation of BIM. Rijkswaterstaat can use this research to further promote the implantation of BIM at Rijkswaterstaat and its contractors. The results of this research can also be used as a prototype for similar problems.

Several limitations for the proposed method and have been identified in the research and design process. These limitations include the not having the functionality to generate geometric data, not having the functionality to compare the geometric data of two objects, and the technical difficulties currently found in code checking compliance, due to the current nature of requirement set which allow for a high level of human interpretation.

It is concluded in this research that it is possible and profitable to perform audits through automated processes. However, the solution provided is specifically designed for a single type of audit. This limitation is a result of the great variety in available types of audit and how these audits are communicated. In this research, a need is identified for a focus of future research for the development of a standard for the automation of audits. The development of this standard would allow audits to be interoperable with BIM systems, which would greatly reduce the required time for the audits and make the audits less prone to errors. This standard could be developed in the form of an ontology specifically designed for audits, or a standard file extension that is specifically designed for audits.

Samenvatting

De bouwsector kan worden beschouwd als een omgeving met een hoog samenwerkingsverband tussen meerdere disciplines. Informatiestromen binnen de bouwsector worden gekenmerkt door interdisciplinaire iteratieve data uitwisselingen en communicatie. Voortdurende inspanning om deze uitwisseling van gegevens en vormen van communicatie te optimaliseren wordt geleverd in vorm van de ontwikkeling van nieuwe tools. Met behulp van de ontwikkeling van deze tools wordt een onderling verbonden omgeving gecreëerd in de vorm van Bouw Informatie Modellen (BIM). Om te profiteren van de voordelen die BIM biedt, wordt BIM op dit moment steeds meer geïmplementeerd binnen Rijkswaterstaat. BIM is op dit moment nog niet volledig geïntegreerd in de processen van Rijkswaterstaat. Onder deze processen bevinden zich de verificatie en validatie processen van Rijkswaterstaat. Deze verificatie en validatie processen worden bij Rijkswaterstaat beheerd volgens de systematiek Systeemgerichte Contractbeheersing (SCB).

Het werken volgens de SCB systematiek in Design-, Build-, Finance- and Maintain (DBFM) projecten bij Rijkswaterstaat houdt in dat de verantwoordelijkheid van het managen van de kwaliteit van de systemen, producten en processen bij de aannemer ligt. De aannemer dient daarom zelf de verificatie en validatie van zijn producten, systemen en processen uit te voeren. Om de aannemer te kunnen controleren is de aannemer verplicht om bewijs van deze verificatie en validatie processen aan Rijkswaterstaat te leveren. Dit bewijs wordt geleverd in volgens voorgeschreven BIM formaat genaamd COINS container. Een COINS container biedt de mogelijkheid om object gerelateerde informatie en bijbehorend documenten van een BIM te communiceren tussen verschillende gebruikers en bijdragers van BIM data.

De toetsen in het SCB proces worden uitgevoerd door auditors die werkzaam zijn bij Rijkswaterstaat. Op dit moment bestaat er geen geautomatiseerde tool bij Rijkswaterstaat die een auditor in staat stelt om geautomatiseerde verificatie en validatie toetsen uit te voeren op BIM data in COINS containers. Het is daarom nu nog noodzakelijk voor een auditor om deze toetsen handmatig uit te voeren. Het handmatig uitvoeren van deze toetsen vereist een uitgebreide kennis van de structuur van COINS containers om data te kunnen uitlezen. Het doel van dit onderzoek is om de SCB processen bij Rijkswaterstaat te optimaliseren, door deze verificatie en validatie processen te automatiseren om een diepere implementatie van het werken met BIM bij Rijkswaterstaat te bewerkstellen. Hiermee zullen meer voordelen van uit het werken met BIM worden behaald en wordt het SCB proces verder geoptimaliseerd. In dit onderzoek wordt een nieuwe methode voorgesteld om verificatie en validatie binnen SCB processen uit te voeren. In deze methode wordt gebruik gemaakt van een script om de data op geautomatiseerde wijze uit de COINS container te verkrijgen, om deze vervolgens te toetsen op de vastgestelde eisen. Om de gebruiksvriendelijkheid van het script te vergroten is een grafische user interface gecreëerd waardoor de auditor de toets kan uitvoeren zonder vereiste technische kennis van het functioneren van het script of COINS containers.

COINS containers bevatten data die is opgeslagen volgens de principes van het Semantische Web. Het Semantische Web is een kennisstructuur met als doel om informatie te delen en te vertegenwoordigen door middel van het modeleren en creëren van een kader van relevante concepten en semantische relaties tussen deze concepten. Semantische data is data die is verreikt met semantische relaties en is opgeslagen op basis van het Resource Description Framework (RDF). De Web Ontology Language (OWL) is een uitbreiding van het RDF die de

mogelijkheid creëert om de expliciete betekenis van termen in de talen voor te stellen en de relaties tussen deze termen vast te leggen. Hierbij vergroot OWL de mogelijkheden van het Semantische web. Data die is opgeslagen volgens het RDF of OWL kan worden opgevraagd met behulp van de SparQL query-taal. Deze taal is interoperabel met de programmeertaal Python via een Python module. De tool die is gecreëerd voor de voorgestelde methode is daarom opgebouwd uit de programmeertaal Python en de SparQL query-taal.

Drie use case scenario's zijn uitgevoerd om de vereisten voor de voorgestelde methode te identificeren. Deze vereisten worden gebruikt om het gebruikers proces, dat de auditor doorloopt, te definiëren. Uit dit proces blijken de stappen die dienen te worden geautomatiseerd. De drie scenario's bieden ook inzicht in de potentiele voordelen van de voorgestelde methode. Deze voordelen zijn: het versnellen van het proces door het verminderen van de benodigde arbeid, het proces wordt toegankelijker voor nieuwe (BIM) gebruikers en het proces wordt gestandaardiseerd en geautomatiseerd waardoor (lees)fouten worden voorkomen.

Vanwege het feit dat de voorgestelde methode de auditor in staat stelt om de toets uit te voeren op een manier die meer voordelen biedt dan de huidige methode, is deze methode van toetsen toepasbaar in het huidige SCB proces bij Rijkswaterstaat. Het wordt aangenomen dat de voorgestelde geautomatiseerde methode beter is dan de huidige handmatige methode, vanwege de voordelen, gemak van uitvoering en gebruiksvriendelijkheid van de tool. Dit wordt bevestigd in conclusies die zijn getrokken uit interviews, die zijn afgenomen bij het personeel van Rijkswaterstaat. Uit de interviews blijkt ook dat de voorgestelde methode toepasbaar is in toekomstige SCB processen die op dit moment worden ontwikkeld door Rijkswaterstaat.

In dit onderzoek is bewezen dat de toets processen bij Rijkswaterstaat verbeterd kunnen worden door middel van een diepere implementatie van het werken met BIM. Rijkswaterstaat kan dit onderzoek en de onderzoeksresultaten gebruiken om de verdere implementatie van BIM te promoten bij Rijkswaterstaat en gerelateerde aannemers. De resultaten van dit onderzoek kunnen ook als prototype worden gebruikt voor vergelijkbare problemen.

Tijdens de onderzoeksfase en de ontwerpfase zijn verschillende beperkingen geïdentificeerd van van de voorgestelde methode en tool. Deze beperkingen bestaan uit: het niet kunnen faciliteren om geometrische data te genereren ten behoeve van geometrische berekeningen, niet de functionaliteit bieden om geometrische data van twee verschillende objecten te vergelijken en de technische beperkingen die zijn geïdentificeerd in het vertalen van bepaalde eisen waarbij een hoge mate van vrije interpretatie mogelijk is. De huidige versie van de voorgestelde methode en bijbehorende tool is ontworpen voor een specifieke toets en is dus niet gereed voor verschillende soorten toetsen. De uitkomst van dit onderzoek kan wel worden gebruikt als prototype om meerdere soorten toetsen op eenzelfde manier uit te voeren. Deze beperking is een direct resultaat van de grote diversiteit in bestaande toetsen en de manieren waarop de data die gerelateerd is aan deze toetsen wordt gecommuniceerd. Tijdens het onderzoek is een vraag geïdentificeerd naar een focus voor toekomstig onderzoek naar de ontwikkeling van een standaard voor het automatiseren van audits. De ontwikkeling van deze standaard zal de interoperabiliteit tussen toetsen en een BIM vergroten en hiermee de benodigde tijd voor een toets reduceren en minder gevoelig maken voor menselijke fouten. Deze standaard kan worden ontworpen in de vorm van een, voor toetsen ontwikkelde, ontologie of een nieuwe extensie voor een standaard bestandssoort voor toetsen.

Abstract

Working with Building Information Models (BIM) is more and more becoming the new standard for data storage and management within the Architecture, Engineering, Construction, and Facility Management (AEC/FM) industry. Rijkswaterstaat therefor constantly strives to further implement BIM in its processes. Rijkswaterstaat uses Systematic Contract Control (SCB) processes in order to perform audits on data which contractors deliver in form of a BIM in a COINS container. These audits are currently performed manually by auditors at Rijkswaterstaat. In this thesis research, a new automated method of performing audits at Rijkswaterstaat is designed in order to research and promote the further implementation of BIM at Rijkswaterstaat and identify and research the benefits and limitations of the proposed method. The proposed method includes the use of a script with a user friendly graphical user interface to allow auditors at Rijkswaterstaat to semi-automatically extract data from OWL files that are stored in COINS containers and to test the data on requirements set by Rijkswaterstaat. This script is constructed using the programming language Python. The script also utilizes the query language SparQL in order to extract data from OWL files which is tested on requirements set by Rijkswaterstaat. This research includes the design and analysis of three use case scenarios that are designed and researched in order to form the theoretical foundation of the proposed method. The script is designed based on the outcome of the analysis of these three use case scenarios. The current SCB process is redesigned in order to allow the auditor to perform audits at Rijkswaterstaat while using the proposed method. The benefits and limitations of the proposed method are identified, hereby further identifying the benefits that the implementation of BIM at Rijkswaterstaat offers. Conclusions on the proposed method are formed and recommendations are presented for future development and research on the further implementation of BIM in audit processes at Rijkswaterstaat and similar companies in the AEC/FM industry.

Contents

PART A: Problem definition and research methodology

1. Introduction

This report documents the graduation research which is conducted as part of the graduation of the master Construction Management and Engineering at Eindhoven University of Technology. The research in this document focusses on an implementation of building information modelling (BIM) at Rijkswaterstaat. This chapter presents the introduction to the problem, the motivation on why and how to solve this problem using the suggested approach, and the description of the thesis using the research questions and the research design. The expected results of the research are presented in this chapter as well in order to conclude the introduction.

1.1. Improving information management with BIM at Rijkswaterstaat

The architecture, engineering, construction (AEC) industry can be regarded as a highly collaboration environment with multiple disciplines that require repeated interdisciplinary iterative data exchanges and communications (Jakob Beetz, Zhang, & Weise, 2015). Continuous effort is being made in the development of new tools to create an interconnected environment in form of BIM in order to facilitate these data exchanges and types of communication. (Yalcinkaya & Singh, 2015). This trend can be deducted from both the available infield applications to from the rapidly increasing number of BIM related publications.

Working with BIM implies the use of a single system of computer models, rather than traditional approaches which use sets of separate drawings or separate 2D or 3D models. BIM facilitates a more integrated design and construction process that results in better quality buildings and lower costs and reduced project duration (Eastman, 2011). The integrated design also allows for better predications of construction performance, more accurate cost estimates, deeper insight in risks and effects, and creates the ability to visualize consequences of design modifications.

Currently, BIM is more and more being implemented at Rijkswaterstaat in order to take advantage of the benefits which BIM offers (Rijkswaterstaat, 2016). Rijkswaterstaat is part of the Dutch Ministry of Infrastructure and the environment. Founded in 1798, it has the role of constructing and maintenance of the public waterways and roads within The Netherlands. Rijkswaterstaat prescribes the use of BIM in all new Design-, Build-, Finance- and Maintain (DBFM) contracts. This implies that both Rijkswaterstaat, as well the contractors, are required to work with the standards of BIM.

Although Rijkswaterstaat uses BIM to store and exchange construction information between Rijkswaterstaat and its clients, not yet all tools and processes that facilitate an optimal use of working with BIM have been designed and implemented. BIM is therefore considered to be not yet fully implemented within all processes at Rijkswaterstaat. The verification and validation processes at Rijkswaterstaat are amongst the processes in which BIM is not yet fully implemented. Rijkswaterstaat uses the SCB processes as the verification and validation processes (Rijkswaterstaat, 2011). SCB is a Dutch abbreviation of ‘Systeemgerichte contract beheersing’, which translates into ‘System oriented contract control’. Working according to the principles of the SCB processes in a Design-, Build-, Finance- and Maintain (DBFM)

construction project implicates that the responsibility of managing the projects quality control system lie with the contractor. Therefore, a contractor has to verify his own products and processes. In order to check the contractors' work, the contractor is required to deliver proof of the validation and verification of their products and processes to Rijkswaterstaat in a standard prescribed BIM format. The BIM format prescribed by Rijkswaterstaat are COINS containers. COINS containers are a means of communication for object related content and associated documents between various users and contributors of BIM data (COINS-projectgroep, 2015). In order to validate the works of the contractor, an auditor at Rijkswaterstaat has to perform sample verification tests on the data in a COINS container based on the requirements which are set in the contract. These tests are planned on beforehand, based on risk- and impact analysis.

1.2. Problem definition

At present, no automated tool exists at Rijkswaterstaat which allows the auditor to perform planned verification and validation tests of COINS containers. Therefore, the auditor has to perform the tests manually and is required to have an extensive knowledge on the structure and classification of data in COINS containers in order to be able to extract the required data from the container and verify and validate the data based on requirements stored in another database. During this manual process, various tools and know-how on COINS containers are required in order to be able to extract the various types of data from the COINS container.

Due to the fact that the overall process mainly works with BIM in the form of COINS containers, BIM can further be implemented in this process by researching and developing appropriate tools and corresponding methods in order to fully identify and obtain the benefits which the working with BIM offers for Rijkswaterstaat. This next step in the implementation of BIM at Rijkswaterstaat consists of adapting all processes to be able to fully profit from the benefits of BIM. A gap exist in the knowledge on what the possibilities are for the verification and validation process and what the benefits and limitations are of various methods of performing automated audits. The partial implementation of working with BIM at Rijkswaterstaat results in opportunities for optimizing processes. The exact nature of these opportunities for benefits are not exactly known and are therefore open for research.

1.3. Motivation

Rijkswaterstaat strives for the continuation of the implementation of BIM within the company and related contractors in order to fully profit from the benefits which the method of data management and communication offers. By researching the possibilities of the optimization of the implementation of BIM within specific processes, Rijkswaterstaat hopes to further promote the implementation of BIM within its own departments.

The redesign of the SCB process at Rijkswaterstaat creates a deeper level of implementation of working with BIM, which consequently creates an increase in acquired benefits which the working with BIM offers. This creates an opportunity for Rijkswaterstaat to use this research as a showcase of the possible opportunities that the further implementation of BIM at Rijkswaterstaat offers, allowing Rijkswaterstaat to further promote BIM-related research for design and implementation of BIM related tools and processes at Rijkswaterstaat.

1.4. Research questions

Three research questions are presented below in order to further define the research problem that is stated above. The process of answering these research questions guides the research into attaining a thorough insights in the problems and the proposed solutions for the problems. The process of answering these questions will also guide the extended analysis of the proposed method of approaching the problem in order to improve the current process. The research questions have been stated at the beginning of the research and were supplemented with additional questions which arose during the research. These research questions which are stated in this thesis are:

- Will the implementation of BIM improve existing processes in DBFM projects for contractors and clients such as Rijkswaterstaat?
- What are the benefits of using an automated tools in order to perform verification and validation of BIM?
- What are the limitations of automated checking using procedural programming approaches in order to perform verification and validation of data files?
- What should be the focus of further research to improve the working with BIM in verification and validation processes?

1.5. Research design

The research is divided in twelve distinct stages in order to acquire the results for the thesis. These twelve stages are listed below in chronicle order with a short explanation of the logic of why the stage is included in the research design. The twelve completed phases were distinguishable as the:

1. **Literary research** in order to acquire insight in the existing state of the art and related conclusions formed in prior research. This literary research focused on books, scientific articles, and research documentations of prior thesis on the subject of BIM.
2. **Interviews at Rijkswaterstaat** in order to acquire insight in current processes at Rijkswaterstaat and the corresponding level of implementation of BIM at Rijkswaterstaat.
3. **Analysis of internal documents at Rijkswaterstaat** in order to further deepen the insight of the current processes and implementation of BIM at Rijkswaterstaat.
4. **Research for tools and processes currently used at Rijkswaterstaat** in order to gain more insight on the procedures for individual actors at Rijkswaterstaat and how these actors use the existing and implemented set of tools.
5. **Design of use case scenarios** in order to identify the requirements for the proposed method with scrutiny when used by a user at Rijkswaterstaat.
6. **Design of the prototype method** in order to attain insight in the limitations and benefits of the proposed method of using a script with a user-friendly graphical user interface (GUI) for verification and validation of data by creating opportunity for a thorough analysis of this design process.
7. **Adaptation of the current processes at Rijkswaterstaat** in order to facilitate the implementation of the newly proposed method of verification and validation at Rijkswaterstaat and to prove the feasibility of the tool.
8. **Feedback from auditors at Rijkswaterstaat** on the newly proposed method in order to gain a deeper insight in the benefits which the method carries.

9. **Formulate conclusion** in order to form explicit answers on the main research questions which are set beforehand.
10. **Formulate discussion** in order to summarize the questions unanswered or opened in the research for this thesis.
11. **Formulate recommendations for future work** in order to spark and kick-start further input in this field by suggesting and promoting follow-up research topics, based on the previously formulated discussion.

A time restriction of 6 months was set for this research. Prior to the start of the research the twelve phases were planned out in a strict planning in order to manage this time restriction.

1.6. Expected results

Several results were expected at the start of this research. These vary from conclusions on the prior stated research questions to the design and construction of a useable script. The main expected results are the following:

- Results of the research on main research question: "*How can BIM further improve the current information flows of verification and validation processes?"* The answers will provide Rijkswaterstaat and the reader of this report with in-depth insight in how BIM can assist in the verification and validation processes between client and contractor when using BIM as means of communicating data.
- A tool which allows the user to verify and validate data on requirements, in contrary to having to manually perform the assessment. A fully functional script will be written which allows the user to verify and validate data while working according to the BIM principles of communicating data.
- Insight on the benefits and limitations of coding languages, such as Python, when writing a script in order to perform validation and verification audits. This will include an extend description on what the limitations and benefits are, and why these limitations exist. Approaches to bypass these limitations are included in the analysis as well.
- An adaptation of the current verification and validation processes in order to benefit from the improvements which the use of the method offers. This will create further insight in the improvements of the process and therefore the benefits which the script will offer within the overall construction process. This will also lower the threshold for Rijkswaterstaat to include the output of this research within their company.
- Insight in more possible research and development topics for the optimisation of

1.7. Readers guide

The research report contains 8 chapters, which are partitioned into four parts. These parts are Part A, B, C and D. Part A introduces the reader to the problem which is to be tackled within this thesis. The methodology used to approach, identify and solve the problem is also presented in this part of the report, as well as a glossary. Part B contains the results of the desk research. This consists of literary research on the subject, as well as the conclusions from interviews and research on documentation at Rijkswaterstaat. Part C consist of the design and development of a script and method. This model and method are reviewed in part D in order to form conclusions on the research.

PART B: Desk Research

2. Building Information Modelling

In this chapter, the concept of working with Building Information Modelling (BIM) is presented. First, a general introduction to BIM is presented in paragraph 3.1. The benefits of BIM are presented in paragraph 3.2. Thereafter, disadvantages of BIM are presented in chapter 3.3. Current BIM-related research topics are listed in paragraph 3.4. The topic code compliance checking is discussed in 3.5. Conclusions on the literary research for BIM are presented in paragraph 3.6.

2.1. Introduction to Building Information Modelling (BIM)

The AEC industry is constantly evolving due to the introduction of new building technologies. More and more information is digitized and information management is therefore becoming increasingly complex. One of the most promising developments in the AEC industry is Building Information Modelling (BIM) (Eastman, 2011).

BIM can be regarded as a method of building information management and communication that implies the use of a digital database model in which the information is stored in order to facilitate interoperability and exchange of information with related software applications (Miettinen & Paavola, 2014). The construction industry has confirmed BIM as a method that improves the efficiency and effectiveness of delivering a project from inception to operation and maintenance (Ding, Zhou, & Akinci, 2014). The use of BIM can therefore be regarded as the latest chapter of the evolution of information management systems within the AEC industry, as is visualized in figure 1 (Bew & Richards, 2008).

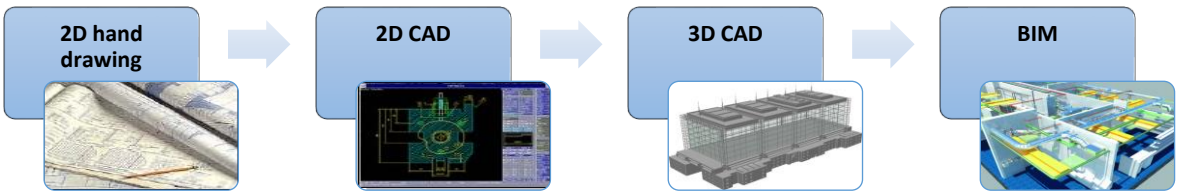


Figure 1. The chapters of information management in the AEC industry.

There are many definitions of 'BIM'. It can both regarded as a verb, meaning 'working with building information models' and as a noun 'a building information model' (Lu, Fung, Peng, Liang, & Rowlinson, 2014). A commonly used definition of BIM is the one used by the US National BIM Standards (NBIMS), who define a building information model as:

> *'A Building Information Model is a digital representation of physical and functional characteristics of a facility. As such it serves as a shared knowledge resource for information about a facility forming a reliable basis for decisions during its lifecycle Fig. 1. Variety management strategies at the product level and at the process level from inception onward. A basic premise of BIM is collaboration by different stakeholders at different phases of the life cycle of a facility to insert, extract, update or modify information in the BIM to support and reflect the roles of that stakeholder. The BIM is a shared digital representation founded on open standards for interoperability'* (National Institute of Building Science, 2007).

2.2. Benefits of BIM

As elaborated in the BIM Handbook, working with BIM allows the actors in a construction process to solve problems that arise when using traditional 2D paper-based construction methods and the corresponding predominant business in the construction industry (Eastman, 2011). These traditional paper-based modes of communication are prone to errors and omissions in paper documents that lead to conflicts, which in term lead to delays and failure costs. When working with BIM, information is not stored separately in 2D or 3D drawings and documents, but it is interrelated in one Building Information Model. This leads to two main advantages which working with BIM realizes. Firstly, version management is easier achievable since all information is interlinked in the same centralized repository, leading to more data integrity, which mitigates the risk of poor communication for project managers (Rokooei, 2015). Secondly, since all information is interlinked in the same centralized repository, new levels of spatial visualization, building simulation and more efficient building and process management are possible (Miettinen & Paavola, 2014). This, in turn, leads a higher design consistency, better design visualisation, more accurate cost estimations, extended possibilities of clash detection, and improved stakeholder collaboration (Volk, Stengel, & Schultmann, 2014).

2.3. Disadvantages of BIM

Although BIM brings many advantages, it also brings disadvantages (Eadie, Browne, Odeyinka, McKeown, & McNiff, 2013). The main disadvantage of BIM arises from the fact that the implementation of a new information management system is not a simple task. Processes and tools have to be redesigned for all actors of the process. It can therefore not be implemented as a single software tool. This results in a barrier for contractors and clients to implement BIM in projects. This barrier consist of the required time and costs involved with the acquisition of resources and training of personnel. Due to the relatively new nature of BIM, a knowledge gap exists; there is no standard BIM design for each project. This means that when a company wants to implementing BIM, the company has to conduct research in order to identify or create the optimal resources, tools and processes.

2.4. Current BIM-related research topics

BIM related research topics are presented here in order to create a general understanding of the state-of-the-art of BIM and current hot topics in BIM related research. There are numerous examples of infield BIM applications and publications of BIM related research (Farr, Piroozfar, & Robinson, 2014; Yalcinkaya & Singh, 2015). Volk, Strengel and Schultmann identified the main problems which BIM-related research currently encounters (Volk et al., 2014). These main research topics are functional issues, organizational and legal issues, information interoperability issues, and technical issues.

Volk et al. identified in that the functional issues arise due to the fact that the BIM toolset is required to have an extend level of functionalities to facilitate the design, engineering, construction, and maintenance of a project. These functionalities are connected to the 3D, 4D or 5D BIM (e.g. quantity take-off, scheduling or cost calculation) or they are attached to BIM as independent expert applications. Volk et al. recognize that currently, most research focusses on expert functionalities (e.g. carbon reduction analyses, construction progress tracking). These are a result of the architects, engineers and constructors, who played a major role as the early adopters of BIM. Implementing BIM in existing constructions is however not

yet as developed as BIM in the construction and engineering phases of a construction project. However, according to Volk et al., research related to functionalities of BIM for existing constructions is intensifying.

Organisational and legal issues are identified in the topics of collaboration of stakeholders and the responsibility, liability and model ownership by Volk et al. Legal frameworks related to BIM differ from country to country. The Netherlands does not provide an optimal conclusive jurisdictional framework (Koot, 2012). According to Koot, current BIM related contracts and general terms prove to still be too abstract when regarding exchange and reliability of information and coordination. The design risks are too hard to identify when using BIM in order to use existing Dutch contracts.

Information Delivery Manual (IDM) frameworks, Model View Definitions (MVD) provide relative information for expert functionalities that are linked with BIM which facilitate data exchange and avoid ambiguities (Venugopal, Eastman, Sacks, & Teizer, 2012). According to the literary research by Volk et al., the IDM and MVD have been well developed for construction and engineering processes. However, research for IDM and MVD applications related to BIM for existing constructions has not been a hot topic of recent BIM related research. IFC is identified as one of the most popular construction industry standards in which many BIM models can be converted to and exported to other systems for various applications (Abanda, Tah, & Keivani, 2013).

The main BIM-related technical research topics are identified by Volk et al. as the automation of data capture and BIM creation without pre-existing BIM, the update and maintenance of information in BIM and the handling and modelling of uncertain data, objects and relations in existing buildings in BIM. New techniques try to overcome lacking building information at low costs, but face the challenges of capturing structural, concealed or semantic building information under changing environmental conditions and transforming this data into unambiguous semantic BIM objects and relationships.

2.5. Code Checking Compliance

Errors in manual code checking compliance occur due to the fact that inspectors may base their quality decision-making on experience, leading to time-consuming and error-prone manual processes (Zhong et al., 2012). Zhong et al. note that the required strictness in regulation-based construction quality compliance checking for, which is required for automated checking processes, would reduce quality inspection errors, consequently improve quality compliance and reduce violations to the regulations that govern the construction process.

BIM based clash detection allow automatic geometry-based clash detection to be combined with semantic and rule-based clash analysis in order to perform selective clashes check between specified systems (Eastman, 2011). According to Eastman, several problems must be overcome in order to realize automated code checking compliance with realistic expectations. First of all, the building information model must be modelled with an appropriate level of detail, so that clashes can be accurately detected. Furthermore, if the geometric data are not solids, clash detection tools cannot detect clashes between two objects. Another problem according to Eastman is that qualification of clashes into meaningful categories for the contractor is greatly inhibited due to a lack of available semantic information.

Within the world of BIM, more and more research is being conducted on automated code checking compliance due to its benefits (Tan, Hammad, & Fazio, 2010). However, Tan et al. acknowledge the lack of knowledge in the general automatization of code compliance checking. They state that, although continues efforts have been made to improve automated code compliance checking, most of the research aims at only one specific domain such as construction design, fire safety, or accessibility. Tan et al. attribute this fact to be a result of the high number of possible alternatives and performance attributes checks in the construction industry.

2.6. Conclusions on BIM

As is concluded in this chapter, BIM can be regarded as the biggest research topic in AEC/FM related processes. Due to the benefits which the interoperability of working with BIM is more and more being integrated into the construction sector and numerous applications of BIM already exist. However a high demand for BIM related research still exists due to functional issues, organizational and legal issues, information interoperability issues, and technical issues. The problematics of code compliance checking have been researched as well and a lack of general automatization of code compliance has been identified.

3. Semantic Web

In order to create a proper introduction to the Semantic Web, this chapter first introduces the Semantic Web in paragraph 1. Semantic data is data that is stored based on the Resource Description Framework (RDF), which is therefore part of the Semantic Web. (Wood, Gearon, & Adams, 2005). The RDF is thus presented in paragraph 2. The Web Ontology Language (OWL) allows to explicitly represent the meaning of terms in vocabularies and the relationships between those terms, further extending the possibilities of the Semantic Web (Harmelen & McGuinness, 2004; World Wide Web Consortium, 2009). An introduction to OWL is therefore presented in paragraph 3. Thereafter, the query language SparQL and programming language Python are respectively presented in paragraph 4 and 5. Lastly, a conclusion on the literary research on the topic 'Semantic Web' is presented.

3.1. Introduction to the Semantic Web

The Semantic Web is a knowledge structure used to formally represent and share information through the modelling and creation of a framework of relevant concepts and the semantic relations between the concepts (Abanda et al., 2013). The use of Semantic Web technologies represent methods of formatting data based on the meaning of the data, rather than on the structure of the data (Berners-Lee, Hendler, & Lassila, 2001; Shadbolt, Berners-Lee, & Hall, 2006). The key strength of the semantic web is that it allows computer systems to conduct automatic reasoning, hereby better enabling computers (and people) to work in cooperation than more traditional knowledge representation structures. This allows for a more uncomplicated linking of data structures.

The Semantic Web must include collections of information and sets of inference rules in order to facilitate the automatic reasoning process (Berners-Lee et al., 2001). Inference rules are the rules on which new links between data can automatically be created. These inferences rules of the Semantic Web are not centralized for each data base, but are decentralized so that databases can be interoperable. This interoperability of data and inference rules creates opportunities for reasoning through standard languages. This facilitates the possible reasoning of new relationships which could be added to a data set or returned by a query.

The Semantic Web creates possibilities for the support of large scale information sharing in the Architecture, Engineering, Construction, and Facility Management (AEC/FM) industry (J Beetz, 2009a, 2009b). This is a direct results of the information management in the AEC/FM industry, which by nature is decentralized and multidisciplinary (Karan & Irizarry, 2015). These possibilities of large scale information sharing allow the Semantic Web to offer a solution to one of the main obstacles of BIM implementation, which is the interoperability between BIM systems (Volk et al., 2014).

Semantic languages have been developed to create greater opportunities for encoding, leading to that the support for information integration and interoperability (Shadbolt et al., 2006). Commonly used linked data technologies are used as underlying technologies of the Semantic Web. These technologies are presented in figure 2 in the stack of technologies which makes up the Semantic Web according to Tim Berners-Lee. The main layers of the Semantic

Web are elaborated on in the following paragraphs in order to describe the workings of the Semantic Web.

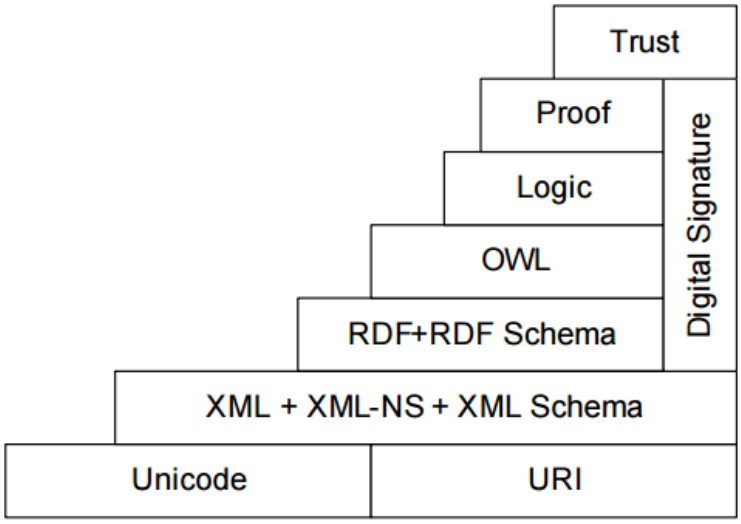


Figure 2. The stack of underlying technologies of the Semantic Web according to Tim Berners-Lee.

3.2. RDF: Research Description Framework

The RDF type of data storage can be regarded as a basic model of the Semantic Web (Allemang & Hendler, 2011; Bonabeau, 2002; Yu, 2014). RDF stands for Resource Description Framework and is a methodology for data modelling (Lassila & Swick, 1998). It provides a standard foundation for a distributed network of data (Allemang & Hendler, 2011; Yu, 2014). The RDF is an enabling technology that is recommended by the World Wide Web Consortium (W3C) (Klyne & Carroll, 2006).

Data that is stored based on the RDF is represented by triples. These triples consist of a subject, predicate, and an object. The predicate can be regarded as the type of relation between the object and the subject. One can replace the word 'predicate' with 'type of relation'. Large storages of linked data can be created by linking the subjects and objects semantically through these predicates. This semantically enrichment of data allows machines to automatically process and integrate available information. A more elaborate explanation on the construction of triples is included in Appendix 1.

When visualizing multiple triples that are stored in a data in a RDF, a graph will emerge as is presented in figure 3. A RDF database wherein all data is linked is therefore referred to as a graph. When merging multiple graphs the essence of the merge comes down to: "When is a node in one graph the same node as a node in another graph?" (Allemang & Hendler, 2011). This problem is solved by giving each node within a graph a Uniform Resource Identifier (URI). A URI can represent a classes, properties or individuals. This is helpful within the Semantic Web because it provides a mechanism to uniquely identify a given resource. It also specifies a uniform way to retrieve machine-readable descriptions about the resource being identified by the URI. The previously mentioned example in figure 3 presents a graph which consist of two merged graphs. The graphs have the same URI for 'England' and therefore England exists as a separate node in both separate graphs and is represented as a single node in the merged

graph. Also the predicates, such as for example 'partOf', have a URI. As example of an URI, the URI for the node 'Shakespeare of the example in figure 3 is:

http://www.workingontologist.org/Examples/Chapter3/Shakespeare#Shakespeare

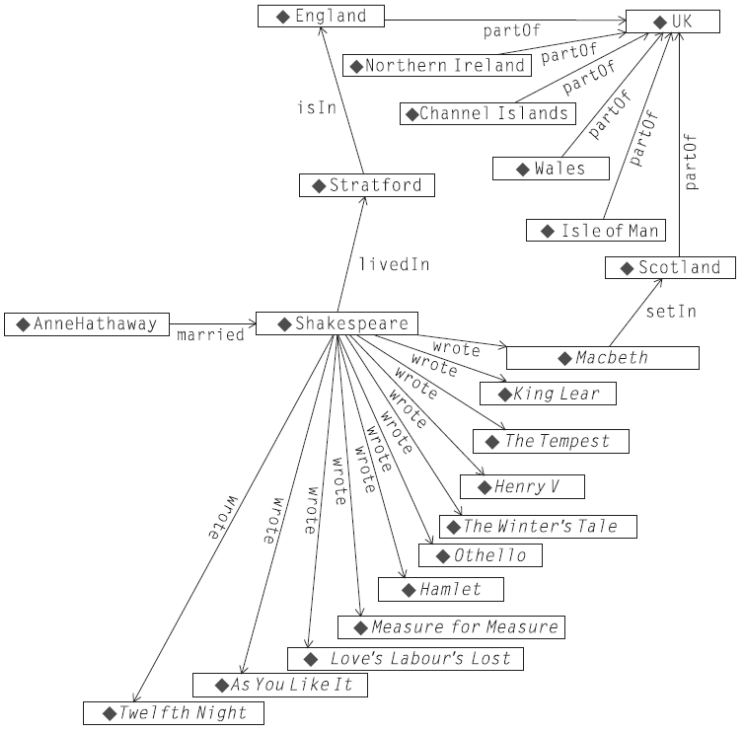


Figure 3 Graphic representation of triples: example of a RDF graph (Allemang and Hendler 2011).

Since URI's have a global scope and are used consistently across contexts, the use of URI's is the main key of what gives the RDF and the Semantic Web its interoperability (Shadbolt et al., 2006).

Various types of RDF serialization exist for the storage of RDF triples (Wood et al., 2005). Examples of these alternatives are N-Triples, Turtle, and RDF/XML (Allemang & Hendler, 2011). The form N-Triples correspond mostly directly to raw RDF triples since refers to resources by using their fully unabbreviated URIs. The Turtle format is a more compact serialization of RDF, which combines the apparent display of triples from N-Triples with the compactness of qnames. These qnames can be regarded as abbreviations for a specific URI. The use of these abbreviations allows turtle to be convenient for human consumption and more compact for a printed page. The RDF/XML format is processable by many Web infrastructures, and is therefore recommended by the W3C (Beckett & McBride, 2004). The RDF/XML format consists of RDF data expressed in XML. XML is a language that is both processable by machines as well as human-legible (Beckett & McBride, 2004).

The Resource Description Framework Schema (RDFS) is the basic language in which the syntax is defined that is used to structure RDF based data (Yu, 2014). RDFS is an extensible knowledge representation language that can be used to create a vocabulary for describing classes, subclasses and properties of RDF resources (Guha & Brickley, 2004). This implies that the use

of RDFS creates the possibility of making statements about classes of subjects and types of relationships. This also implies that RFDS allows the description of the meaning of a relationship or a class in text readable by both humans and machines. The RDFS contains the most basic elements to describe RDF based ontologies (P. Pauwels et al., 2011). The RDFS is also a technology that is recommended by the W3C (Guha & Brickley, 2004).

3.3. OWL: Web Ontology Language

The Web Ontology Language (OWL) is, like RDFS, an extension of the RDF. OWL is an ontology language that offers a greater expressivity in object and relation descriptions by enabling efficient representation of ontologies that are amendable to decision procedures (Shadbolt et al., 2006).

When regarding the topic 'Semantic Web', an ontology formally defines a common set of domain-specific terms that are used to describe and represent a domain. An ontology defines the terms used to describe and represent an area of knowledge (Bechhofer, 2009). The OWL is a language that can be used to create these ontologies. The OWL is standardized by the W3C on February 10 2004 (World Wide Web Consortium, 2004).

W3C formed the following definition of OWL 2: *"The W3C OWL is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be reasoned with by computer programs either to verify the consistency of that knowledge or to make implicit knowledge explicit* (World Wide Web Consortium, 2009)*."*

Alike the RDFS, the purpose of OWL is to define ontologies that include classes, properties, and their relationships for a specific application domain. However, when compared to the RDFS, the OWL provides the capability to express the relationships on a much more complex and richer level. This allows for the creation of ontologies with much stronger and reasoning abilities. OWL can be defined as:

OWL = RDF Schema + 'New constructs for better expressiveness' (Yu, 2014).

Since the OWL is standardized by the W3C in 2004, several updated versions of the standard OWL ontology have been standardized by the W3C. These were OWL 1.1 in 2005, and thereafter OWL 2.0 on October 27 2009 (World Wide Web Consortium, 2009). These new updated versions can be considered as an subset of the previous versions (Yu, 2014). The updated versions of OWL presented a adaptations as solution for issues such as expressivity issues, problems with its syntaxes, and deficiencies in the definition of OWL species (Grau, Horrocks, & Motik, 2008).

3.4. SparQL

SparQL is an abbreviation of Simple Protocol And RDF Query Language (Wood et al., 2005). SparQL is a RDF query language and data access protocol for the Semantic Web (Yu, 2014). The query language can query process data base systems that consist of RDF triples (Curé, 2015; Prud'hommeaux & Seaborne, 2014). Since the query language can be used to

query for data that is stored in RDF, it can also query for data based stored based on OWL (Birte, 2011). The query results can be result sets or RDF graphs. SparQL has four query forms to form result sets or RDF graphs. According to Birte, these four query forms are:

- SELECT, which returns all variables bound in a query patter match
- CONSTRUCT, which returns an RDF graph by substituting variables in a set of triple templates.
- ASK, which returns a Boolean indication whether a query pattern matches or not.
- DESCRIBE, which returns a RDF graph that describes the resources found that are queried.

A SparQL query can query a set of triples like RDF triples, except that each of the subject, predicate, and object may be a variable targeted by the query (Karan & Irizarry, 2015). An example is presented in figure 4. In this example, a selection query is presented which could query for the first name and last name of people in a database who speak English.

```
SELECT ?fn ?ln
WHERE {
   ?x    :language      :English.
   ?x    :firstName     ?fn.
   ?x    :lastName      ?ln.}
```

Figure 4 An example of a SparQL query (Curé, 2015).

The SparQL facilitates various functionalities. The main functionalities are:

- To facilitate the querying of RDF graphs to get specific information.
- To run automated regular queries against RDF datasets to generate reports
- To enable application development at a higher level.

SparQL is used in this research because of these main query capabilities. The ability to query for RDF data allows for data retrieval from RDF based BIM systems. Its interoperability with programming languages, such as the programming language Python, allows the development of an application that can carry out these queries.

3.5. Python

Python is a programming language that can implement the query language SparQL in its code. The programming language Python is not directly linked with the Semantic Web, but it is discussed in this chapter due to its operability with the SparQL language. Characteristics of Python are that the language is object-oriented, has an imperative character, and has a very functional oriented syntax (Remmen et al., 2015). This functional orientation of the syntax makes Python intuitively comprehensible for non-expert programmers (Prechelt, 2000). This allows programmers who do not have excessive programming experience, such as the author of this thesis, to read and create lines of Python code. Due to its functionalities, Python can be used as a command execution and modelling environment (Lampert & Wu, 2015; Schmitz, Karssenberg, de Jong, de Kok, & de Jong, 2013).

Python consist of multiple source files called modules (Zhang, 2015). These modules can be imported and executed within the Python application by the Python language. Modules are files containing python code which can be used to import pre-written Python code that can contain additional Python commands and capabilities (K. D. Lee, 2014). An example of such a module is the Math module, which allows to import the Pi.

Figure 5 shows an example of Python code. In this example a command is given to print a predetermined sentence and an imported instance from the module 'math'. The outcome of this code would be: "Pi with three decimals: 3.142"

```
>>> format = "Pi with three decimals: %.3f"
>>> from math import pi
>>> print format % pi
Pi with three decimals: 3.142
```

Figure 5. An example of Python code (Hetland, 2008).

Python is chosen as main programming language in this thesis because of these main characteristics. A disadvantage is that Python itself cannot directly run queries against Semantic Web database with OWL ontology. Due to its functionality of importing various modules it can however use modules to execute SparQL queries on OWL-files in order to perform these queries (Techentin & Gilbert, 2014).

3.6. IFC: Industry Foundation Class

IFC is an abbreviation of Industry Foundation Classes and in it selves, not a technology that is directly related, or linked, to the Semantic Web. IFC is a type of data storage which provides an environment of interoperability among IFC-compliant software applications in the architecture, engineering, construction, and facilities management industry (Bazjanac & Crawley, 1997). It allows building simulation software to automatically acquire building geometry and other building data from project models created with IFC compatible software. IFC was developed by the International Alliance for Interoperability (K. Lee, Chin, & Kim, 2003). The use of IFC has proven to dramatically reduce the cost and time needed for simulations (Bazjanac & Crawley, 1999).

IFC databases are not based on semantic web data sets like RDF or OWL data, but on the EXPRESS modelling language (Nour & Beucke, 2008). IFC databases can therefore not be directly connected to Semantic Web technology without converting an IFC schema into an OWL ontology, which is considered to be straightforward except for some minor complications (Schevers & Drogemuller, 2005; Torma, 2013). According to Schevers (who currently works at Rijkswaterstaat) and Drogemuller, IFC can benefit of the advantages of Semantic-Web related technologies due to the possibility of converting IFC files into OWL files.

However, various tools are being developed and research is being conducted to the increase of interoperability between the IFC and other BIM technologies. A recent example of a tool to convert IFC files into a Semantic Web language is proposed by Pauwels and Terkaj (Pieter Pauwels & Terkaj, 2016). This project of creating an IFC –to-OWL converter is already freely available on the GitHub repository under the link: https://github.com/mmlab/IFC-to-RDF-converter/. An on-going development of a framework for a domain specific, open query language for BIM exists in the form of BIMQL (Mazairac & Beetz, 2013). BIMQL is an extendable open domain specific query language for BIM which can be used to select and update partial aspects that are stored in an IFC format. The current state of implementation of BIMQL allows the selection of objects and attributes based on their schema names or arbitrary properties stemming from standardized or custom property sets. A prototypical implementation of a model view checker for model instance validation of IFC models is developed as well (Jakob Beetz et al., 2015). This open source IFC model view checker is developed to facilitate automated checking procedures.

3.7. Conclusion

The interoperability between databases which the Semantic Web through semantic data enrichment offers is very beneficial for the AEC/FM industry and therefore key to BIM. For this research, the technologies behind the Semantic Web, regarding the RDF, RDFS, OWL and SparQL are analysed and their potential researched in this chapter. In addition, the programming language Python has been researched and selected as a capable and appropriate command execution language within this thesis due to its operability with the SparQL query language. Also the type of data storage IFC is presented in this chapter due to recent developments in the operability between IFC and Semantic Web Technologies.

4. Systems, tools and processes used at Rijkswaterstaat

Rijkswaterstaat has used simulation models extensively in carrying out its tasks for a long period of time (Vollebregt, Roest, & Lander, 2003). This made the implementation of BIM a logic option at Rijkswaterstaat. This chapter presents the BIM related systems and processes used at Rijkswaterstaat that are of interest for this research. The COINS systematic is presented in paragraph 5.1. The SCB process at Rijkswaterstaat is presented in paragraph 5.2.

4.1. COINS framework

Both from interviews at Rijkswaterstaat as well as from literature research it is determined that Rijkswaterstaat uses COINS containers as form of data exchange between Rijkswaterstaat and other actors in the construction process (Brous, Herder, & Janssen, 2015). Rijkswaterstaat uses COINS to structure and store objects and related data in a BIM container exchange format (Rijkswaterstaat, 2015). The acronym COINS stands for 'Construction Objects and the Integration of processes and systems'. The COINS systematic is presented on a single website and in this research, most of the retrieved information on the COINS systematic is retrieved from this site (COINS-projectgroep, 2015). The development of COINS started in 2003. The COINS framework is developed by multiple companies, consultancies and universities in the Netherlands, who are related to the built environment. These developers are represented in the COINS project group.

COINS is an open BIM-standard which supports the exchange of System Engineering information and facilitates the storage of: an object tree, Geographical Information Systems, 2D-drawing, 3D-models and object type libraries in a single data set, stored in a single container. The focus of the COINS project lies on the content of communication between the participants of the construction process (Nederveen, Beheshti, & Willems, 2010). The content of this communication consist of construction object information. This includes object characteristics as material, planning data, cost data and its 3D geometry.

The system consist of two important components, namely the COINS Building Information Model (CBIM) and the COINS Engineering Method (CEM). The CBIM is the building information model and it contains the structure of objects and their associated data, the CEM describes the methods and procedures that are followed in the engineering process (Nederveen et al., 2010).

In the COINS systematic, the core of the project is the construction (COINS-projectgroep, 2015). This construction can be visualized as a collection of connected objects. The COINS systematic regards all objects that can be defined within the object tree of a construction. This would mean, for example, that a pomp is regarded as an object while a screw within the pomp is also regarded as an object.

All objects and their corresponding information are stored within the COINS container. This COINS container is usually associated with the file extension ".ccr". The container is a zip-file which contains all building information in folders that contain RDF-files, authorisation data and linked documents. A visualization by Rijkswaterstaat of the contents of a COINS container are presented in figure 6. This information is semantically enriched via relationships between objects and object type libraries.

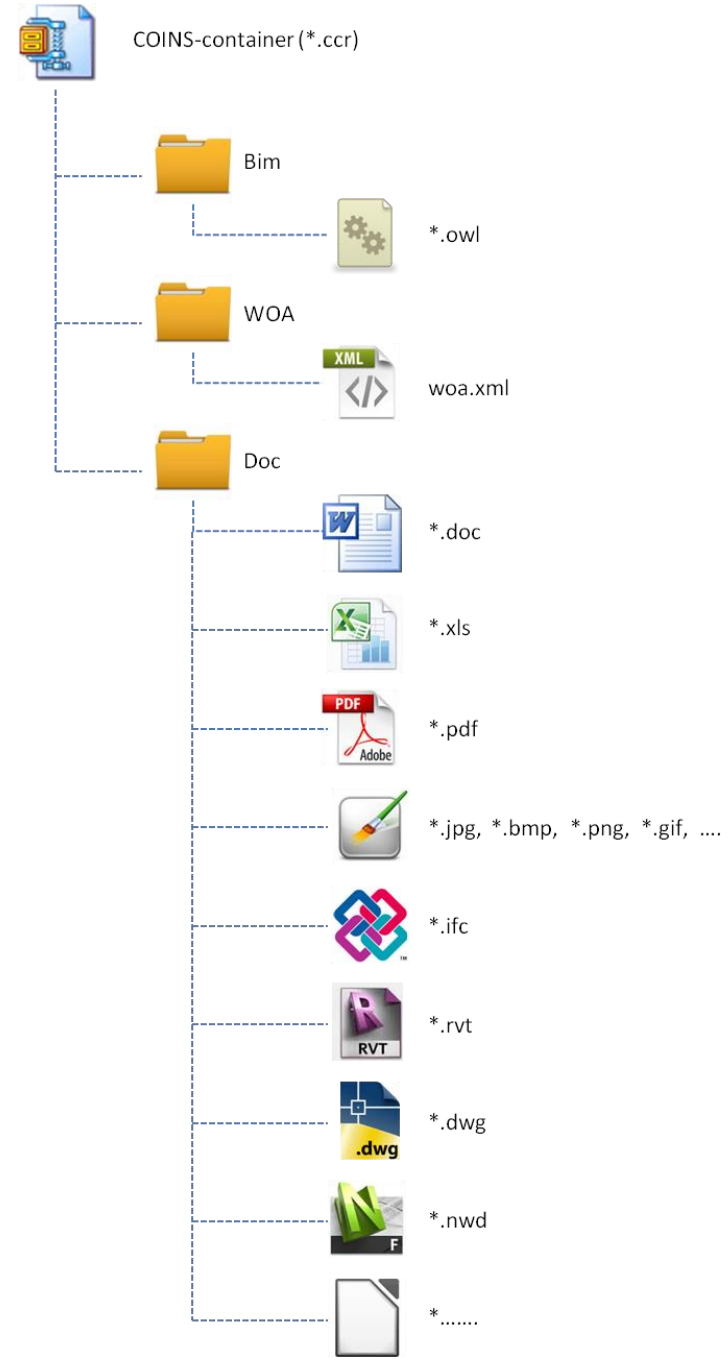


Figure 6. Visualization of the contents of a COINS container by Rijkswaterstaat

4.2. SCB process

Rijkswaterstaat uses SCB processes in order to perform verification and validation on the work of its contractors (Rijkswaterstaat, 2011). SCB is a Dutch abbreviation of 'Systeemgerichte contract beheersing', which translates into 'System oriented contract control'. The company uses the SCB process in all its contract forms, which are D&C, DBFM, E&C, performance contracts, and the framework contract engineering services. The primary goal of the process is to efficiently and effectively control contracts and to preserve legitimate payments. All information on the SCB process presented here is derived from interviews and internal documents at Rijkswaterstaat.

In the SCB process, the responsibility of complying with project requirements lie with the contractor. The contractor manages the project and must prove to comply with project requirements by delivering evidence which is prescribed in the verification and validation planning and corresponding reports. This evidence must suggest that the contractor fully controls its process and therefore that the Deming circle works on his project. The Deming circle exist of three phases (contract, process, product), in which the process consists of a circle that includes 'plan', 'do', 'check', and 'act' (Basu, 2004). The Deming circle is presented in figure 7.

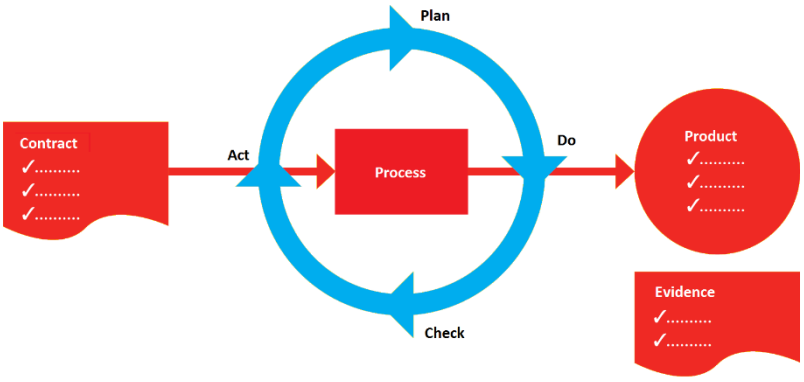


Figure 7. Circle of Deming (Rijkswaterstaat, 2011).

For the delivery of this verification and validation evidence, three main audits are identified in Deming's circle, the system audits, the process audits and the product audits. The system audits aim on auditing the quality management systems that are included in the contract. The process audit is and audit which aims on auditing the contractors' processes. This includes audits such as the effectiveness of the use of the contractors' resources and the contractors' risk management. A product audit is an audit which tests whether the products comply with the demanded requirements and/or technical specifications. This includes audits such as object measurement and comparison with audit rapports or verification and validation data of Rijkswaterstaat. Based on the outcome of these audits, payments are made to the contractor on various moments in the construction process.

The audits are always conducted by an auditor of Rijkswaterstaat, who are certified to perform the audits. The auditor performs the checks that are planned in advance in the SCB process. The auditors and audits are managed by a lead auditor with use of a management tool called the UR-SCB environment, which will be elaborated on in the next paragraph.

These audits are planned in an audit plan on beforehand by an auditor at Rijkswaterstaat. These audits are sample tests, meaning that only occasionally tests are conducted. These sample tests are based on a predetermined risk analysis. Multiple factors are included in the risk calculation, including factors as chance on failure, impact of failure or previous experience with the contractor. The audits will be planned throughout the entire process as can be seen in the graph in figure 8. The audit plan can change throughout the process.

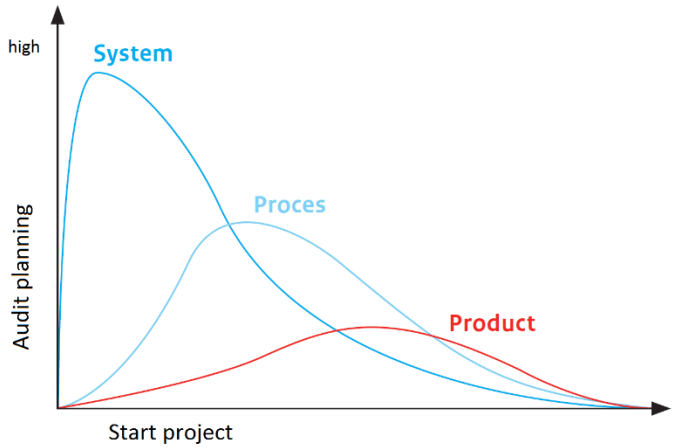


Figure 8. Audits throughout the construction process (Rijkswaterstaat, 2011).

An audit must be clearly defined. The goal of the audit, the scope of the audit, the basis of the audit, and the organisation of the audit must be determined. The scope of the audit refers to the subjects that are audited. The basis of the audit refers to the specific requirement, norm or a contract reference which the audit will test the contractors' compliance. The organisation of the audit refers to the topics as location, actors and required time of the audit.

The outcome of the audit must be defined and recorded according a predetermined process. This recorded outcome of the audit must be communicated to the contractor. The recording and registration of the audit must contain: the reference or basis of the audit, the factual conclusions and the corresponding evidence, and the conclusion 'positive or negative'. Based on the outcome of the audit appropriate consequences will be facilitated via the standard SCB processes. These consequences for a negative audit outcome will result in a discussion for measures. A positive outcome will result in a declaration of positive performance delivery and therefore positively influence the appropriate payment process.

More research on the information flows in the SCB process is conducted and presented in the next part of this thesis: 'Part C: Model Design'. Rijkswaterstaat uses a software environment called 'UR-SCB' in order to facilitate the audit process. The UR-SCB environment is presented in the next paragraph.

4.2.1. UR-SCB

The UR-SCB is the online audit administration environment used by Rijkswaterstaat (Rijkswaterstaat, 2011). All information on the UR-SCB software environment presented here is derived from personal experience with the environment, and internal documents and

interviews at Rijkswaterstaat. The UR-SCB environment presents the auditor with the ability to easily create and manage audits. The online environment is accessible through the Rijkswaterstaat intranet can only be accessed via a password guarded account that is assigned to the appropriate staff members at Rijkswaterstaat. The UR-SCB is a Dutch abbreviation of 'Uniforme Registratie van Systeemgerichte Contract Beheersing', which translates into 'Uniform Registration of System oriented Contract Control'.

The use of the UR-SCB environment when working with construction contracts is the standard projects at Rijkswaterstaat. The functionalities which the UR-SCB environment facilitates include:

- The creation of the audit plan
- The planning of audits
- The updating and managing of audits already performed
- Storage and guarding of the outcome and findings of audits

When performing an audit, the auditor can use the UR-SCB to withdraw all required information, perform the audit, and store the output of the audit. In the current version of the SCB process, the audit itself has to be performed manually by the auditor.

4.3. Conclusion

Systems, tools and processes at Rijkswaterstaat have been identified in this chapter. Rijkswaterstaat uses BIM to manage and control construction processes. In these processes the COINS systematic is used to transfer information between Rijkswaterstaat and contractors and the COINS systematic is used to transfer information internally at Rijkswaterstaat. The SCB process is presented as a process that handles verification and validation of contractors by Rijkswaterstaat, hereby using the UR-SCB tool. The SCB process has been identified as a process in which BIM is more and more being integrated and optimized.

PART C: Model design

5. Determining method requirements

To create more insight in the benefits and limitations of the proposed method of using queries to approach BIM data stored in a Semantic Web in processes at Rijkswaterstaat, the SCB process is analysed in order to identify the required functionalities to which the proposed method must comply. In paragraph 6.1 the SCB process is identified as an appropriate process at Rijkswaterstaat to further implement BIM by implementing the suggested method. Thereafter in paragraph 6.2, three use case scenarios are presented which focus on three common scenarios that present themselves in the SCB process, in order to identify the requirements for the method and hereby the foundation of the design of the method. These requirements are analysed in paragraph 6.3 in order to analyse the requirements en to define the possibilities. The limitations for the proposed method that arise from this analyses are presented in paragraph 6.4. In paragraph 6.5 a possible use of the method is presented. Conclusions on the research that is presented in this chapter is presented in paragraph 6.6.

5.1. Potential for BIM related process improvement at Rijkswaterstaat

From internal literary research and interviews conducted at Rijkswaterstaat it is concluded that Rijkswaterstaat strives to further implement BIM in its processes. Rijkswaterstaat asks external sources to conduct BIM related research on existing processes and to create new, and further develop existing, case studies at Rijkswaterstaat. Rijkswaterstaat requests this in order to further promote the implementation and hereby use of BIM by Rijkswaterstaat and corresponding contractors.

Rijkswaterstaat has created case studies which implement the use of COINS containers as a BIM in order to store and transfer data in SCB processes. The results of a literary research on the science behind these COINS containers is presented in paragraph 5.1 of this report. The results of literary research and interviews at Rijkswaterstaat on the SCB processes is presented in paragraph 5.2 of this report. Currently, in the SCB process, an auditor has to manually perform audits on BIM because no automated processes or method are available. These manual audits require of the auditor that he/she has an extend knowledge on the setup of a COINS container and experience in manually analysing the contents of a COINS container. This makes the performing of audits very labour intensive, as well as time intensive. The labour intensity and relative complexity of the COINS containers makes this process also vulnerable for communication errors due to misinterpretation of data.

Because of the disadvantages which the current SCB process holds, Rijkswaterstaat expressed during internal interviews the need for a new automated method and a corresponding redesigned of the SCB processes at Rijkswaterstaat. The redesign of this currently manual process into an automated process that fully operates with the BIM systematics allows Rijkswaterstaat to showcase the possibilities of BIM and the corresponding benefits within the company.

5.2. Three use case scenarios

Use case scenarios have been designed in order to acquire insight in the steps which the user of the tool needs to take, hereby analysing the system and creating the opportunity to identify the corresponding requirements for the proposed method according to a proven research

method (Rolland & Achour, 1998). A total number of three use case scenarios are designed. Each of these three use case scenarios focus on a different type of audit scenario which an auditor at Rijkswaterstaat could encounter in common audit situations at Rijkswaterstaat. The description of these three scenarios includes all possible outcomes for the each of the specific scenarios in order to cover all aspects of the situation, hereby fully discover all requirements for the tool.

The choice to perform a total number of three use case scenarios is based on the fact that three main types of methods when performing audits can be identified. The choice to review a total number of three possible audits, and to not evaluate more possible audits, is also based on the limitations of this research with regard to the available time considering the scope of this graduation research. The use case scenarios are visualized in an information flow schema in figure 9, in order to present a better overview of the location of the requirements within the analysed processes. Note that certain referrals such as 'S1' and 'T1' are used to refer to the steps as shown in the information flow schemas.

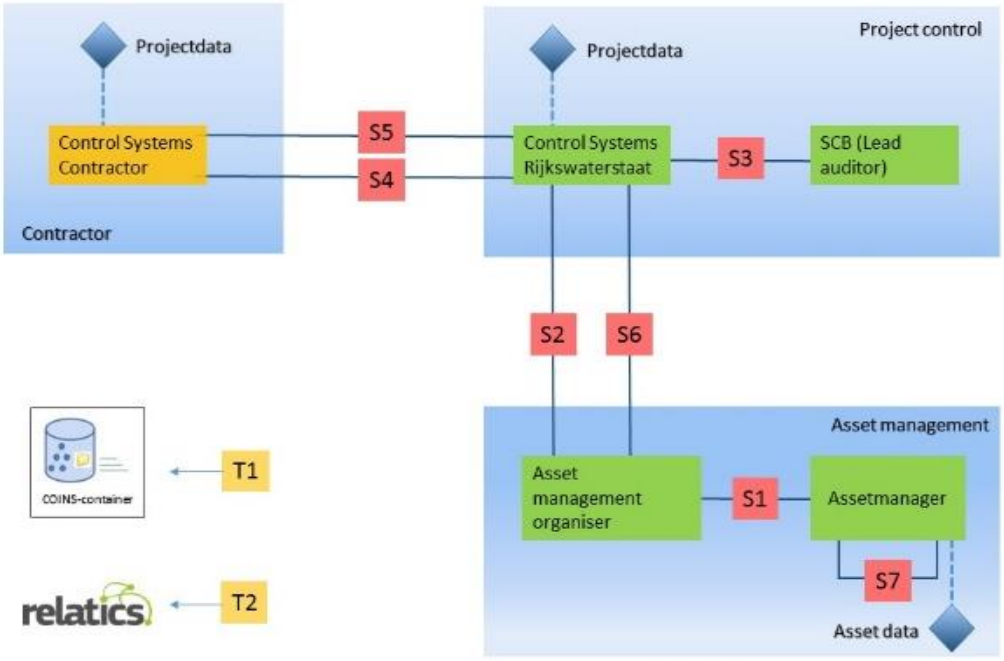


Figure 9. Information flow schema audit processes.

5.2.1. Use case scenario 1: Testing with known geometrics of an objects

The first use case scenario consist of a scenario in which a risk is identified where the thickness of a slap of pavement might be thinner than the minimum thickness that is required in the contract by Rijkswaterstaat. A too thin pavement slap could lead to fractures in the pavement. This scenario is selected because of the fact that the main query of the audit consist of testing existing geometrics of an object within a COINS container. First, the action steps which the user of the tool will take will be presented and corresponding actions and information flows will be defined. Based on these actions and corresponding information flows, conclusions will be drawn considering the requirements for the proposed method. Step 6 is elaborated on a deeper level compared to the other steps since the tool will mostly be used in step 6.

Step 1 (S1) The asset manager at Rijkswaterstaat determines the need to repave the lane of a highway based on infield observations. The asset manager orders the asset management organizer to direct this task.

Step 2 (S2) The asset management organizer collects all asset data of the current situation from the asset management data systems in a COINS container (T1) as objects and linked data. This asset data of the current situation is 'as-is', and therefore reflects the current situation. This COINS container is communicated to the management systems of Rijkswaterstaat. Since Rijkswaterstaat is both the client as the asset manager, this information flow is internal within the organisation.

Step 3 (S3) The project requirements and associated risks within the management systems of Rijkswaterstaat are communicated to the lead auditor. The lead auditor decides to plan several audits based on existing experience with the chosen contractor, experience with similar projects, and presumed impact of failure. One of these audits is to test whether the thickness of the new sheet of pavement is sufficient according to the requirements.

Step 4 (S4) The management systems of Rijkswaterstaat sends the 'as-is' data in form of a COINS container to the contractor.

Step 5 (S5) The contractor plans and realises the work and modifies and complements the COINS container respectively. The contractor sends the adapted COINS container is send to Rijkswaterstaat for verification and validation of the work.

Step 6 (S6) The lead auditor validates the data by performing the planned audits. Among these audits is the planned audit which tests the thickness of the pavement. The audits are based on the requirements which are set up by Rijkswaterstaat. The requirements are stored in a Relatics environment.

The auditor therefore needs to validate and verify the data stored in the COINS container based on the data stored in the Relatics environment. This data is available as a geometric object stored in an IFC-file, with additional information relevant to the object linked via the CBIM file. The auditor needs to select the object, measure the thickness (or 'height') of the layer and compare it with the required thickness of the layer in the Relatics file.

If the data in the COINS container complies with the requirements, the positive audit will be communicated to the management system of Rijkswaterstaat, who will subsequently accept the validated work from the contractor. The positive outcome of the audit will also be stored in the audit system of Rijkswaterstaat in order to include it in future risk calculations. This storing of the output of the audit includes evidence of the outcome of the audit

If the data in the COINS container does not comply with the requirements, the negative audit will be communicated to the management system of Rijkswaterstaat, who will declare the contractors work invalid and will take measures accordingly. The negative outcome of the audit will also be stored in the audit system of Rijkswaterstaat in order to include it in future

risk calculations. This storing of the output of the audit includes evidence of the outcome of the audit

Step 7 (S7) The data from COINS containers that is validated will be used to update the asset data base of Rijkswaterstaat in order to bring the asset data base back up to date on the current situation. In order to check whether the current situation still upholds to (new) requirements, it may also be possible that the asset manager has to do audits on their own data-base of the 'as-is' situation. Therefore the elaborated actions in step 6 regarding the audit may be necessary in step 7 as well.

5.2.1.1. Requirements for the tool according to use case scenario 1

Based on use case scenario 1, a number of requirements can be deducted which would allow the proposed method to aid in the improvement of the SCB process. These requirements are the following:

- The functionality to select an object.
- The functionality to query for selected object geometrics.
- The functionality to acquire requirements and apply as checkable code.
- The functionality to query whether requirements are met.
- The functionality to show the outcome of the comparison functionality.
- The functionality to generate output from the comparison to use as evidence of the audit.
- Additionally: the functionality to visualize the selected object in order prevent mistakes in the object selection process.

5.2.2. Use case scenario 2: Geometric relation between selected 2 objects

The second use case scenario consist of a scenario in which a risk is identified where the height clearance for a bridge over a road might not be high enough according to the requirements. This scenario is selected because of the fact that the main query of the test consist of a comparison between the relational geometrics of two objects within a COINS container.

Similar to use case scenario 1, based on these actions and corresponding information flows, conclusions will be drawn considering the requirements for the proposed method. Step 6 is elaborated on a deeper level compared to the other steps since the tool will mostly be used in step 6. Steps 2, 4, 5, and 7 are identical as the process described in use case scenario 1. These therefore refer to the corresponding step in use case scenario 1.

Step 1 (S1) & Step 2 (S2) Similar step as use case scenario 1 in paragraph 6.2.1.

Step 3 (S3) The project requirements and associated risks within the management systems of Rijkswaterstaat are communicated to the lead auditor. The lead auditor decides to plan several audits based on existing experience with the chosen contractor, experience with similar projects, and presumed impact of failure. One of these audits is to test whether the height clearance for a bridge over a road is high enough according to the requirements. The minimum height is a national standard.

Step 4 (S4) & step 5 (S5) Similar step as use case scenario 1 in paragraph 6.2.1.

Step 6 (S6) The lead auditor validates the data by performing the planned audits. Among these audits is the planned audit which tests the height of the lowest part of the bridge over the road. The audits are based on the requirements which are set up by Rijkswaterstaat. The requirements are stored in a Relatics environment.

The auditor therefore needs to validate and verify the data stored in the COINS container based on the data stored in the Relatics environment. This data is available as a geometric object stored in an IFC-file, with additional information relevant to the object linked via the CBIM file. The auditor needs to select two objects, measure the distance between those two objects and compare it with the required height of the overpass in the Relatics file.

If the data in the COINS container complies with the requirements, the positive audit will be communicated to the management system of Rijkswaterstaat, who will subsequently accept the validated work from the contractor. The positive outcome of the audit will also be stored in the audit system of Rijkswaterstaat in order to include it in future risk calculations. This storing of the output of the audit includes evidence of the outcome of the audit

If the data in the COINS container does not comply with the requirements, the negative audit will be communicated to the management system of Rijkswaterstaat, who will declare the contractors work invalid and will take measures accordingly. The negative outcome of the audit will also be stored in the audit system of Rijkswaterstaat in order to include it in future risk calculations. This storing of the output of the audit includes evidence of the outcome of the audit

Step 7 (S7) Similar as use case scenario 1.

5.2.2.1. Requirements for the tool according to use case scenario 2

Based on use case scenario 2, two new requirements can are identified which would allow the proposed method to aid in the improvement of the SCB process. These two requirements for the proposed method that are an addition to the requirements that are identified in use case scenario 1 following:

- The functionality to select 2 objects
- The functionality to query for the smallest distance between the two selected objects

5.2.3. Use case scenario 3: Angle of embankment

The third use case scenario consist of a scenario in which a risk is identified where the angle of an embankment might be too steep according to the requirements set by Rijkswaterstaat. A too steep angle could lead to the collapse of an embankment. This scenario is selected due to the fact that the main query of the test consist of generating new geometric data based on the existing geometric data of a single object within a COINS container.

Similar to use case scenario 1 and 2, based on the actions and corresponding information flows, conclusions will be drawn considering the requirements for the proposed method. Step 6 is elaborated on a deeper level compared to the other steps since the proposed method will

mostly be used in step 6. Steps 2, 4, 5, and 7 are identical as the process described in use case scenario 1 & 2. Therefore, these are referred to the steps in use case scenario 1.

Step 1 (S1) & Step 2 (S2) Similar step as use case scenario 1 in paragraph 6.2.1.

Step 3 (S3) The project requirements and associated risks within the management systems of Rijkswaterstaat are communicated to the lead auditor. The lead auditor decides to plan several audits based on existing experience with the chosen contractor, experience with similar projects, and presumed impact of failure. One of these audits is to test whether the angle of an embankment is steep enough according to the requirements. A too steep embankment could cause instability of the embankment.

Step 4 (S4) & step 5 (S5) Similar step as use case scenario 1 in paragraph 6.2.1.

Step 6 (S6) The lead auditor validates the data by performing the planned audits. Among these audits is the planned audit which tests the angle of the embankment. The audits are based on the requirements which are set up by Rijkswaterstaat. The requirements are stored in a Relatics environment.

The auditor therefore needs to validate and verify the data stored in the COINS container based on the data stored in the Relatics environment. This data is available as a geometric object stored in an IFC-file, with additional information relevant to the object linked via the CBIM file. The auditor needs to select an object, measure the height and width of the object, calculate the angle of the embankment, and compare it with the required height of the overpass in the Relatics file.

If the data in the COINS container complies with the requirements, the positive audit will be communicated to the management system of Rijkswaterstaat, who will subsequently accept the validated work from the contractor. The positive outcome of the audit will also be stored in the audit system of Rijkswaterstaat in order to include it in future risk calculations. This storing of the output of the audit includes evidence of the outcome of the audit

If the data in the COINS container does not comply with the requirements, the negative audit will be communicated to the management system of Rijkswaterstaat, who will declare the contractors work invalid and will take measures accordingly. The negative outcome of the audit will also be stored in the audit system of Rijkswaterstaat in order to include it in future risk calculations. This storing of the output of the audit includes evidence of the outcome of the audit

Step 7 (S7) Similar step as use case scenario 1 in paragraph 6.2.1.

5.2.4. Requirements for the method according to use case scenario 3

Based on use case scenario 3, one new requirements be deducted which would allow the proposed method to aid in the improvement of the SCB process. The requirement for the tool that are an addition to the requirements that are identified in use case scenario 1 & 2 is the following:

- Functionality to create new geometric data, based on existing geometric data in the COINS container.

5.3. Analysing the requirements for the method

In the previous paragraphs a total number of ten functionalities are identified which the proposed method must be able to fulfil in order to be able to fully perform all three types of audit. This proposed method is the method of using scripted queries in order to perform audits on BIM data that is stored in a COINS container. These ten functionalities are analysed in this paragraph in order to identify the possibilities for the proposed method to improve the use of BIM in the SCB process. Any limitations that are identified are further elaborated on in paragraph 6.4. Each functionality is regarded separately below. Note that this is all functionalities are regarded as a preliminary analysis. The conclusions on the exact design of the proposed method will be presented and elaborated on in chapter 7.

The functionality to select an object is required in order to perform analysis on the data is possible to include in the method. When using the proposed method is possible to select an object by using Python to import files from the COINS container, and SparQL to perform a query that selects the object from the COINS container.

The functionality to query for selected object geometrics is possible when using the proposed method to query for selected objects geometrics by using the SparQL language. This geometric data can be available in two different formats. One format is in an XML-file that is included in the COINS container, in which the geometrics are stored using the Geography MarkUp Language (GML). This geometric information may also be available as relational data in a RDF file or OWL file within the COINS container, which can be retrieved via SparQL query.

The functionality to acquire requirements and apply as checkable code is partially possible to include in the method. In the literary research section of this report it is determined that code checking compliance of sets of requirements is still a topic of research. Requirements vary too much from project to project for all requirements to be semantically accurate and hereby machine-readable. A solution to this current problem is to manually convert the requirements to audits, which is already part of the SCB process.

The functionality to query whether requirements are met can be achieved when using the proposed method by using the Python language to compare the selected requirement with the data of the selected object. The conclusion on this comparison can automatically be generated by the Python code.

The functionality to show the outcome of the comparison made in the previously mentioned functionality can be achieved in the proposed method due to the ‘print’ functionality of Python. The Python language can use the ‘print’ command to print one of several predetermined messages based on outcome of a comparison.

The functionality to create evidence based on the performed audit is possible when using the proposed method. A XLS spread sheet data file can be created by using the Python code. The Python code can store the outcome of the audit related calculations and findings in this XLS file. This XLS file can be used as evidence in the SCB process.

The functionality to visualise the selection made by the script cannot be directly included in the script due to the fact that there is not (yet) a working Python module discovered that can visualize the GML data that is stored in the XML files. It is however possible to copy the selected XML file from the COINS container and present this copy to the user. The user can then use a different program to visualize the GML data from the files, hereby making it possible to review the selection in order to reduce mistakes in the selection process.

The functionality to select 2 objects is possible when using the proposed method via the Python and SparQL languages. The selection of two objects is required when the requirement aims at data that is not included specifically in the dataset, but as implied data between two objects in the COINS container.

The functionalities of querying for the distance between two selected objects is not possible when using the proposed method. This functionality is required when performing audits such as the requirement for the minimum height clearance of a road underneath an object such as a bridge. The reasons on why the proposed method lacks this functionality is elaborated in the next chapter.

The functionality to create new geometric data, based on existing geometric data in the COINS container, is required for audits that tests requirements based on a maximum or minimum angle, such as the as the maximum angle of an embankment.

5.4. Analysing the limitations of the method

The limitations of the proposed method that are identified in the paragraph 6.3 are elaborated in this paragraph. This proposed method consists of using scripted queries to perform audits on BIM data that is stored in RDF or OWL in a COINS container. In the previous paragraphs a total number of ten functionalities are identified which the proposed method must be able to fulfil in order to fully be able to perform all three types of audit. Two of these functionalities do not fit in any of the functionalities which the proposed method offers.

5.4.1. A limitation of the proposed method: calculating geometric data

The functionality of creating geometric data is required in order to determine a maximum or minimum angle between two surfaces based on RDF data. When reviewing an object that is stored in a data file with geometric properties, such as for example an object in an IFC file, a software program with a geometric engine is required. The geometric engine creates the dots, lines, and surfaces from the data file en presents the information in 3D in a viewer. IFC files cannot be imported by any Python module, nor can they be queried by the SparQL language since they are not stored as RDF data. This means that the proposed method contains no option to select or calculate the required information in the COINS container in order to test in on the requirement.

There are IFC-to-RDF and IFC-to-OWL converters available, as is discovered in the literature research. These converters allow data that is stored according to an IFC schema to be interoperable with Semantic Web technology. However, converting the 3D geometric data stored using an IFC schema to 3D geometric data stored as RDF data is not considered as an option for this research. This is a result of the fact that currently the conversion of 3D

geometry still leads to inconsistencies (P Pauwels, Meyer, & Campenhout, 2010). The reason for these inconsistencies is that same geometry can be described in many different ways using varying schemas. One schema might, for example, describe a sphere through a circular arc and a central axis, another might describe a sphere through a triangular mesh. This can lead to inconsistencies in the description or over-simplified geometry as a result of the translation process. An alternative approach is to rely on rules and an inference engine to infer the duplicate information (P Pauwels & Deursen, 2011). However, further research is required for this alternative approach to conclude for the practical applicability of this approach.

The proposed method uses Python to import files and execute commands, and SparQL to select objects in the RDF or OWL files and retrieve data. Since both the languages Python and SparQL do not have the capabilities of a geometric engine, they cannot perform the required calculations in order to determine the greatest angle of embankment. The proposed method can therefore not be used when performing an audit that deals with requirements set to the angle of an embankment. However, this might change when future research develops IFC-to-OWL converters that allow a complete conversion of geometric data without any inconsistencies.

5.4.2. A limitation of the proposed method: comparing geometric objects

The functionality of comparing the geometrics of two selected objects in order to determine the distance between two surfaces of these objects is not a functionality which the proposed method offers. When trying to identify the lowest height clearance of a road underneath a bridge for example, both the bridge and the road have to be visualized. If both these objects are stored in an IFC files and loaded into a software program with a geometry engine, a 3D representation can be created as is visualised in figure 10.

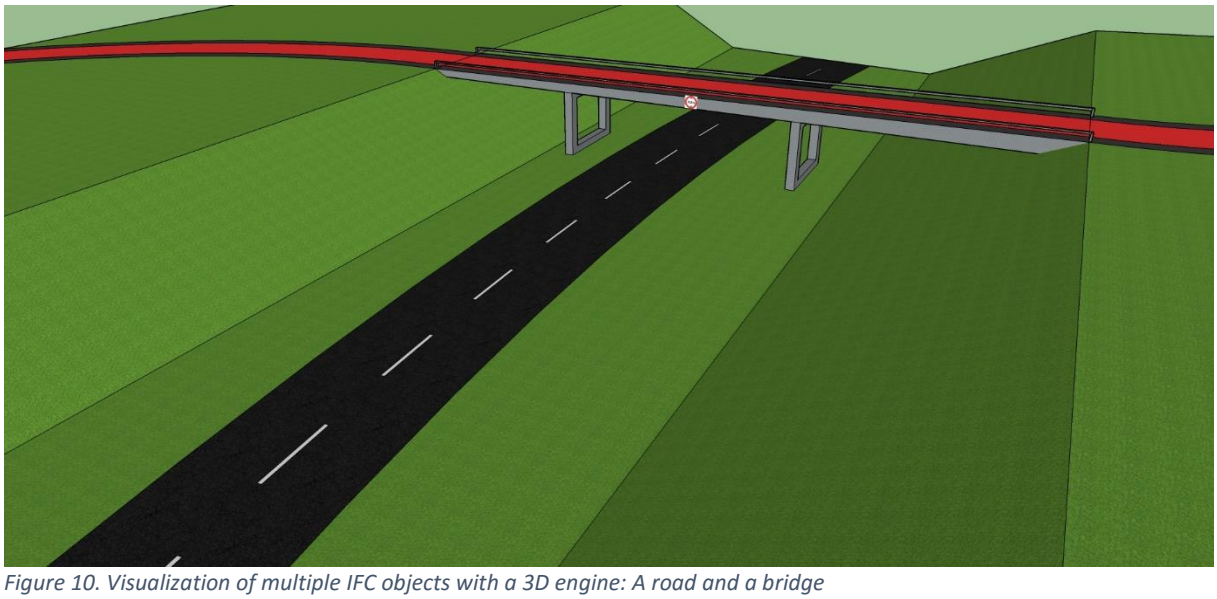


Figure 10. Visualization of multiple IFC objects with a 3D engine: A road and a bridge

The opportunity arises to calculate the lowest height clearance due to the fact that the geometry engine has calculated and portrayed all surfaces included in the data file in a 3D environment. The smallest distance between the surface of the road and the surface of the lowest part of the bridge can be calculated by selecting both of these surfaces and use a

specialized command. Another option is clash detection. This option is presented in the model review in chapter 8.

The proposed method of using a script that consists of Python and SparQL in order to extract data from a COINS container and to test this data on the requirements set by Rijkswaterstaat does not include the functionality of comparing two geometric objects. Similar as to the limitation presented in paragraph 6.4.1, IFC cannot be imported by Python or SparQL.

As stated before, geometric data in an IFC file, such as for example the data presented in figure 11, cannot be converted into RDF or OWL without inconsistencies. There are no functionalities or possible scripts within Python or SparQL to calculate the surfaces and determine the shortest distance between two selected surfaces.

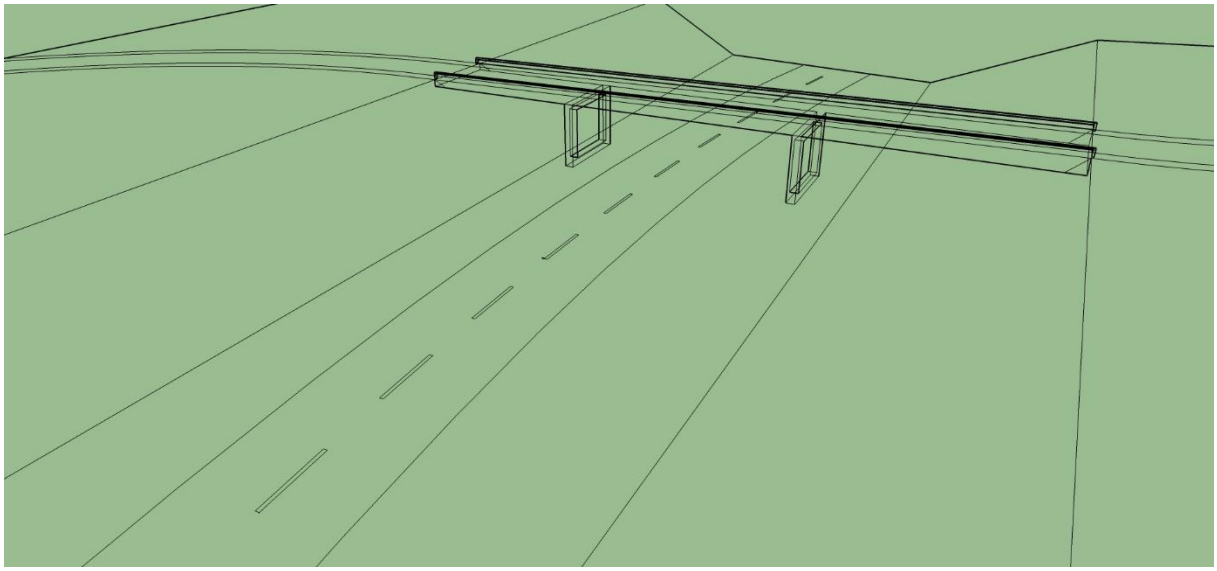


Figure 11. Theoretical visual representation of 2 objects Theoretical visual representation of two geometric objects stored in RDF or OWL.

5.5. Analysing a potential use of the method

The proposed method of using queries with Python and SparQL languages can automatically perform audits that check the thickness of a pavement slab on requirements set by Rijkswaterstaat. The geometrics of a slab of pavement is included in the COINS container as a 2D representation of the slab as GML data in a XML file. The thickness of a pavement slab is included in the COINS container as a value of an OWL file. This thickness is based on design by the contractor or included in the COINS container after the realisation of the pavement slab. Rijkswaterstaat can require the contractor to implement the thickness of the pavement slab as a value related to the pavement slab in the OWL file in order to make certain that the value is always stored in a uniform manor.

By using the Python language to generate a copy of the xml file containing the GML data of the selected object in a predetermined file, one can use a GML viewer to create a visualisation of the selection as is presented in figure 12. An example of such a program is Quantum GIS. The auditor can hereby visually check the selected object in order to reduce the risk of selection mistakes.

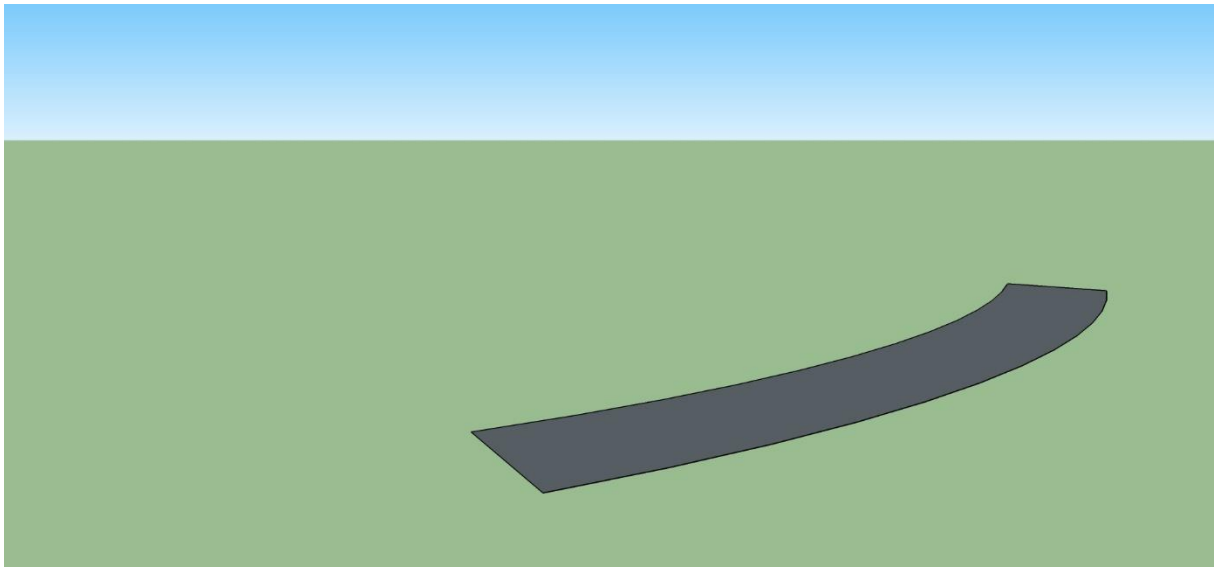


Figure 12. Theoretical visualisation of a pavement slab as GML data.

5.6. Conclusion

The SCB process at Rijkswaterstaat and the implementation of BIM in the SCB process can be improved by creating a method which includes all the functionalities that are identified in paragraph 6.2 and paragraph 6.3 of this chapter. Alternate solutions to the proposed method have to be used in order to overcome the limitations that have been identified in paragraph 6.3 and which are elaborated on in paragraph 6.4.

This will lead to t The BIM tool hereby improves the process on multiple levels. These expected levels of improvements achieved by implementing the proposed method are:

- Speeding up the process of the audits and reducing the labour intensity
- Making the process more accessible for new (BIM) users.
- Standardising the process in such a way that errors are less likely to occur.
- Allowing the user to have a better understanding of what data the user is assessing via a 3D GML viewer, hereby reducing risks of selection errors.

6. Designing of the method and tool

In this chapter, a script is created in in order to improve the current audit processes within the SCB processes at Rijkswaterstaat. This script will include a GUI which will allow the user to execute the tasks without any mandatory background knowledge on the programming language that is used to run the script. Hereafter, this script will be referred to as 'the tool'. Paragraph 7.1 will include an introduction to the design and creation of the tool. Thereafter, in paragraph 7.2, an analyses is presented the types of data which have to be retrieved from the BIM data in order to conduct the audit. The design of the script will be presented in paragraph 7.3. This design will be based on the prior acquired requirements for the script. The design of the Graphic User Interface (GUI) is presented in paragraph 7.4 in order to present the ease of the user experience of the tool. Lastly, the redesign of the SCB process with the implementation of the use of the tool is presented in paragraph 7.5

6.1. Introduction

The tool and corresponding redesigned SCB process are based on the required functionalities that are acquired in chapter 6. These required functionalities have to be included in the script and redesigned process in order to fully improve the implementation of BIM in the current SCB process at Rijkswaterstaat. The redesigned SCB process wherein the use of the new tool is implemented will further be referred to as 'the proposed method'. The proposed method is specifically designed for an automated audit regarding the requirements for the minimum thickness of a slab of pavement. However, the major part of the method can be used for other audits as well. Since only small adaptions of the method are required in order to use the method for other audits, we regard the method as a prototype method for audits by Rijkswaterstaat. These other possible audits will be discussed later in this chapter and following chapters.

The script is constructed using the programming language Python language and query language SparQL. Multiple Python modules were used to facilitate several functionalities of the script. These will be presented in the following paragraph. During the design phase, the Integrated Development Environment (IDE) PyCharm is used in order to design and run the scripts. The program Notepad ++ was used during the construction of the script in order to analyse data packages.

After the completion of the design of the script, the SCB process is adapted in order to implement the script in the overall processes at Rijkswaterstaat. The Python module Tkinter is used in order to create the GUI for the tool. This allows the user to use the script without being required to know how the Python programming language is constructed or operates.

6.2. Analyses of the data types

The two main data sources that are encountered in the SCB audit process at Rijkswaterstaat are the requirements in an audit XLS spreadsheet file and the BIM data in a COINS container. The XLS spreadsheet file is the standard file through which Rijkswaterstaat currently communicates relevant data on requirements. The COINS container is provided by the contractor and this contains the data on which the audit is performed.

The data in the COINS container consists of the entire BIM. This data is compressed by the standard zipping algorithm used by Windows. In order to make the data readable for the Python language it is therefore unzipped in a standard folder. This folder is generated by the script on a location specified by the user and will be used to store relevant audit data, including data output as well as the unzipped COINS Container.

The unzipped COINS container consists the two folders ‘bim’ and ‘doc’ as can be seen in figure 6 in chapter 5. The folder ‘bim’ container contains one OWL file and one XML file. The folder ‘doc’ contains all other data in various formats. The OWL file contains semantic data that is coupled via relations with the data in the doc folder. A code snippet of the OWL file is presented below in figure 13. This code snippet is made by opening the OWL file using the program Notepad ++. The OWL file in the example below is created by Rijkswaterstaat in a BIM related pilot case. Some sentences are therefore displayed in Dutch.

```
1093517 <rdf:Description rdf:about="http://www.buildingbits.nl/validatieContainer.rdf#_BB507node1a398qjj2x716">
1093518     <rdf:type rdf:resource="http://www.coinsweb.nl/cbim-1.1.owl#PropertyValue"/>
1093519     <cbim:creationDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2015-11-04T13:56:01+01</cbim:creationDate>
1093520     <cbim:creator rdf:resource="http://www.buildingbits.nl/validatieContainer.rdf#node1a398qjj2x4083"/>
1093521     <cbim:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Toplaag (weg) heeft Geluidsreductie tijd</cbim:name>
1093522     <cbim:value rdf:datatype="http://www.w3.org/2001/XMLSchema#int">3</cbim:value>
1093523     <cbim:propertyType rdf:resource="http://bim.rws.nl/OTL/COINS/otl-otl.11.owl#PR00599.0"/>
1093524 </rdf:Description>
1093525
1093526 <rdf:Description rdf:about="http://www.buildingbits.nl/validatieContainer.rdf#_BB857node1a398qjj2x717">
1093527     <rdf:type rdf:resource="http://www.coinsweb.nl/cbim-1.1.owl#PropertyValue"/>
1093528     <cbim:creationDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2015-11-04T13:56:01+01</cbim:creationDate>
1093529     <cbim:creator rdf:resource="    "/>
1093530     <cbim:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Asfaltplak heeft Werkelijke dikte</cbim:name>
1093531     <cbim:value rdf:datatype="http://www.w3.org/2001/XMLSchema#float">47</cbim:value>
1093532     <cbim:propertyType rdf:resource="http://bim.rws.nl/OTL/COINS/otl-otl.11.owl#OB02859-PR00501.0"/>
1093533 </rdf:Description>
1093534
1093535 <rdf:Description rdf:about="http://www.buildingbits.nl/validatieContainer.rdf#_BB780node1a398qjj2x718">
1093536     <rdf:type rdf:resource="http://www.coinsweb.nl/cbim-1.1.owl#PropertyValue"/>
1093537     <cbim:creationDate rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2015-11-04T13:56:01+01</cbim:creationDate>
1093538     <cbim:creator rdf:resource="http://www.buildingbits.nl/validatieContainer.rdf#node1a398qjj2x4083"/>
1093539     <cbim:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Verhardingslaag heeft Laagnummer</cbim:name>
1093540     <cbim:value rdf:datatype="http://www.w3.org/2001/XMLSchema#int">4</cbim:value>
1093541     <cbim:propertyType rdf:resource="http://bim.rws.nl/OTL/COINS/otl-otl.11.owl#PR00502.0"/>
1093542 </rdf:Description>
```

Figure 13 Cut out from a COINS container created by Rijkswaterstaat.

The middle section of the example shows a property value of an object of which the name is ‘Asfaltplak heeft werkelijke dikte’, which translates from Dutch to: ‘Pavement slab has actual value’. The described value of the property of the object is ‘47’, which would means that the contractor has entered the value 47 mm as thickness of the pavement slab.

An error in this file, which could easily be solved, is the fact that it cannot be deducted whether the length measurement of the value ‘47’ is in millimetres or in meters. This can be solved by including the length measurement of millimetres in the property name, since it is not included in the URI ‘cbim:value’. These titles can be prescribed by Rijkswaterstaat to the contractor in an information delivery manual (IDM) in order to prevent errors. For this research we assume that the measurement of the given value is in millimetres.

6.3. Design of the tool

The design of the script consists of modules which are called in several steps which the user will walk through. These steps are presented to the user in the GUI which is created in the script. First, the setup of the GUI will be presented in chronological order to create an overview of the functionalities of the script. The order of these steps will be used as path through which the tool will be explained. This sequence of steps will further be referred to as the ‘user path’. The workings of the main segments of the script are presented in the next paragraph. The completed script is included in Appendix 3. This appendix includes the full script as well as

explanatory text on the main parts of the script that is included by use of the notation function '#'.

The script is wrapped into a single executable file. When this executable file is launched, a GUI will be opened. In this GUI, step 1 is presented in the window. Each time when a step is completed, the window will automatically load the next step. The layout of the possible window configurations of the GUI are presented in Appendix 2. The windows of each step are constructed from a grid which includes a main label presenting the number and title of the step, a brief explanation on the required actions, and all list boxes, entry boxes and buttons required to provide the required data or perform the required action. Once the action or all required actions of a step are undertaken, the script will clear all labels and other attributes of the step and present all labels and attributes of the next step. The user will therefore always only see the step and corresponding action that he has to perform, making the tool more user friendly. These steps are based on a predetermined user path for each specific audit.

The setup of the path is based on the required functionalities that have been identified in chapter 6. The steps are designed for 2 user scenarios. This path is presented in figure 14 in paragraph 7.3.2 for scenario 1 and in figure 15 in paragraph 7.3.1 for scenario 2. These figures present the purpose of each step and the functionality that is added in order to facilitate this purpose (e.g. a button to open a directory search window). Each step consist of one main user action. The user path of each of these scenarios is elaborated on further in the two following subparagraphs. First, the purpose of each step will be described and thereafter the attributes that facilitate the user actions.

6.3.1. Setup of user path for scenario 1

The first scenario consists of the most common scenario. Scenario 1 is a scenario in which the user is an auditor at Rijkswaterstaat who performs a standard audit. The standard audit process at Rijkswaterstaat includes the retrieval of a standard audit XLS file from the UR-SCB environment and the obliged delivery of proof of audit output by the auditor, as is elaborated on in chapter 5. The steps are further elaborated below in order to present an elaborate on the user path.

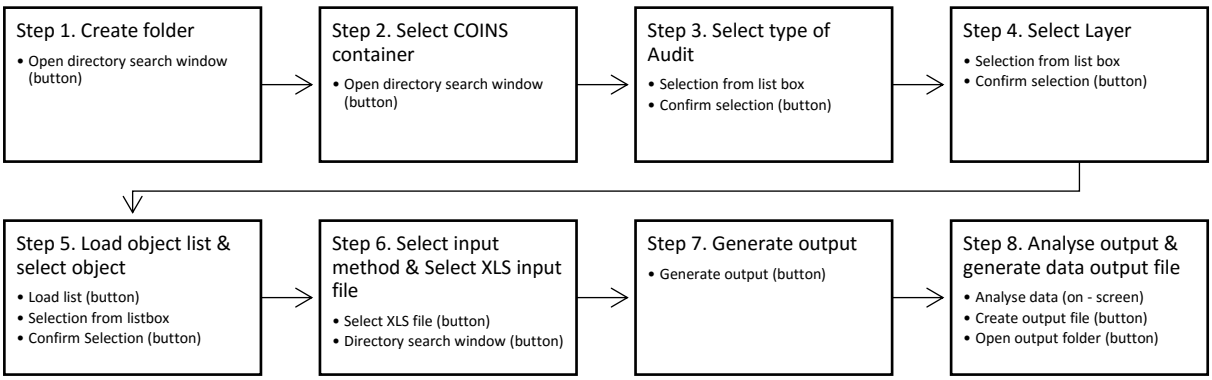


Figure 14 User path for scenario 1.

The purpose of **step 1** is to create a folder on the computer system in which the COINS container will be extracted into its uncompressed form. The output file of the audit will also be stored in this folder. The folder is created by use of the directory search window option of the Tkinter module. This function is called via a button labelled 'Create/select

folder' in the user interface. After the folder selection is confirmed, all attributes from label 2 will emerge and replace the labels of step 1.

The purpose of **step 2** is to select the COINS container which is required to be audited. Similar to step 1, a button with the label 'select' is clicked in order to launch the directory search window option of Tkinter.

The purpose of **step 3** is to select an audit from an available list of audits which the tool can perform. The list is created by use of the Tkinter list box option. Once an audit is selected it will be highlighted in the list box. The user can enter the selection by clicking on the button with the label 'select'. In this scenario, the selected audit would is titled 'Thickness of pavement slab'.

The purpose of **step 4** is to select the layer of pavement. Rijkswaterstaat distinguishes three types of pavement layers. These are the Top layer, Middle layer, and Bottom layer. Similar to step 3, the options are presented in a list box, and the selection is confirmed by clicking a button with the label 'Select'.

The purpose of **step 5** is to load the object list from the COINS container that has been selected in step 2. The selection is based on the type of audit that is selected in step 3 and the type of layer criterion that is provided in step 4. In order to load the list, a button is clicked with the label 'Load object list'. The list of objects will be loaded into a list box. After the list is loaded, the 'Load object list' button will replaced with a button labelled 'Select' to confirm the selection.

The purpose of **step 6** is to select the method of data input and to provide this input. The input for the audit consist of the requirement for the selected object. Two buttons are provided, of which one is labelled 'Select XLS file' and of which the other button is labelled 'Manual input'. In scenario 1, this input is provided in form of a standard XLS spreadsheet file. The button labelled 'Select XLS file' is therefore clicked. This action opens a Tkinter directory search window in which the XLS file can be selected. The resulting process of the selection of the button labelled 'Manual input' is presented in scenario 2 in paragraph 7.3.2.

The purpose of **step 7** is to perform the audit and generate audit data output. Only one button labelled 'Perform audit' is presented in the window of step 7. When the user clicks this button, the comparison between the required value and obtained value will be made and the conclusion will be presented. This step is inserted in the tool to create certainty for the program user that the audit has been executed and that output can now be generated.

The purpose of **step 8** is to present the analysed data and conclusions, create data output that can be used as audit results, and open the folder in which the output is stored. The analysed data will be presented on screen, including the obtained object value and the obtained required value. In the screen will clearly be stated whether the audit outcome is positive (in a green font) or negative (in a red font). The window of step 8 presents two buttons, respectively the button labelled 'Create output file' and the button labelled

'Open output folder'. This window also includes a message that the audit has been completed and that the window can be closed using the '[x]' at the top right of the screen.

The audit results will be uploaded by the user in the UR-SCB online environment of Rijkswaterstaat to serve as evidence of the audit. At this point, the standard SCB process will be resumed and an appropriate responds will be based on the positive or negative audit outcome.

6.3.2. Setup of user path for scenario 2

Second scenario consists of a second common scenario. Scenario 2 is a scenario in which the user is not performing an official audit at Rijkswaterstaat, but wants to perform a quick common check on an object value in the COINS container. An example of such a check at Rijkswaterstaat is an Asset Manager who wants to check a model in the asset data base, which is stored in a COINS container.

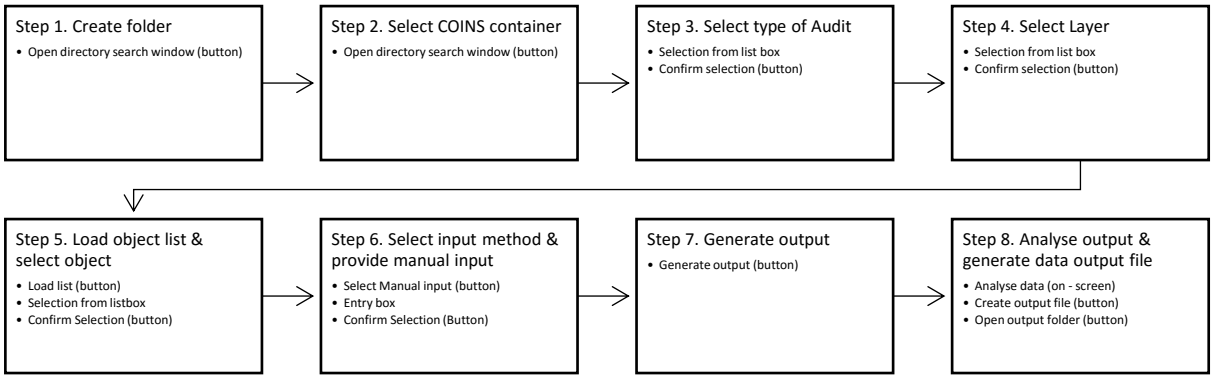


Figure 15 User path for scenario 2.

As can be derived from the two figures presenting the scenario user paths, step 1 to 5 are identical for both scenarios. The description of scenario 2 therefore starts from step 6.

The purpose of **step 6** is to select the method of data input and to provide this input. The input for the audit consist of the requirement for the selected object. Two buttons are provided, of which one is labelled 'Select XLS file' and of which the other button is labelled 'Manual input'. In scenario 2, this input is provided manually. When the user clicks on the button labelled 'Manual input', an entry box and a button labelled 'Confirm entry' will appear. The auditor has to manually type the value of the required value in the entry box and click on the button to confirm this entry.

The purpose of **Step 7** is identical to the purpose of step 7 in scenario 1, which is to perform the audit and present conclusions on the outcome of the audit.

The purpose of **Step 8** is to present the acquired values provide and present the conclusion on the comparison between the two values. Since this is not an official audit but a simple test, no option is included to create audit data output. The user can close the window using the '[x]' in the top right of the screen after interpreting the audit results.

The asset manager will use the output from the check to start appropriate asset management processes.

6.4. Implementation of the script

In this paragraph the implementation of the script is elaborated on. First, the modules used in the script are described. Thereafter, a selection of key functionalities of the script are described in the order in which they are called on when following the user path.

This paragraph will not include the full script, but a description of the most important sections of the script. The full script is included in Appendix 3. The windows created by the GUI are presented in Appendix 2.

6.4.1. Python modules used

Several Python modules were imported used in the script in order to facilitate the various functionalities of the tool, including amongst others the creation of the GUI, the retrieval of values, and creation of data output. These modules and their main functionality are presented first.

The module **Tkinter** is imported in order to create the GUI. Tkinter allows the user to define a GUI and place, remove, and forget attributes in the window using a grid or placement functionality. Examples of such attributes are list boxes, buttons, entry boxes and labels. Tkinter can return values or button triggers in order to connect other Python functionalities with user interaction in the GUI. In this script, the grid functionality is utilized to manage attribute placement.

The module **Zipfile** is imported in order to extract the files from the compressed COINS container format and place them in a folder in an uncompressed state. This allows other modules to interact with the contents of the COINS container. The Zipfile module can uncompress COINS containers since these are compressed via standard ZIP file algorithm.
The module **Rdflib** is imported in order to extract information from the OWL file in the COINS container. Rdflib creates possibilities for this functionality by creating the opportunity to implement the query language SparQL in the python script. This allows for the retrieval of RDF based data.

The module **Os** is imported in order to utilize the modules functions on pathnames and to create folders. This allows the tool to create a folder in which the data from the uncompressed COINS container and the output data from the audit are stored.

The module **Xlrd** is imported in order to extract the data from XLS spreadsheet files. This allows the script to retrieve the required values from the standard XLS spreadsheet files that are used when planning audits at Rijkswaterstaat.

The module **Xlutils** is imported in order to facilitate the functionality to copy and edit an XLS file. This allows the script to create a standard XLS spreadsheet file which contains the audit results and can be used as evidence in the SCB processes of Rijkswaterstaat.

The module **Webbrowser** is imported in order to create the functionality of opening the folder used to store audit data. This increases the user friendliness of the tool and also prevents

mistakes of possibly opening the wrong folder via a manual approach and retrieving a wrong audit output data file.

The module **Gc** is imported in order to gain access to the functionalities of the underlying memory management mechanism of Python. The module is used to collect and clear garbage that is generated during the audit when the tool is closed.

6.4.2. Elaboration on key sections of the script

A selection of sections of the script are described in this subparagraph. These include the sections of the script that facilitate the main functionalities for the separate steps in the user path. The order of the user path will be maintained to explain each functionality. Code snippets of the original script are presented in this paragraph. Note that some of textual sentences in the snippets are shortened in this paragraph.

A major part of the code consist of lines of code which define the setup of the GUI. These lines of code are repeated in a similar form throughout the script to add and/or remove attributes such as labels, entry lists and buttons throughout the user path. The GUI for step 1 that is created by these lines of code is presented below in figure 16. All possible GUI configurations are presented in appendix.

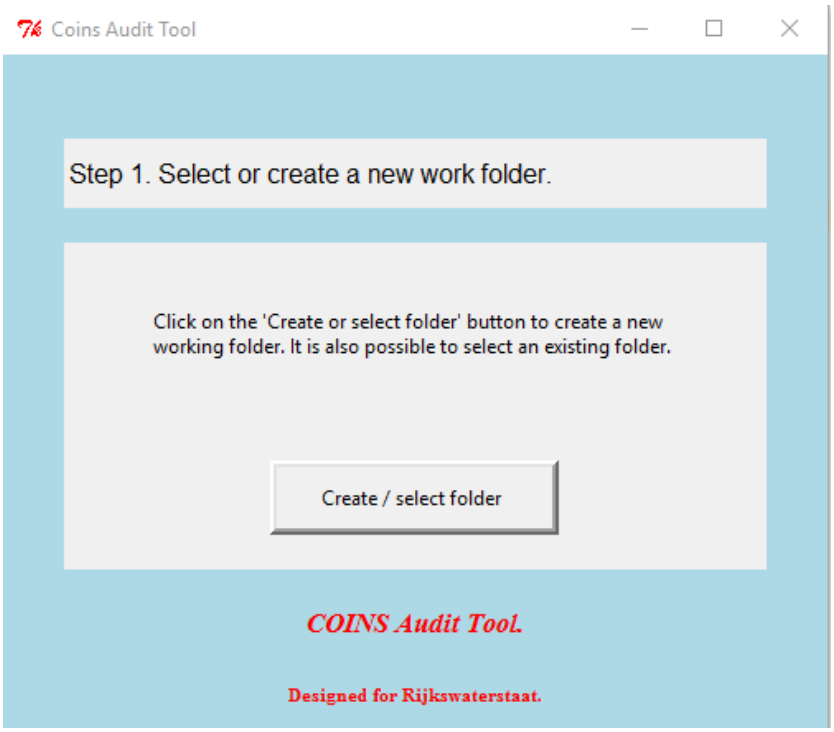


Figure 16. The GUI window, setup for step 1.

The method used to place the various pre-defined attributes on the GUI is the 'grid' function of Tkinter. The lay-out of each step is created by adding and removing these tributes from the grid. As can be seen in the last two command lines from the code snippet above, a button is placed on the grid. When the button is clicked, it executes a command named '`command=self.createfolder)`'. The actions related to this command are defined in the script. The code snippet of the definition of the related actions are presented in listing 1.

```
140.    def createfolder(self):
141.
142.      self.folder_location = tkFileDialog.askdirectory(**self.dir_opt)
143.
144.      if not os.path.exists(self.folder_location):
145.        os.mkdir(os.path.expanduser(self.folder_location))
```

Listing 1. Folder creation.

The first and second step in the user path both require a functionality that opens a directory search window. Settings for the directory search window, such as the initial search folder and title of the search window, are included in the script. As presented above in listing 1, the Python Os module is used in step 1 to create the new folder if it does not yet exist.

A similar command is used to identify the COINS container in step 2. The definition of the actions in step 2 are extended with a few lines of code that identify the OWL file in the folder named 'bim' in the uncompressed COINS container, as is shown in listing 2.

```
167.       if self.selectCOINSc:
168.
169.         zip_ref.extractall(self.folder_location)
170.
171.       # Defining the folder location path in form of a string.
172.
173.       owlpathtupple = self.folder_location, "/bim/"
174.       owlpath = "".join(owlpathtupple)
175.
176.       for file in os.listdir(owlpath):
177.
178.         if file.endswith(".owl"):
179.           owlfilename = file
180.           owlfilepathtupple = (owlpath, owlfilename)
181.           self.owlfilepath = "".join(owlfilepathtupple)
```

Listing 2. COINS containter identification and uncompression.

Step 3 utilizes a list box of which the entries are predetermined. The user can select one of the possible audits and confirm the selection. The code to create the list is presented in listing 3. Currently, only one audit is fully included in the tool. This is the audit 'Thickness of pavement slab'. Another example of an audit which can be included using almost identical code is the audit 'Average size of grain in asphalt mixture'. Due to the time restrictions and scope of this research, this audit has been left out of the prototype tool. Note that in the current version, all audits are hard-coded into a single script. This property of the script is discussed in part D of this report.

```
44.     self.List_1 = Listbox(self, height=3, width=55, selectborderwidth=2, selectmode='sin
       gle', exportselection=0)
45.         self.List_1.insert(1, 'Thickness of pavement slab')
46.         self.List_1.insert(2, '[for example] Average size of grain in asphalt mixture')

47.         self.List_1.insert(3, 'Audit #3')
48.         self.Button_step3 = Button(self, text="Select", width=23, height=2, borderwidth=
       4, command=self.AuditSelection)
```

Listing 3. Creation of listbox containing audit types.

The same method of list creation and selection confirmation has been used for the selection of the layer of pavement slab. The choice of audit and layer are both used as search criteria for the object list in step 5. A SparQL query is used to query for the object list, using the Rdflib module. The section of the script containing the query for the object list is presented in listing 4.

```
264.        g=rdflib.Graph()
265.        g.load(self.owlfilepath, format='xml')
266.
267.        self.qres = g.query(
268.
269.            """SELECT DISTINCT ?value ?name ?file ?frag
270.               WHERE {
271.                  ?pv a cbim:PropertyValue.
272.                  ?pv cbim:propertyType <http://bim.rws.nl/OTL/COINS/otl-
       otl.11.owl#OB02859-PR00501.0> .
273.                  ?pv cbim:value ?value.
274.                  ?asfaltplakCP cbim:propertyValue ?pv.
275.                  ?asfaltplakCP cbim:name ?name.
276.                  ?asfaltplak cbim:contains[cbim:cataloguePart ?asfaltplakCP].
277.                  ?asfaltplak cbim:shape ?rep.
278.                  ?rep cbim:documentAliasFilePath ?file.
279.                        }""",
280.
281.             initNs=dict(
282.                cbim=Namespace("http://www.coinsweb.nl/cbim-1.1.owl#")))
```

Listing 4. SparQL query for objects and corresponding values for thickness of pavement.

The query presented above returns a list of objects with corresponding filenames and thickness of pavement. Line 272, as can be seen above in listing 4, is the line of code in the SparQL query that adds the selection criterion which only selects objects that correspond with the selected audit type. This criterion is the requirement for the objects in the CBIM file of the OWL container to have the cbim:propertyType ‘Actual thickness of pavement’ (translated from Dutch). The selection criteria for the list of the cbim objects and corresponding properties is presented below in figure 17.

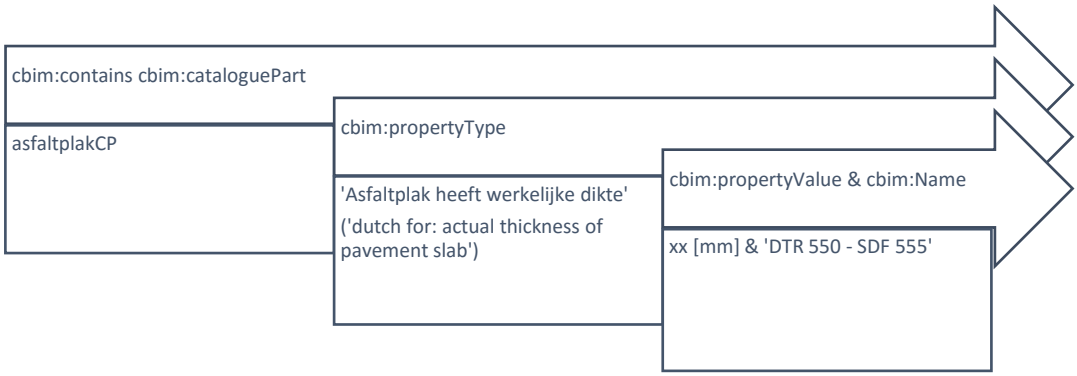


Figure 17. Query criteria in the cbim.

The selection retrieved by the query is used to create a list of objects from which the user can select the required object. The code used to translate the query results into a selection list is presented below in listing 5.

```
281.     for row in self.qres:
282.             if rdflib.term.Literal(self.selectedlayer, datatype=rdflib.term.URIRef(u'htt
        p://www.w3.org/2001/XMLSchema#string')) in row:
283.                 output = row['file']
284.                 self.lst = output.split("\n")
285.                 self.List_3.insert("end", *self.lst)
```

Listing 5. Creation of a list of objects that comply with selection criteria.

The list is loaded when the button 'Load object list' is clicked. The loading of the list will take several minutes since the computer system has to scan the entire OWL file for all the objects that fit the selection criteria. The user can select the relevant object in the list box and click on the button 'Confirm'. The value for the thickness of the pavement of the selected object is now retrieved by the code in listing 6. This value is the thickness of the pavement slab in millimetres and is stored as the variable '`self.obtainedvalue`'.

```
436.         for row in self.qres:
437.
438.          if (rdflib.term.Literal(self.selectedlayer, datatype=rdflib.term.URIRef(u'http:
        //www.w3.org/2001/XMLSchema#string')) in row) and (rdflib.term.Literal(self.selected
        object, datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#string')) in
        row):
439.
440.             self.obtainedvalue = row['value']
```

Listing 6. Identification of the value for 'thickness of pavement' of the object that complies with the selection criteria.

Step 6 and 8 are different for use scenario 1 and user scenario 2. The path and corresponding functionalities for scenario 1 and scenario 2 are presented respectively for each scenario below.

Step 6 for **scenario 1** consists of the selection of the method that uses the standard XLS spreadsheet file from Rijkswaterstaat as input method. Again the directory search window function of the Tkinter module is used to allow the user to select the appropriate XLS file. When this file is selected, the Xlrd module is used to acquire the required value for the audit from the XLS file. The code used to retrieve the value is presented in listing 7.

```
343.                 self.workbook = xlrd.open_workbook(self.XLSfilelocation)
344.                 self.sheet = self.workbook.sheet_by_index(0)
345.                 self.required_value = self.sheet.cell_value(1, 12)
346.                 self.required_value_output = StringVar()
347.                 self.required_value_output.set(self.required_value)
348.                 self.obtained_value_output = StringVar()
349.                 self.obtained_value_output.set(self.obtained_value)
```

Listing 7. Retrieval of the audit dat from the official Rijkswaterstaat audit XLS file.

When the button 'Generate output' in step 7 clicked, the correct selection of labels to presenting the audit outcome are placed on the grid in the window of step 8. The placement

of labels on the grid have already been presented and specific code snippets of these labels has not been included in this chapter. However, another function of step 8 is to create data output in form of a XLS spreadsheet file using the Xlrd and Xlutils modules. The code that creates the data output file is presented in listing 8.

```
379.     def generateoutputxls(self):
380.
381.         tuple_xls_savename = self.folder_location, "/Audit results.xls"
382.         xls_savename = "".join(tuple_xls_savename)
383.         df = self.XLSfilelocation
384.         rb = open_workbook(df)
385.         wb = copy(rb)
386.         s = wb.get_sheet(0)
387.         s.write(0,14, "Are requirements met?")
388.         s.write(0,15, "Selected layer")
389.         s.write(0,16, "Selected object")
390.         s.write(1,15, self.List_2_Selection)
391.         s.write(1,16, self.selectedobject)
392.         s.write(1,13, self.obtained_value)
393.
394.         if self.required_value <= self.obtained_value:
395.
396.                     s.write(1,14, 'Meets the requirements, positive audit outcome.')
397.                     wb.save(xls_savename)
398.                     self.Label_message8xlsP2.grid_remove()
399.
400.         if self.required_value > self.obtained_value:
401.
402.                     s.write(1,14, 'Does not meet requirements, negative audit outcome.')

403.                     wb.save(xls_savename)
404.                     self.Label_message8xlsN2.grid_remove()
```

Listing 8. Creation of the audit output file in xls file format.

Another functionality that has been provided in step 8 is to instantly open the folder which contains the data output file and uncompressed COINS container. This functionality is created using the module Webbrowser. The lines of code used for this functionality are presented in listing 9.

```
417.     def openfilelocation(self):
418.
419.       webbrowser.open(self.folder_location)
```

Listing 9. Faciliate the 'open folder' fuction.

Step 6 in the user path of **scenario 2** includes the manual entry of the required value. Facilitating the manual input is achieved using the entry box function from the Tkinter module. The code snippet in listing 10 is used to create the entry box and to retrieve the value. This retrieved value is compared to the obtained value from the OWL file in step 7. Conclusions are presented using labels in step 8.

```
93.         self.manualentrysv = StringVar()
94.         self.Entry_step6man = Entry(self, textvariable=self.manualentrysv)
95.


440.              self.manualentry = self.manualentrysv.get()
```

Listing 10. Retrieval of the manual entry

After the Tkinter window is closed, the module Gc is used to clear all garbage memory dumps that are created in the process. The lines of code added at the bottom of the script are presented in listing 11.

```
492.      gc.collect()
493.      del gc.garbage[:]
```

Listing 11. Deleting all garbage memory dumps.

Running the script will automatically open the GUI. Closing the GUI will perform the last section of command lines, regarding the garbage collection. The script is converted into an executable file. This will elaborated on in the next paragraph.

6.5. Creating the executable file

Several programs and scripts exist that allow a script and the modules used in the script to be converted into an executable file. This greatly increases the usability for users that are unexperienced with Python since they only have to launch the executable file. The python module Py2Exe is used to convert the script into a single installation folder with an executable file. All modules are stored in the installation folder, allowing the script to be executed without having to install additional software. For ease of operation, it is advised that the installation folder is stored anywhere on the computer system, and a shortcut is created to the executable file. Launching this file will run the script and launch the GUI.

6.6. Redesign of the SCB process at Rijkswaterstaat

The current SCB process at Rijkswaterstaat is redesigned in this paragraph in order to implement the proposed method that uses the designed tool in order to perform the audit. This paragraph will present the exact steps which have to be taken in both the primary scenario in which an auditor uses the tool to perform an audit, and the secondary scenario, in which an asset manager uses to tool to check the value of an object in the asset data base.

First, a brief summary of the current SCB process at Rijkswaterstaat is presented. A more in-depth description of the current SCB process at Rijkswaterstaat is presented in chapter 5.2 Thereafter, the redesigned SCB process is presented. This redesigned SCB process is designed specifically for the audit ‘minimal thickness of pavement slab’, but can be used as a blue print for similar audits.

In the current SCB process at Rijkswaterstaat, the auditor receives a notification in his agenda in the UR-SCB environment that an audit has been planned regarding the minimal thickness of a specific pavement slab. The auditor retrieves the audit XLS spreadsheet file from the UR-SCB environment in order to acquire the required value for the audit. The auditor then performs the audit manually on the BIM data that has been delivered by a contractor. This BIM data is stored in a COINS container. The auditor has to open the COINS container and manually search for the correct object, property type and corresponding property value. After forming his conclusion on the audit, the auditor fills in the audit outcome and corresponding value in the XLS spreadsheet file and uploads it into the UR-SCB environment where the audit will be further processed according to the SCB process design.

The redesigned SCB process for performing audits at Rijkswaterstaat strives to automate as many steps as currently possible in the audit process at Rijkswaterstaat. Two main steps that cannot yet be easily automated are the retrieval of the XLS spreadsheet file from the UR-SCB environment and the delivery of audit outcome and evidence in the UR-SCB environment. This results in the three main process sections which the auditor has to take, as is presented below in figure 18. These steps show where the redesigned process fits in the current process.

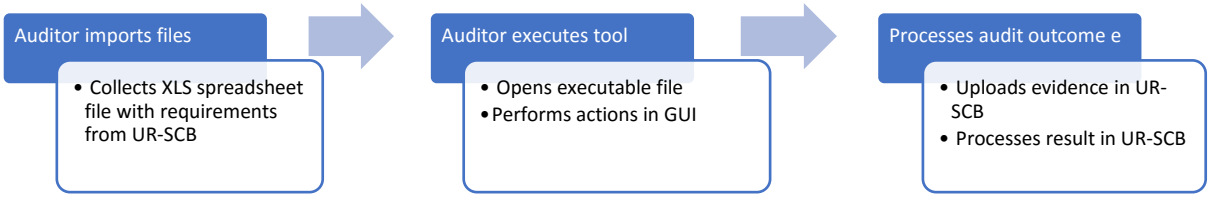


Figure 18. Redesigned SCB process for scenario 1.

The actions that the auditor performs are presented in a full process map in Appendix 4. This process map includes the user steps which the auditor has to take when using the tool to perform the audit. A full list of steps which the auditor has to take is presented below in order to create a clear overview of auditor actions. Note that first and last list of steps are steps in the current SCB process on which the redesigned process connects, as is shown in figure 16.

Standard steps in SCB process:

1. Receive a notification that an audit has been planned on a specified object in a specified COINS container.
2. Retrieve the standard XLS spreadsheet file from the Rijkswaterstaat intranet environment UR-SCB.
3. Retrieve the COINS container from the contractor.

Redesigned steps in SCB process:

4. Open the GUI of the tool by executing the tool's executable file.
5. Perform step 1 of the tool: Create or select the work folder.
6. Perform step 2 of the tool: Select the relevant COINS container.
7. Perform step 3 of the tool: Select the relevant audit type
8. Perform step 4 of the tool: Select the layer specified in the UR-SCB environment.
9. Perform step 5 of the tool: Load the object list and select the object specified in the UR-SCB environment.
10. Perform step 6 of the tool: Select the relevant XLS spreadsheet file.
11. Perform step 7 of the tool: Generate output results
12. Perform step 8 of the tool: Read audit outcome from screen and create audit output.

Standard steps in SCB process:

13. Upload the audit outcome in the UR-SCB Process and start appropriate steps regarding the audit outcome.

The steps for the secondary functionality that has been added to the tool have also been included below in this paragraph. These steps relate to the second user scenario 2, which is the scenario of an asset manager who wants to check current BIM models in an asset data base on a known value. This might occur when a minimum requirement of a certain object is updated and a check has to be performed whether the object still complies with the updated requirement. Note that again, first and last list of steps are steps in the current Asset management processes on which the redesigned process connects, as is shown in figure 19.

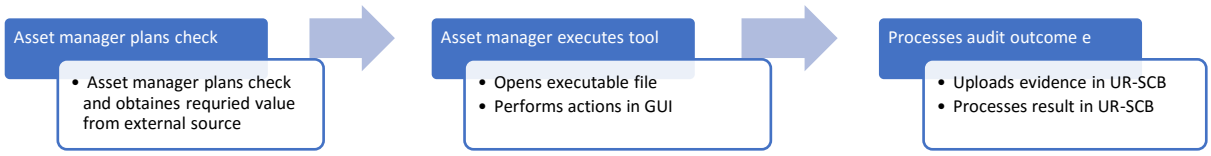


Figure 19. Redesigned SCB process for scenario 2.

The actions that the asset manager performs are presented in a full process map in the Appendix 5 as well. The steps in the user path of the asset manager are presented below:

Steps in the current asset management process:

1. Decision to check asset based on new requirement
2. Collect latest version of BIM model in the COINS container format.

Steps in the redesigned asset management process:

3. Open the GUI of the tool by executing the tool's executable file.
4. Perform step 1 of the tool: Create or select the work folder.
5. Perform step 2 of the tool: Select the relevant COINS container.
6. Perform step 3 of the tool: Select the relevant audit type
7. Perform step 4 of the tool: Select the required layer.
8. Perform step 5 of the tool: Load the object list and select the relevant object.
9. Perform step 6 of the tool: Select and enter the required value.
10. Perform step 7 of the tool: Generate output results
11. Perform step 8 of the tool: Interpret output results

Steps in the current asset management process:

12. Process outcome in the relevant asset management processes.

6.7. Results

The resulting script and redesigned process together form a new method for audits on BIM stored in COINS containers at Rijkswaterstaat. The proposed automated method brings several benefits when compared to the current manual method. The benefits identified in the design and construction phase of the tool are:

- Reduced process time, since the majority of actions are performed automatically and therefore instantly.
- Reduced risk for interpretation errors, the manual approach is more prone to errors since the read-out of the data in the COINS container is now automated.
- Reduced fundamental knowledge on how a COINS container and its contents are constructed is required for the auditor. The auditor is only required to have background knowledge on the properties of the audit and relevant values, instead of also having to know how to extract this data from a relatively new type of data storage.
- Improved implementation of working with BIM and BIM related tools and therefore promotion of the use of BIM and related tools. This is an extra benefit since Rijkswaterstaat strives to the increase the working with BIM at Rijkswaterstaat and its partners.

The tool and corresponding script and GUI have been discussed in this chapter. The practicality and discussion on the functionality of the tool in the field are presented in the following chapters.

PART D: Method review, conclusion and discussion.

7. Proposed method review and discussion

The proposed method is reviewed and discussed in this chapter. First, the applicability of the proposed method is discussed in paragraph 8.1. Limitations are summarized and discussed in paragraph 8.2. Thereafter, feedback from interviews that are conducted at Rijkswaterstaat are presented in paragraph 8.3.

7.1. Applicability of the proposed method

The proposed method for audits is applicable in the current SCB processes at Rijkswaterstaat due to the fact that it allows an auditor to perform the audit while hereby attaining multiple new benefits when compared to the current method. The current version of the proposed method is considered to be better than the current method as a direct result of the new benefits, ease of operation and user friendliness.

The optimum use for the output of this research is to use the designed method and corresponding script as a prototype update for the current manual audits at Rijkswaterstaat. A software developer can use this prototype tool to create a complete tool or set of tools which provides all required functionalities to perform all audits and create the required data output with as few manual actions required.

The method design is based on the use of a Python and SparQL script in order to automate most of the manual steps of the audit procedure within the SCB processes at Rijkswaterstaat. The method has been proven useful for audits that check on minimum or maximum requirements for a value that corresponds to a specific property of a specific object from an OWL file that is stored in a COINS container. However, this is just a single type of audit at Rijkswaterstaat. A notable variety of types of audits which Rijkswaterstaat performs on a contractors work is identified during the research. More research is required to identify all possible types of audits and to design methods that automate the testing of the BIM on the relevant requirements.

In this research, a blueprint is created for a comprehensive tool which allows auditors at Rijkswaterstaat to perform multiple types of audits on COINS containers. Currently, all code for all types of audits are hard-coded into one single script. This resulted in a script of notable size which brings disadvantage for the clarity of the script and might lead to longer computation time when executing the file.

7.2. Solutions for the limitations

In chapter 6, several limitations have been identified the use of a Python script that utilizes a SparQL query in order to perform automated audits on a BIM stored in a COINS container. These limitations exist for the proposed method because of the limited functionalities of the programming languages that are chosen and the diversity of types of data and property types in the COINS container.

The limitation of not having the functionality to calculate the angle of an embankment of which the geometric data is stored in the COINS file can be solved by importing the IFC file from the COINS container into an IFC capable software program with a geometric engine. An

example of such a program with a geometric engine is Solibiri. Another solution is to convert geometric data in the IFC file into RDF or OWL data. However, as is discovered in the literary study, more research and development is needed for current converters to be able to handle geometric data.

A second limitation of the proposed method that identified in this research is not as easy to solve. This limitation is not having the functionality to determine whether the height clearance of a bridge over a road was high enough. A solution might lay in a method which creates a surface perpendicular to the road, of which the height is equal to the minimum height clearance that is allowed by the requirement. Such a surface is visualized in figure 20. A software program with a geometrics engine and clash detection functionalities, could use the clash detection to identify whether the height clearance requirements are met.

A third limitation of the proposed method, and any current development regarding automated audits, are the current obstacles in code checking compliance as is identified in chapter 3.5 of this report. Requirements are not yet stored in Semantic Web databases due to the high number of possible alternatives and performance attributes checks in the construction industry. In order to overcome this issue, a solution could be to create a national standard to create ontologies for standard rule sets. An even better solution would be to create an international standard. The problem could theoretically then be overcome by using Semantic Web Technology.

A fourth limitation of the current version of the tool and corresponding script is that the code is written in a single file. This won't allow the script to be extended to include more lines of code to create the ability to solve multiple types of audit without the file becoming unnecessary long, resulting in higher computation times and lower clarity for future developers. The solution for this problem is modular script writing. All functionalities from the script could be written in modules of the script, rather than in the single file. This would allow the script to call upon these modules to facilitate the functionalities of the script. Using this method of script writing, only a small section of the script would be required to be executed at the same time.

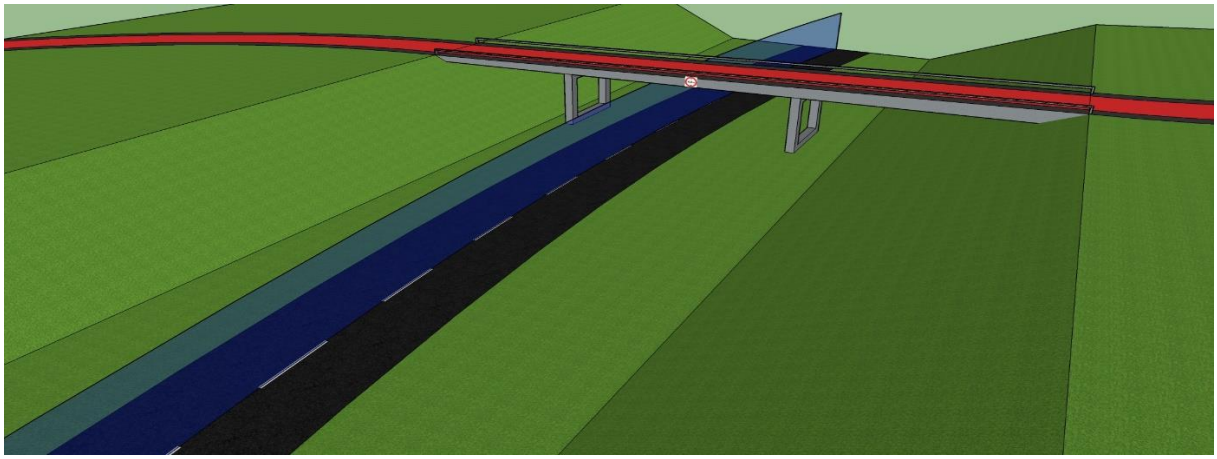


Figure 20. Perpendicular surface added to the road to facilitate clash detection

A fifth limitation of the script consist of the lack of a functionality to visualize geographic data related to the selected object. This visualization would allow the user to perform an additional visual check on the selected object in order to reduce the chance on selection errors. By adding a geometry visualizer to the tool, this functionality could be facilitated. Multiple types of geometric data are included in COINS containers. An example for the proposed method for the audit 'Thickness of pavement' is the possibility to visualize the GML data that is stored in the COINS container and linked to the objects. Currently, no GML viewer is available that can be easily included in the script. This functionality is therefore not included in the current version of the proposed method.

7.3. Feedback from interviews

Five people have been interviewed at Rijkswaterstaat, who are linked with audits, the development of software at Rijkswaterstaat, and the implementation of BIM at Rijkswaterstaat. These interviews were open interviews were the proposed method was discussed in order to gain feedback from the interviewees. The conclusions from these open interviews are presented in this section.

The current SCB process has BIM implemented. However, BIM is not yet fully implemented within the entire SCB process. Performing audits is considered to as a manual task. Automatization of these audits and the further implementation of BIM in these audits is therefore desired, in order to acquire more benefits from the BIM systematic.

When regarding the method proposed in this thesis, the improvements and the potential is clear. However, the method solely focusses on product audits. System audits and process audits are outside of the scope of this research. System audits and process audits are also not yet implemented in BIM at Rijkswaterstaat. There is therefore still a lot of room for BIM related improvement when regarding the SCB process. It is also noted that it is still a point of discussion whether BIM can help the SCB process in system and process related audits.

The potential for this script for asset managers at Rijkswaterstaat is also acknowledged in the open interviews. Asset managers regularly test their own BIM. This includes testing COINS containers on old and revised requirements. It is acknowledged that the proposed method can therefore also have value for asset managers at Rijkswaterstaat. It is stated that the proposed method might carry even more value for the asset managers than the auditors at Rijkswaterstaat. The proposed method could become part of a 'toolbox of tools' which could help both auditors and asset managers to validate BIM.

In the interviews it is stated that, although the process can be heavily automated, auditors will always be required as human interpreter in order to control the BIM and guide the BIM related actors in the process.

7.4. Discussion

The research in this report researched whether the SCB process at Rijkswaterstaat could benefit from a more thorough implementation of working with BIM. The design of the new method of performing automated audits on BIM proved that the current method of manually performing audits can profit from BIM related benefits. The time required for the process has been reduced and the required knowledge of an auditor on how data is stored in OWL files

that are stored in COINS container has been lowered significantly. The method can be used as a prototype to improve more audits in a similar fashion.

The method proved that an auditor can perform an audit on data in a COINS container, without having much knowledge on how data is stored in a COINS container. The required know-how and effort for achieving the benefits of BIM is therefore lowered. This greatly increases the potential of this research on being a promotion tool for the further implementation of BIM at Rijkswaterstaat because it proves that methods can be developed that aid users in working with BIM.

The proposed method is designed, constructed and tested for one specific type of audit only. This is the type of audit where the required data is stored in an OWL file within the COINS container. Multiple types of audits can be identified at Rijkswaterstaat and further research is required in order to identify and achieve more possible SCB process enhancements by further implementing BIM at Rijkswaterstaat.

The proposed method of using a script to query for data is strongly dependent on the data which the contractor provides. It is therefore of high importance that the data is delivered in a prescribed manner. This could be achieved by implementing an Information Delivery Manual (IDM) in the contract between the contractor and Rijkswaterstaat.

Limitations of the designed method that uses a script to query for data from the COINS containers are identified as well within this research in chapter 5.4. Possible solutions to these problems have been pointed out, but further research is required in order to determine the optimum solutions for these limitations.

Code checking compliance is still a barrier when trying to automate certain other types of audits. These are the types of audits that cannot easily be formalized into semantic code that can be processed by a computer. This is often a result of the audit being open to interpretation or very case specific. However, the literary study showed that progress in code checking compliance is continuously being made.

The currently used method of researching only includes the possibility of using queries to perform data analysis. There are almost no negative side effects for this method because the method is efficient, not prone to user interpretation errors due to the fact that the process is automatized, and easy to further develop due to the fact that only open source software is used in the development of the tool. However, no other method has been researched and compared to this method. This is a limitation of this research.

Another limitation of this research is the fact that no extend testing phase of the tool on multiple auditors with multiple COINS containers has taken place is also a limitation of this research. This is a result of the fact that only one COINS container from a pilot case by Rijkswaterstaat was available.

8. Conclusion

In this research, a method has been developed to automatically perform an audit by checking a Building Information Model (BIM) on requirements set by Rijkswaterstaat. This method included the development and design of a tool which uses the programming language Python and the query language SparQL to extract a specific value that is stored in a COINS container in order to perform the audit. By using the proposed method, the auditor no longer has to have extend knowledge on the data structures of OWL files that are stored in a COINS containers in order to preform audits on these COINS containers. This improvement of an employees' task provides Rijkswaterstaat with a promotion tool in order to further promote the implementation of BIM at Rijkswaterstaat.

By implementing the developed method in the SCB processes at Rijkswaterstaat, it has been proven that the further implementation of BIM in processes at Rijkswaterstaat is possible and more benefits of BIM can hereby be attained.

From analysing the potential of BIM in the literary research it has been concluded that the implementation of BIM in construction processes can bring many benefits for the construction industry. In order to further implement BIM at Rijkswaterstaat, a method of performing automated audit methods at Rijkswaterstaat by using queries for semantic data has been designed. Three use case scenarios have been created and analysed in order to define the required functionalities for the new method. These functionalities have been identified and analysed in order to create the theoretical foundations for the design of the new method.

The three use case scenarios also backed the identification of the limitations of the proposed method of using queries in order to improve audit processes. These limitations were not identified in the literary research prior to the research and are therefore analysed. These limitations considered of the lack of functionality to generate new geometric data from IFC files and perform audits based on this geometric data. Alternative methods and developments have been identified and proposed as solutions to these problems.

The proposed method of automating the audit processes on data stored in an OWL file in a COINS container has been developed based on the theoretical foundations that are discovered in the analysis of the use case scenarios. A script has been written that uses the Python and SparQL programming languages in order to select the object and related data from the COINS container through a user-friendly graphical user interface. In the script, this data is compare with requirements that are stored in the standard online environment at Rijkswaterstaat. The standard online environment is called the UR-SCB. The script and method proved to work and a certain type of audit can now automatically be checked by using the script and method.

The main research question of this thesis is 'will the implementation of BIM improve existing processes at DBFM projects for contractors and clients such as Rijkswaterstaat?' The design of the method and analysis of the results proves that the implementation of BIM will create benefits for the SCB process and therefore for contractors and Rijkswaterstaat. These benefits are identified as a reduction in required time and labour force for performing audits, a lower

risk of errors in data selection when performing audits, and a lower required level of knowledge on how COINS containers and OWL files in COINS containers are constructed.

The second research question of this thesis is 'what are the benefits of using a script in order to perform verification and validation of BIM?' These benefits have been identified and analysed in this research. The use of scripts allows a developer to create software tools that automate audits which require data retrieval from OWL files in COINS containers. The use of a script hereby enables an auditor to perform audits without having an extend knowledge on the data structure of COINS Containers. It also reduces the time required for an Audit, as well las the labour intensity.

The third research question of this thesis is 'what are the limitations of using programming languages in order to perform verification and validation of data files? These limitations have been identified and analysed in this research. These limitations arise due to the lack of functionalities of the scripts that allow for geometric data calculations. The use of scripts offer no functionality for when geometric data has to be checked, such as for example when testing the angle of an embankment.

Area for further research exists since many different types of audits exist at Rijkswaterstaat and similar companies. These audits consist of a great variety of tests, since there are many aspects to construction projects as can be seen from the many different types of data stored in a COINS container. These types of data storage that differ from the RDF and OWL data files require new methods in order to extract and test data. These methods have yet to be identified, analysed and optimized.

In this research it has been proven that audit processes at Rijkswaterstaat can be improved by extending the implementation of BIM. Rijkswaterstaat is able to use this research to further promote the implantation of BIM at Rijkswaterstaat and its contractors. The results of this research can also be used as a prototype for similar problems. The script is also useable for asset managers to check the validity of their BIM models. BIM is constantly being further researched and developed, and new ways of improving data management and communication in construction processes is hereby being discovered. This thesis proved that the benefits of BIM are accessible when actively implementing BIM in standard construction processes.

The last research question is: 'What should be the focus of further research to improve the working with BIM in verification and validation processes?' It is concluded in this research that it is possible and beneficial to perform audits through automated processes. However, the solution provided is specifically designed for a single type of audit. This limitation is a result of the great variety in available types of audit and how these audits are communicated. The focus of future research should be aimed at the development of a standard for the automation of audits. The development of this standard would allow audits to be interoperable with BIM systems, which would greatly reduce the required time for the audits and make the audits less prone to errors. This standard could be developed in the form of an ontology specifically designed for audits, or a standard file extension that is specifically designed for audits.

References

Abanda, F. H., Tah, J. H. M., & Keivani, R. (2013). Trends in built environment semantic Web applications: Where are we today? *Expert Systems with Applications*, *40*(14), 5563–5577. http://doi.org/10.1016/j.eswa.2013.04.027

Allemang, D., & Hendler, J. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. *Semantic Web for the Working Ontologist* (Second Edi). Waltham: Elsevier. http://doi.org/10.1016/B978-0-12-385965-5.10001-9

Basu, R. (2004). Implementing quality: a practical guide to tools and techniques: enabling the power of operational excellence. Retrieved from https://books.google.nl/books?hl=nl&lr=&id=JHdT8rF4GCwC&oi=fnd&pg=PR16&dq=+Implementing+Quality+%E2%80%93+A+Practical+Guide+to+Tools+and+Techniques,+Thomson+Learnin&ots=pAWg9GRfgO&sig=Jh-3RH9tN0tJ3BqEEqMNI7liNhI

Bazjanac, V., & Crawley, D. (1997). The implementation of industry foundation classes in simulation tools for the building industry. *Lawrence Berkeley National …*. Retrieved from http://escholarship.org/uc/item/76c6z6g4.pdf

Bazjanac, V., & Crawley, D. (1999). Industry foundation classes and interoperable commercial software in support of design of energy-efficient buildings. *Proceedings of Building Simulation'99*. Retrieved from http://www.inive.org/members_area/medias/pdf/Inive/IBPSA/UFSC755.pdf

Bechhofer, S. (2009). OWL: Web ontology language. *Encyclopedia of Database Systems*. Retrieved from http://link.springer.com/10.1007/978-0-387-39940-9_1073

Beckett, D., & McBride, B. (2004). RDF/XML syntax specification (revised). *W3C Recommendation*. Retrieved from http://badjoke.demic.eu/papers/liris.cnrs.fr/alain.mille/enseignements/Ecole_Centrale/projets_2004/RDF.pdf

Beetz, J. (2009a). Facilitating distributed collaboration in the AEC/FM sector using Semantic Web Technologies. *Eindhoven: Technische Universiteit Eindhoven*. Retrieved from https://pure.tue.nl/ws/files/2966330/200911977.pdf

Beetz, J. (2009b). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Retrieved from http://journals.cambridge.org/abstract_S0890060409000122

Beetz, J., Zhang, C., & Weise, M. (2015). Interoperable validation for IFC building models using open standards, *20*(November 2014), 24–39.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*. Retrieved from https://scholar.google.nl/scholar?hl=nl&q=Berners-Lee%2C+hendler+lassila+the+semantic+web+scientific+america&btnG=&lr=#0

Bew, M., & Richards, M. (2008). Bew-Richards BIM Maturity Model.

Birte, G. (2011). *Reasoning Web. Semantic Technologies for the Web of Data*. (A. Polleres, C. D'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski, & P. Patel-Schneider, Eds.) (Vol. 6848). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-23032-5

Bonabeau, E. (2002). Agent-based modeling: methods and techniques for simulating human systems. *Pnas*, *99*, 7280–7287. http://doi.org/10.1073/pnas.082080899

Brous, P., Herder, P., & Janssen, M. (2015). Towards Modelling Data Infrastructures in the Asset Management Domain. *Procedia Computer Science*, *61*, 274–280.

http://doi.org/10.1016/j.procs.2015.09.215
COINS-projectgroep, R. (2015). COINSweb. Retrieved from htttps://www.coinsweb.nl
Curé, O. (2015). RDF database systems : triples storage and SPARQL query processing.
Ding, L., Zhou, Y., & Akinci, B. (2014). Building Information Modeling (BIM) application framework: The process of expanding from 3D to computable nD. *Automation in Construction*, *46*, 82–93. http://doi.org/10.1016/j.autcon.2014.04.009
Eadie, R., Browne, M., Odeyinka, H., McKeown, C., & McNiff, S. (2013). BIM implementation throughout the UK construction project lifecycle: An analysis. *Automation in Construction*, *36*, 145–151. http://doi.org/10.1016/j.autcon.2013.09.001
Eastman, C. M. (2011). *BIM handbook : a guide to building information modeling for owners, managers designers, engineers, and contractors. [eBook]*. Chichester : John Wiley & Son.
Farr, E. R. P., Piroozfar, P. A. E., & Robinson, D. (2014). BIM as a generic configurator for facilitation of customisation in the AEC industry. *Automation in Construction*, *45*, 119–125. http://doi.org/10.1016/j.autcon.2014.05.012
Grau, B., Horrocks, I., & Motik, B. (2008). OWL 2: The next step for OWL. *Web Semantics: Science …*. Retrieved from http://www.sciencedirect.com/science/article/pii/S1570826808000413
Guha, R., & Brickley, D. (2004). RDF vocabulary description language 1.0: RDF schema. *W3C Recommendation, W3C*. Retrieved from https://scholar.google.nl/scholar?q=rdf+vocabulary+description+language+brickley&btnG=&hl=nl&as_sdt=0%2C5#3
Harmelen, F. Van, & McGuinness, D. (2004). OWL web ontology language overview. *World Wide Web Consortium (…*. Retrieved from http://www.kramirez.net/SMA_Maestria/Material/Presentaciones/Exposicion Ontologias/Documentos/OWL Web Ontology Language Overview.pdf
Hetland, M. L. (2008). *Beginning Python*. Berkeley, CA: Apress. http://doi.org/10.1007/978-1-4302-0634-7
Karan, E. P., & Irizarry, J. (2015). Extending BIM interoperability to preconstruction operations using geospatial analyses and semantic web services. *Automation in Construction*, *53*, 1–12. http://doi.org/10.1016/j.autcon.2015.02.012
Klyne, G., & Carroll, J. (2006). Resource description framework (RDF): Concepts and abstract syntax. Retrieved from http://www.citeulike.org/group/2170/article/532408
Koot, W. van de. (2012, May 4). De inbedding van Bouw Informatie Modellen (BIM) in Nederlandse Bouwcontracten. Open Universiteit Nederland. Retrieved from http://dspace.ou.nl/handle/1820/4271
Lampert, D. J., & Wu, M. (2015). Development of an open-source software package for watershed modeling with the Hydrological Simulation Program in Fortran. *Environmental Modelling & Software*, *68*, 166–174. http://doi.org/10.1016/j.envsoft.2015.02.018
Lassila, O., & Swick, R. (1998). Resource Description Framework (RDF) model and syntax specification. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.6030
Lee, K., Chin, S., & Kim, J. (2003). A core system for design information management using Industry Foundation Classes. *Computer-Aided Civil and Infrastructure …*. Retrieved from http://onlinelibrary.wiley.com/doi/10.1111/1467-8667.00318/abstract
Lee, K. D. (2014). *Python Programming Fundamentals*. London: Springer London. http://doi.org/10.1007/978-1-4471-6642-9
Lu, W., Fung, A., Peng, Y., Liang, C., & Rowlinson, S. (2014). Cost-benefit analysis of Building

Information Modeling implementation in building projects through demystification of time-effort distribution curves. *Building and Environment*, *82*, 317–327. http://doi.org/10.1016/j.buildenv.2014.08.030

Mazairac, W., & Beetz, J. (2013). BIMQL–An open query language for building information models. *Advanced Engineering Informatics*. Retrieved from http://www.sciencedirect.com/science/article/pii/S1474034613000657

Miettinen, R., & Paavola, S. (2014). Beyond the BIM utopia: Approaches to the development and implementation of building information modeling. *Automation in Construction*, *43*, 84–91. http://doi.org/10.1016/j.autcon.2014.03.009

National Institute of Building Science, N. (2007). *United States: National Building Information Modeling Standard* (Version 1). Washington DC. Retrieved from http://www.wbdg.org/pdfs/NBIMSv1_p1.pdf

Nederveen, S. Van, Beheshti, R., & Willems, P. (2010). Building Information Modelling in the Netherlands: A Status Report. *W078-Special Track 18th CIB …*. Retrieved from http://www.irbnet.de/daten/iconda/CIB18798.pdf#page=33

Nour, M., & Beucke, K. (2008). An open platform for processing IFC model versions. *Tsinghua Science and Technology*, *13*(S1), 126–131. http://doi.org/10.1016/S1007-0214(08)70138-X

Pauwels, P., & Deursen, D. Van. (2011). Three-dimensional information exchange over the semantic web for the domain of architecture, engineering, and construction. *… for Engineering …*. Retrieved from http://journals.cambridge.org/abstract_S0890060411000199

Pauwels, P., Meyer, R. De, & Campenhout, J. Van. (2010). Interoperability for the design and construction industry through semantic web technology. *Semantic Multimedia*. Retrieved from http://link.springer.com/10.1007/978-3-642-23017-2_10

Pauwels, P., & Terkaj, W. (2016). EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, *63*, 100–133. http://doi.org/10.1016/j.autcon.2015.12.003

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., & Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. *Automation in Construction*, *20*(5), 506–518. http://doi.org/10.1016/j.autcon.2010.11.017

Prechelt, L. (2000). An empirical comparison of seven programming languages. *Computer*, *33*(10), 23–29. http://doi.org/10.1109/2.876288

Prud'hommeaux, E., & Seaborne, A. (2014). SPARQL Query Language for RDF, W3C Recommendation 15 January 2008. 2008. Retrieved from https://scholar.google.nl/scholar?q=SPARQL+Query+Language+for+RDF++W3C+Recommendation+15+January+2008&btnG=&hl=nl&as_sdt=0%2C5#1

Remmen, P., Cao, J., Ebertsh@auser, S., Frisch, J., Lauster, M., Maile, T., … van Treeck, C. (2015). An open framework for integrated BIM-based building performance simulations using Modelica. Retrieved January 6, 2016, from http://www.iea-annex60.org/downloads/p2384.pdf

Rijkswaterstaat. (2011). *Kader Systeemgerichte Contractbeheersing (SCB)*.

Rijkswaterstaat. (2015). BIM-producten en standaarden. Retrieved from https://www.rijkswaterstaat.nl/zakelijk/werken-aan-infrastructuur/efficienter-werken/bouwwerk-informatie-model/producten-en-standaarden.aspx

Rijkswaterstaat. (2016). Rijkswaterstaat. Retrieved January 3, 2016, from

http://www.rijkswaterstaat.nl/
Rokooei, S. (2015). Building Information Modeling in Project Management: Necessities, Challenges and Outcomes. *Procedia - Social and Behavioral Sciences*, *210*, 87–95. http://doi.org/10.1016/j.sbspro.2015.11.332
Rolland, C., & Achour, C. Ben. (1998). Guiding the construction of textual use case specifications. *Data & Knowledge Engineering*, *25*(1-2), 125–160. http://doi.org/10.1016/S0169-023X(97)86223-4
Schevers, H., & Drogemuller, R. (2005). Converting the industry foundation classes to the web ontology language. *… and Grid, 2005. SKG'05. First …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4125861
Schmitz, O., Karssenberg, D., de Jong, K., de Kok, J.-L., & de Jong, S. M. (2013). Map algebra and model algebra for integrated model building. *Environmental Modelling & Software*, *48*, 113–128. http://doi.org/10.1016/j.envsoft.2013.06.009
Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The Semantic Web Revisited. *IEEE Intelligent Systems*, *21*(3), 96–101. http://doi.org/10.1109/MIS.2006.62
Tan, X., Hammad, A., & Fazio, P. (2010). Automated Code Compliance Checking for Building Envelope Design. *Journal of Computing in Civil Engineering*, *24*(2), 203–211. http://doi.org/10.1061/(ASCE)0887-3801(2010)24:2(203)
Techentin, R., & Gilbert, B. (2014). Implementing Iterative Algorithms with SPARQL. *EDBT/ICDT …*. Retrieved from http://www.researchgate.net/profile/Adam_Lugowski/publication/261923656_Implementing_Iterative_Algorithms_with_SPARQL/links/0f317535f54b8d336c000000.pdf
Torma, S. (2013). Semantic linking of building information models. *Semantic Computing (ICSC), 2013 IEEE Seventh …*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6693555
Venugopal, M., Eastman, C. M., Sacks, R., & Teizer, J. (2012). Semantics of model views for information exchanges using the industry foundation class schema. *Advanced Engineering Informatics*, *26*(2), 411–428. http://doi.org/10.1016/j.aei.2012.01.005
Volk, R., Stengel, J., & Schultmann, F. (2014). Building Information Modeling (BIM) for existing buildings — Literature review and future needs. *Automation in Construction*, *38*, 109–127. http://doi.org/10.1016/j.autcon.2013.10.023
Vollebregt, E. A. H., Roest, M. R. T., & Lander, J. W. M. (2003). Large scale computing at Rijkswaterstaat. *Parallel Computing*, *29*(1), 1–20. http://doi.org/10.1016/S0167-8191(02)00217-X
Wood, D., Gearon, P., & Adams, T. (2005). Kowari: A platform for semantic web storage and analysis. *XTech 2005 Conference*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.9427&rep=rep1&type=pdf
World Wide Web Consortium. (2004). OWL Web Ontology Language Use Cases and Requirements. Retrieved January 14, 2016, from https://www.w3.org/TR/webont-req/
World Wide Web Consortium. (2009). OWL 2 Web Ontology Language Primer. Retrieved January 14, 2016, from https://www.w3.org/TR/2009/REC-owl2-primer-20091027/
Yalcinkaya, M., & Singh, V. (2015). Patterns and trends in Building Information Modeling (BIM) research: A Latent Semantic Analysis. *Automation in Construction*, *59*, 68–80. http://doi.org/10.1016/j.autcon.2015.07.012
Yu, L. (2014). *A Developer's Guide to the Semantic Web*. Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-662-43796-4

Zhang, Y. (2015). *An Introduction to Python and Computer Programming* (Vol. 353). Singapore: Springer Singapore. http://doi.org/10.1007/978-981-287-609-6

Zhong, B. T., Ding, L. Y., Luo, H. B., Zhou, Y., Hu, Y. Z., & Hu, H. M. (2012). Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking. *Automation in Construction*, *28*, 58–70. http://doi.org/10.1016/j.autcon.2012.06.006

Appendices

Appendix 1: RDF triples explained

In order to elaborate on the methodology of the RDF and RDFS, the example used in paragraph 4.1 of this report will be elaborated. RDF uses triples to store semantic information. In the example of 'Bob knows Fred', the subject 'Bob', has a predicate 'knows someone', and an object, 'Fred'. This type of data relation is known as a triple. The triple of 'Bob knows Fred' is illustrated below in figure 2.

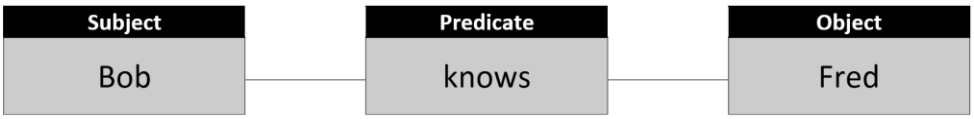


Figure 18 Example of a RDF triple.

The predicate can be regarded as the type of relation between the object and the subject. One can replace the word 'predicate' with 'type of relation'. This type of data storage has the advantage that programming languages can be used to create so-called 'implied data'. Implied data is data that is not stored specifically in the data storage, but can be deducted from the data file.

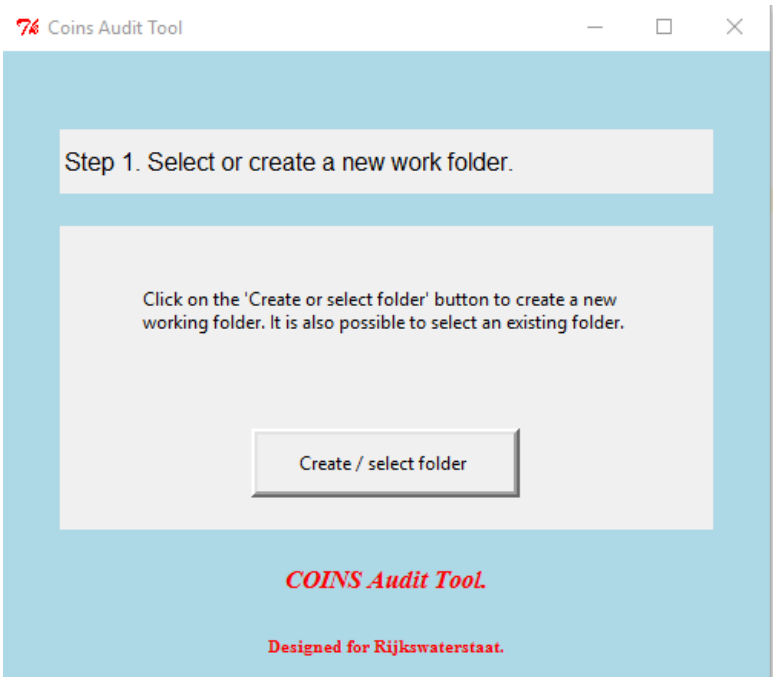
Regard the RDF data in table 1 below:

Subject	Predicate	Object
Bob	knows	Fred
Bob	knows	Anna

Table 1. Example of triples in a RDF based data set

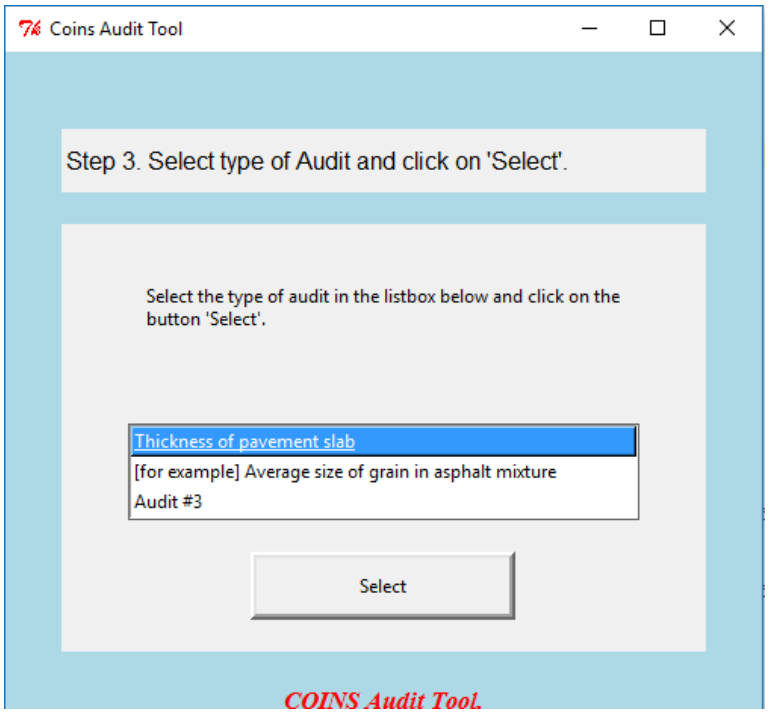
In the data file it is specifically stated that 'Bob knows Fred', and 'Bob knows Anna'. There is no specific data about the relation between 'Fred' and 'Anna' in the data file. However, with this data, a relation between 'Fred' and 'Anna' can be implied. This would be the predicate: 'knows someone who knows'. Although it is not specifically included in the data file, it can be deducted that: 'Fred knows someone who knows Anna', and 'Anna knows someone who knows Fred'.

Appendix 2: The Graphical User Interface

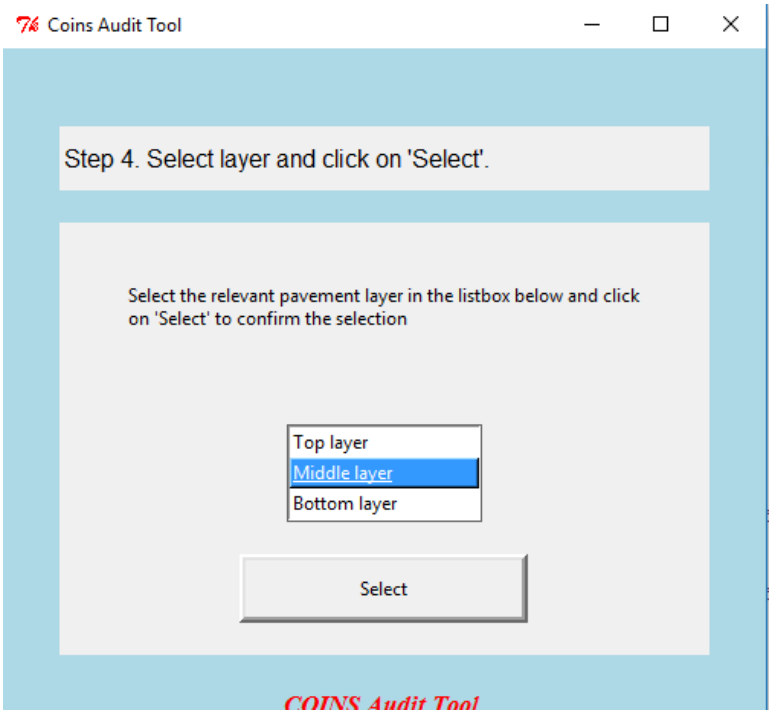


GUI window for step 1

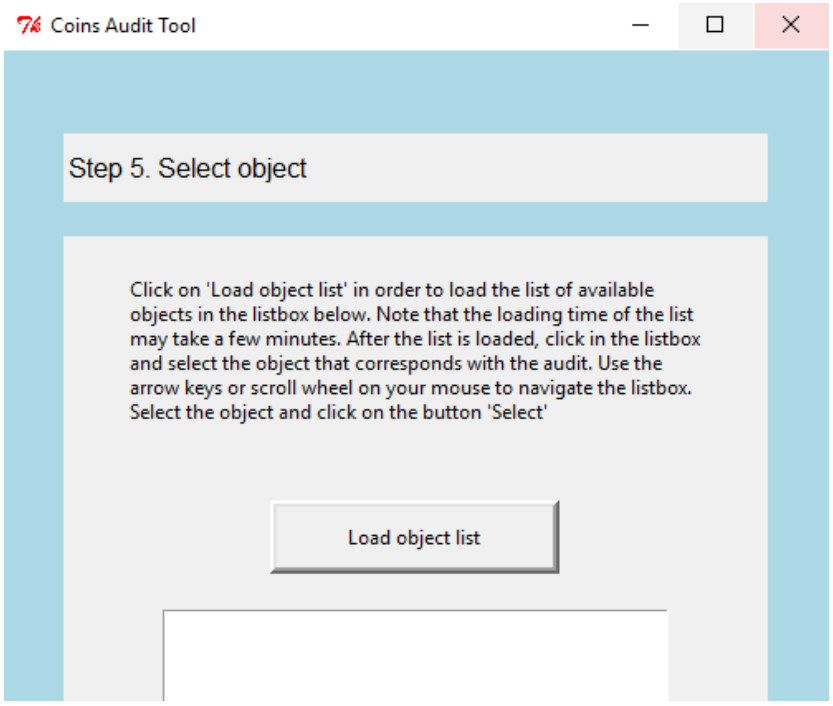
GUI window for step 2



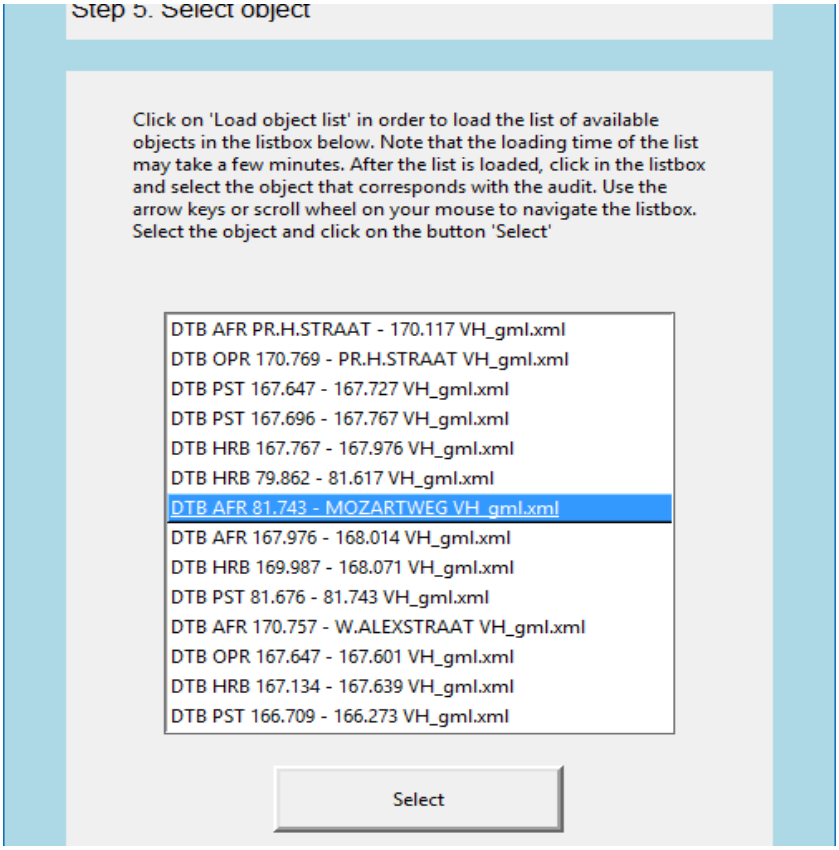
GUI window for step 3



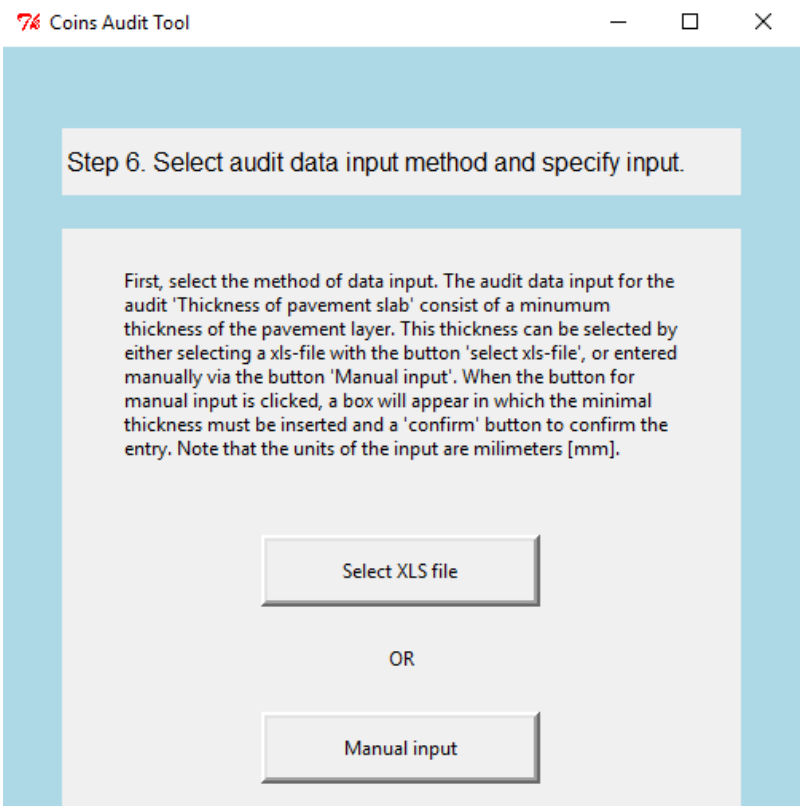
GUI window for step 4



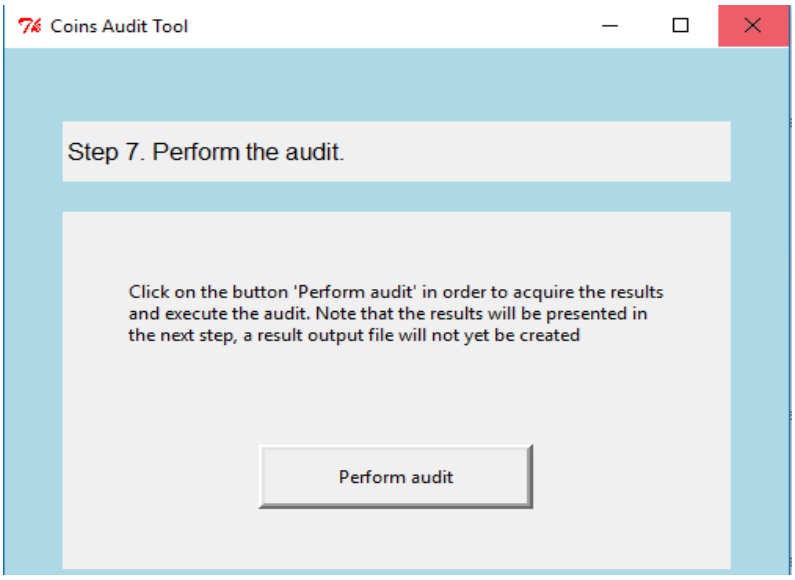
GUI window for step 5



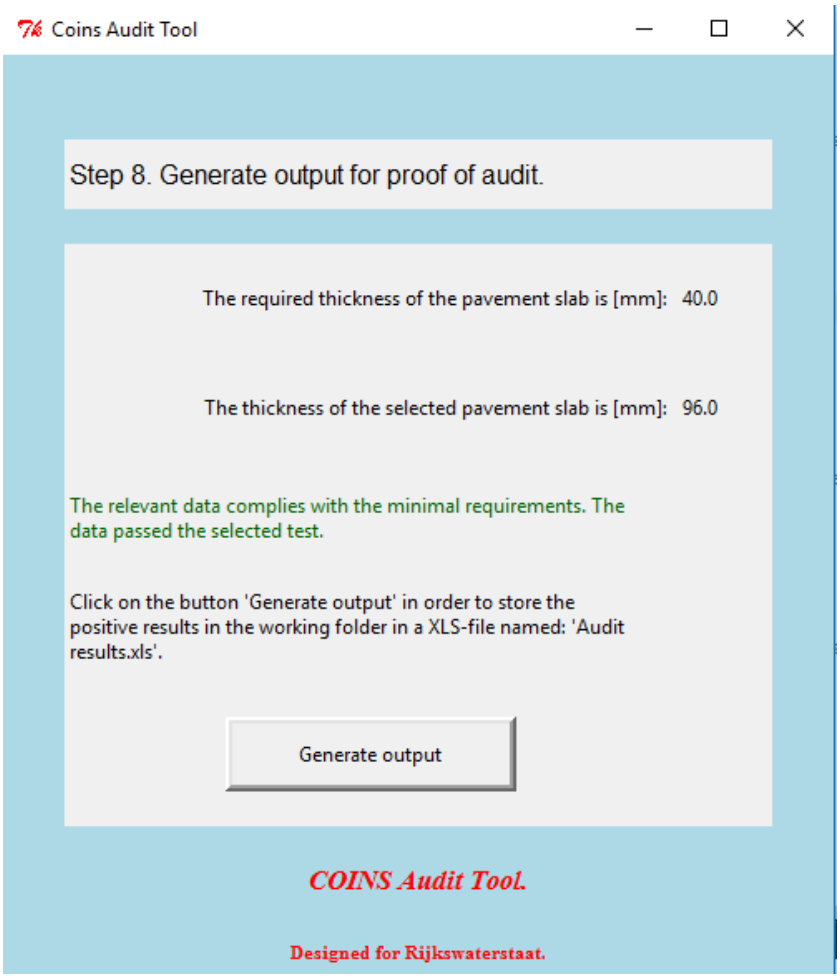
GUI window for step 5 with loaded object list



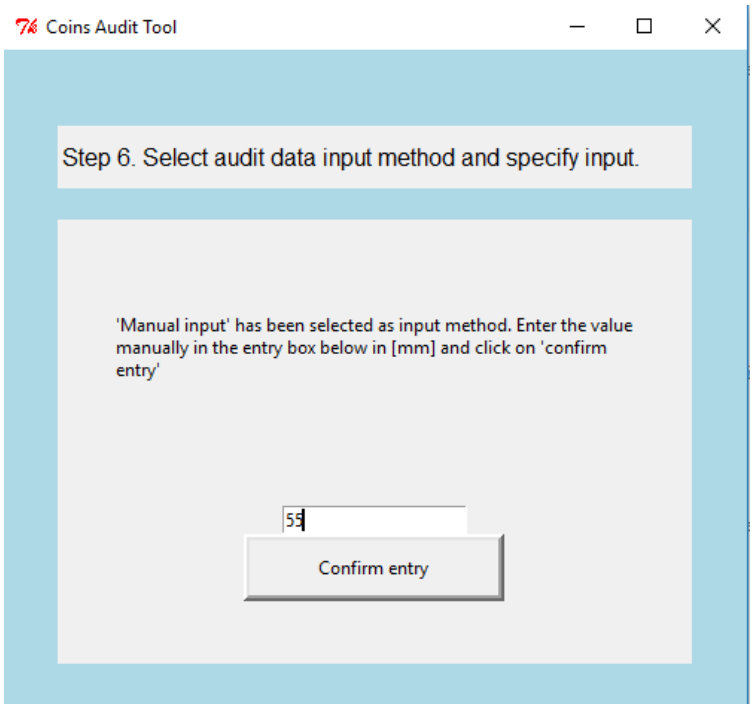
GUI window for step 6



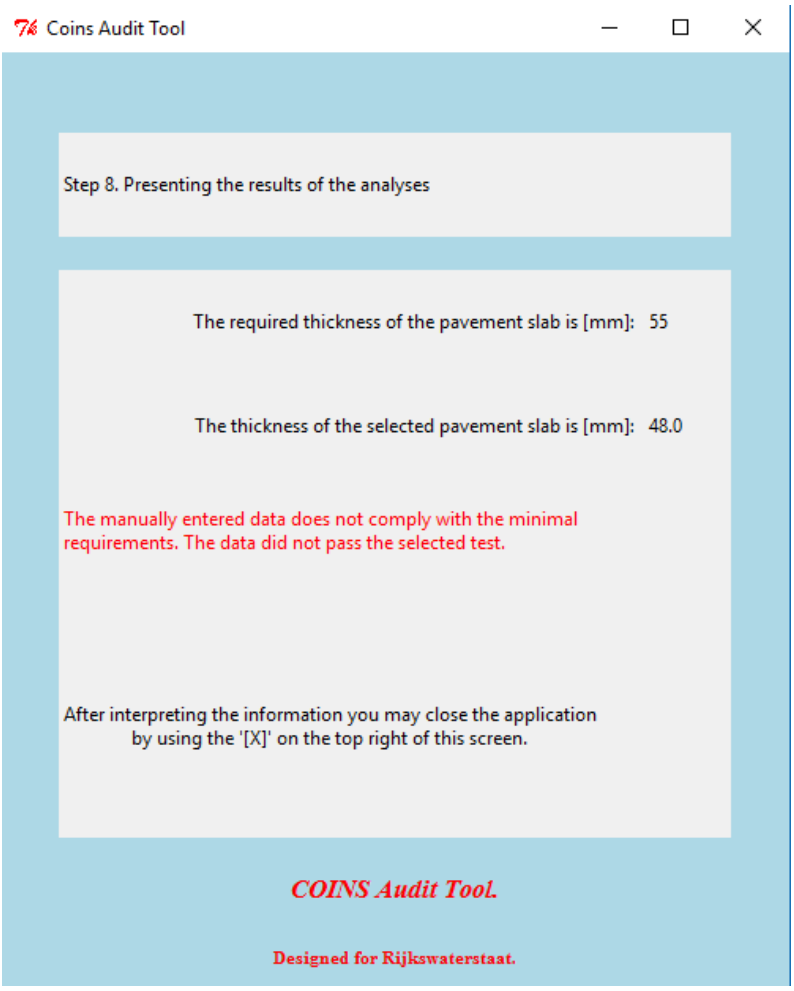
GUI window for step 7



GUI window for step 8 with ‘XLS-file’ as data input



GUI window for step 6 with manual entry as data input



GUI window for step 8 with manual entry as data input

Appendix 3: The script

This appendix contains the script which, when executed, opens the GUI and contains all command lines for the tool. The '#' icon is used to indicate an explanatory sentence.

```
1.      # importing all modules that are used in the code
2.
3.      import Tkinter
4.      import tkFileDialog
5.      import zipfile
6.      from Tkinter import *
7.      import rdflib
8.      from rdflib import Namespace
9.      import os
10.     import xlrd
11.     import gc
12.     from xlrd import open_workbook
13.     from xlutils.copy import copy
14.     import webbrowser
15.
16.         # Creating  a class in which the Graphic User Interface (GUI) will be written by
        use of tKinter.
17.
18.     class TkFileDialogExample(Tkinter.Frame):
19.
20.       def __init__(self, root):
21.
22.         # Creating the frameework and main configurations
23.
24.         Tkinter.Frame.__init__(self, root)
25.         root.configure(background = 'lightblue')
26.         root.wm_title("Coins Audit Tool")
27.         root.geometry('{}x{}'.format(500, 750))
28.
29.         # Creating the main labels, listboxes, entry boxes, and buttons used in the GUI,
        divided into one step per window.
30.
31.         self.Label_step1 = Label(self, height=2, font=11, text="Step 1. Select or create
        a new work folder.")
32.         self.Label_step1.grid(row=1, column=0, sticky=W)
33.         self.Label_message1 = Label(self, height=7, wraplength=350, justify=LEFT, text="
        Click on the 'Create or select folder' button to create a new working folder. It is
        also possible to select an existing folder. ")
34.         self.Label_message1.grid(row=3, column=0, sticky=N)
35.         self.Button_step1 = Button(self, width=23, height=2, borderwidth=4, text="Create
        / select folder ", command=self.createfolder)
36.         self.Button_step1.grid(row=5, column=0, sticky=N)
37.
38.         self.Label_step2 = Label(self, height=2, font=11, text="Step 2. Select the COINS
        container.")
39.         self.Label_message2 = Label(self, height=7, wraplength=350, justify=LEFT, text="
        Click on 'Choose COINS container' to select the COINS container which contains the d
        ata that requires to be tested. This COINS container will automatically be copied an
        d unpacked in the work folder created or selected in step 1. The extention of the CO
        INS container is: '.ccr'.")
40.         self.Button_step2 = Button(self, width=23, height=2, borderwidth=4, text="Choose
        COINS container", command=self.selectCOINSc)
41.
42.         self.Label_step3 = Label(self, height=2, font=11, text="Step 3. Select type of A
        udit and click on 'Select'.")
43.         self.Label_message3 = Label(self, height=7, wraplength=350, justify=LEFT, text="
        Select the type of audit in the listbox below and click on the button 'Select'.")
```

```
44.         self.List_1 = Listbox(self, height=3, width=55, selectborderwidth=2, selectmode=
        'single', exportselection=0)
45.         self.List_1.insert(1, 'Thickness of pavement slab')
46.         self.List_1.insert(2, '[for example] Average size of grain in asphalt mixture')

47.         self.List_1.insert(3, 'Audit #3')
48.         self.Button_step3 = Button(self, text="Select", width=23, height=2, borderwidth=
        4, command=self.AuditSelection)
49.
50.         self.Label_step4 = Label(self, height=2, font=11, text="Step 4. Select layer and
         click on 'Select'.")
51.         self.Label_message4 = Label(self, height=7, wraplength=350, justify=LEFT, text="
        Select the relevant pavement layer in the listbox below and click on 'Select' to con
        firm the selection")
52.         self.List_2 = Listbox(self, height=3, selectmode='single', selectborderwidth=2,
        exportselection=0)
53.         self.List_2.insert(1, 'Top layer')
54.         self.List_2.insert(2, 'Middle layer')
55.         self.List_2.insert(3, 'Bottom layer')
56.         self.Button_step4 = Button(self, text="Select", width=25, height=2, borderwidth=
        4, command=self.SelectLayer)
57.
58.         self.Label_step5 = Label(self, height=2, font=11, text="Step 5. Select object")

59.         self.Label_message5 = Label(self, height=9, wraplength=350, justify=LEFT, text="
        Click on 'Load object list' in order to load the list of available objects in the li
        stbox below. Note that the loading time of the list may take a few minutes. After th
        e list is loaded, click in the listbox and select the object that corresponds with t
        he audit. Use the arrow keys or scroll wheel on your mouse to navigate the listbox.
        Select the object and click on the button 'Select'    ")
60.         self.Button_step5a = Button(self, text="Load object list", width=23, height=2, b
        orderwidth=4, command=self.populateListbox)
61.         self.List_3 = Listbox(self, height=14, width=50, selectborderwidth=2, selectmode
        ='single', exportselection=0)
62.         self.Button_step5b = Button(self, text="Select", width=23, height=2, borderwidth
        =4, command=self.selectobject)
63.
64.         self.Label_step6 = Label(self, height=2, font=11, text="Step 6. Select audit dat
        a input method and specify input.")
65.         self.Label_message6 = Label(self, height=11, wraplength=350, justify=LEFT, text=
        "First, select the method of data input. The audit data input for the audit 'Thickne
        ss of pavement slab' consist of a minumum thickness of the pavement layer. This thic
        kness can be selected by either selecting a xls-file with the button 'select xls-
        file', or entered manually via the button 'Manual input'. When the button for manual
         input is clicked, a box will appear in which the minimal thickness must be inserted
         and a 'confirm' button to confirm the entry. Note that the units of the input are m
        ilimeters [mm].")
66.         self.Button_step6a = Button(self, text="Select XLS file", width=23, height=2, bo
        rderwidth=4, command=self.SelectXLSfile)
67.         self.Label_message6b = Label(self, height=4, wraplength=350, justify=LEFT, text=
        "OR")
68.         self.Button_step6b = Button(self, text="Manual input", width=23, height=2, borde
        rwidth=4, command=self.manualinput)
69.
70.         self.Label_7 = Label(self, height=2, font=11, text="Step 7. Perform the audit.")

71.
72.         self.Label_message8a = Label(self, height=4, wraplength=350, justify=RIGHT, text
        ="The required thickness of the pavement slab is [mm]: ")
73.         self.Label_message8b = Label(self, height=4, wraplength=350, justify=RIGHT, text
        ="The thickness of the selected pavement slab is [mm]: ")
74.
75.         # Data input via .xls file
76.
77.         self.Label_message7xls = Label(self, height=9, wraplength=350, justify=LEFT, tex
        t="Click on the button 'Perform audit' in order to acquire the results and execute t
```

```
he audit. Note that the results will be presented in the next step, a result output file will not yet be created")
78.         self.Button_7xls = Button(self, text="Perform audit", width=23, height=2, borderwidth=4, command=self.executeauditxls)
79.
80.         self.Label_8xls = Label(self, height=2, font=11, text="Step 8. Generate output for proof of audit.")
81.         self.Label_message8xlsP1 = Label(self, height=4, wraplength=350, fg='darkgreen', justify=LEFT, text="The relevant data complies with the minimal requirements. The data passed the selected test.")
82.         self.Label_message8xlsP2 = Label(self, height=4, wraplength=350, justify=LEFT, text="Click on the button 'Generate output' in order to store the positive results in the working folder in a XLS-file named: 'Audit results.xls'.")
83.         self.Label_message8xlsN1 = Label(self, height=4, wraplength=350, fg='red', justify=LEFT, text="The relevant data does not comply with the minimal requirements. The data did not pass the selected test.")
84.         self.Label_message8xlsN2 = Label(self, height=4, wraplength=350, justify=LEFT, text="Click on the button 'Generate output' in order to store the negative results in the working folder in a XLS-file named: 'Audit results.xls'.")
85.         self.Button_8xls = Button(self, text='Generate output', width=23, height=2, borderwidth=4, command=self.generateoutputxls)
86.         self.Label_messagestored = Label(self, height=6, wraplength=350, justify=LEFT, text="Audit data output stored in working folder. Close the application using the [X].")
87.         self.Button_openfilelocation = Button(self, text="Open folder location", width=23, height=2, borderwidth=4, command=self.openfilelocation)
88.
89.         # Manual data input
90.
91.         self.Label_message6m = Label(self, height=11, wraplength=350, justify=LEFT, text="'Manual input' has been selected as input method. Enter the value manually in the entry box below in [mm] and click on 'confirm entry'")
92.         self.Button_step6c = Button(self, text="Confirm entry", width=23, height=2, borderwidth=4, command=self.getmanualentry)
93.         self.manualentrysv = StringVar()
94.         self.Entry_step6man = Entry(self, textvariable=self.manualentrysv)
95.
96.         self.Label_message7man = Label(self, height=9, wraplength=350, justify=LEFT, text="Click on the button 'Perform audit' in order to acquire the results and execute the audit. Note that the results will be presented in the next step.")
97.         self.Button_7man = Button(self, text="Perform audit", width=23, height=2, borderwidth=4, command=self.executeauditman)
98.
99.         self.Label_8man = Label(self, height=4, wraplength=350, justify=LEFT, text="Step 8. Presenting the results of the analyses")
100.        self.Label_message8manP = Label(self, height=4, wraplength=350, fg='darkgreen', justify=LEFT, text="The manually entered data complies with the minimal requirements. The selected object passed the selected test.")
101.        self.Label_message8manN = Label(self, height=4, wraplength=350, fg='red', justify=LEFT, text="The manually entered data does not comply with the minimal requirements. The data did not pass the selected test.")
102.        self.Label_message8manClose = Label(self, height=9, wraplength=350, text="After interpreting the information you may close the application by using the '[X]' on the top right of this screen.")
103.
104.        # Creating the logo and empty lines used in the GUI for graphical purposes.
105.
106.        self.Label_whiteline1 = Label(self, fg='RED', bg='lightblue', height=3, text="", width=60)
107.        self.Label_whiteline1.grid(row=0, column=0, sticky=N)
108.        self.Label_whiteline2 = Label(self, bg='lightblue', height=1, width=60, text="")

109.        self.Label_whiteline2.grid(row=2, column=0, sticky=N)
110.        self.Label_whiteline3 = Label(self, height=1, width=60, text="")
111.        self.Label_whiteline3.grid(row=4, column=0, sticky=N)
112.        self.Label_whiteline4 = Label(self, height=1, width=60, text="")
```

```
113.        self.Label_whiteline4.grid(row=6, column=0, sticky=N)
114.        self.Label_whiteline5 = Label(self, height=1, width=60, text="")
115.        self.Label_logo = Label(self, fg='red', bg='lightblue', height=3, text="COINS Au
    dit Tool.", font=('Times new roman', 13, "bold italic"))
116.        self.Label_logo.grid(row=30, column=0, sticky=W+E+N+S)
117.        self.Label_logo2 = Label(self, fg='red', bg='lightblue', height=1, text="Designe
    d for Rijkswaterstaat.", font=('Times new roman', 8, "bold"))
118.        self.Label_logo2.grid(row=31, column=0, sticky=W+E+N+S)
119.
120.        # Defining the options of the directory search window (used in step 1).
121.
122.        self.dir_opt = options = {}
123.        options['initialdir'] = 'C:\\'
124.        options['mustexist'] = True
125.        options['parent'] = root
126.        options['title'] = 'Select or create a new folder. The output of the audit and t
    he temporary files will be stored in this folder. Create the new folder first, befor
    e renaming it.'
127.
128.        # Defining the options of the file search window (used in step 2).
129.
130.        self.file_opt = options = {}
131.        options['defaultextension'] = '.ccr'
132.        options['filetypes'] = [('all files', '.*'), ('.ccr files', '.ccr')]
133.        options['initialdir'] = 'C:\\'
134.        options['initialfile'] = 'Title_of_COINS_container.ccr'
135.        options['parent'] = root
136.        options['title'] = 'Select COINS container'
137.
138.    # Defining the commands for step 1. Creating/selecting the COINS container.
139.
140.      def createfolder(self):
141.
142.        self.folder_location = tkFileDialog.askdirectory(**self.dir_opt)
143.
144.        if not os.path.exists(self.folder_location):
145.          os.mkdir(os.path.expanduser(self.folder_location))
146.
147.        self.Label_step1.grid_remove()
148.        self.Button_step1.grid_remove()
149.        self.Label_message1.grid_remove()
150.        self.Label_step2.grid(row=1, column=0, sticky=W)
151.        self.Label_message2.grid(row=3,column=0,sticky=N)
152.        self.Button_step2.grid(row=5, column=0, sticky=N)
153.
154.    # Defining the commands for step 2. Selecting the COINS container.
155.
156.      def selectCOINSc(self):
157.
158.        # Selecting the container via dialog box and extracting the contents into the ma
    in folder
159.
160.        self.selectCOINSc = tkFileDialog.askopenfilename(**self.file_opt)
161.        zip_ref = zipfile.ZipFile(self.selectCOINSc, 'r')
162.
163.        if self.selectCOINSc:
164.
165.          zip_ref.extractall(self.folder_location)
166.
167.        # Defining the folder location path in form of a string.
168.
169.        owlpathtupple = self.folder_location, "/bim/"
170.        owlpath = "".join(owlpathtupple)
171.
172.        for file in os.listdir(owlpath):
173.
```

```
174.          if file.endswith(".owl"):
175.            owlfilename = file
176.            owlfilepathtupple = (owlpath, owlfilename)
177.            self.owlfilepath = "".join(owlfilepathtupple)
178.
179.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
180.
181.        self.Label_step2.grid_remove()
182.        self.Label_message2.grid_remove()
183.        self.Button_step2.grid_remove()
184.        self.Label_step3.grid(row=1, column=0, sticky=W)
185.        self.Label_message3.grid(row=3, column=0, sticky=N)
186.        self.List_1.grid(row=5, column=0, sticky=N)
187.        self.Button_step3.grid(row=8, column=0, sticky=N)
188.        self.Label_whiteline5.grid(row=9, column=0, sticky=N)
189.
190.    # Defining the commands for step 3. Selecting the type of audit.
191.
192.      def AuditSelection(self):
193.
194.        # Retrieving and defining of the audit selection from the list box of step 1.
195.
196.        List_1_Selection = self.List_1.get('active')
197.
198.        if List_1_Selection == 'Thickness of pavement slab':
199.          self.selectedaudit = 'Thickness_of_pavement_slab'
200.
201.        if List_1_Selection == '[for example] Average size of grain in asphalt mixture':

202.          self.selectedaudit = 'Average size of grain in asphalt mixture'
203.
204.        if List_1_Selection == 'Audit #3':
205.          self.selectedaudit = 'Audit #2'
206.
207.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
208.
209.        self.Label_step3.grid_remove()
210.        self.Label_message3.grid_remove()
211.        self.List_1.grid_remove()
212.        self.Button_step3.grid_remove()
213.
214.        if self.selectedaudit == 'Thickness_of_pavement_slab':
215.
216.            self.Label_step4.grid(row=1, column=0, sticky=W)
217.            self.Label_message4.grid(row=3, column=0, sticky=N)
218.            self.List_2.grid(row=5, column=0, sticky=N)
219.            self.Button_step4.grid(row=8, column=0, sticky=N)
220.
221.        # Displaying warning if an audit is selected that has not been included in the p
    rototype audit tool.
222.
223.        if self.selectedaudit != 'Thickness_of_pavement_slab':
224.
225.            self.Labelnotincluded = Label(self, height=4, wraplength=360, justify=LEFT,
    fg='RED', text="This audit has not (yet) been included in the code for this audit to
    ol. Further development of the tool is needed to include more types of audit.")
226.            self.Labelnotincluded.grid(row=9, column=0, sticky=N)
227.
228.    # Defining the commands for step 4. Selecting the required layer of pavement.
229.
230.      def SelectLayer(self):
231.
232.        # Retrieving the list box selection of step 2 and defining the selected layer in
     a string encoded in unicode.
```

```
233.
234.        self.List_2_Selection = self.List_2.get('active')
235.
236.        if self.List_2_Selection == 'Top layer':
237.          self.selectedlayer = u'Asfaltplak_toplaag'
238.
239.        if self.List_2_Selection == 'Middle layer':
240.          self.selectedlayer = u'Asfaltplak_tussenlaag'
241.
242.        if self.List_2_Selection == 'Bottom layer':
243.          self.selectedlayer = u'Asfaltplak_onderlaag'
244.
245.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
246.
247.        self.Label_step4.grid_remove()
248.        self.Label_message4.grid_remove()
249.        self.List_2.grid_remove()
250.        self.Button_step4.grid_remove()
251.        self.Label_step5.grid(row=1, column=0, sticky=W)
252.        self.Label_message5.grid(row=3, column=0, sticky=N)
253.        self.Label_whiteline4.grid(row=6, column=0, sticky=W)
254.        self.List_3.grid(row=7, column=0, sticky=N)
255.        self.Button_step5a.grid(row=5, column=0, sticky=N)
256.
257.    # Defining the commands for step 5. Selecting the required object.
258.
259.      def populateListbox(self):
260.
261.        g=rdflib.Graph()
262.        g.load(self.owlfilepath, format='xml')
263.
264.        self.qres = g.query(
265.
266.            """SELECT DISTINCT ?value ?name ?file ?frag
267.               WHERE {
268.                  ?pv a cbim:PropertyValue.
269.                  ?pv cbim:propertyType <http://bim.rws.nl/OTL/COINS/otl-
    otl.11.owl#OB02859-PR00501.0> .
270.                  ?pv cbim:value ?value.
271.                  ?asfaltplakCP cbim:propertyValue ?pv.
272.                  ?asfaltplakCP cbim:name ?name.
273.                  ?asfaltplak cbim:contains[cbim:cataloguePart ?asfaltplakCP].
274.                  ?asfaltplak cbim:shape ?rep.
275.                  ?rep cbim:documentAliasFilePath ?file.
276.                        }""",
277.
278.             initNs=dict(
279.                cbim=Namespace("http://www.coinsweb.nl/cbim-1.1.owl#")))
280.
281.        for row in self.qres:
282.            if rdflib.term.Literal(self.selectedlayer, datatype=rdflib.term.URIRef(u'htt
    p://www.w3.org/2001/XMLSchema#string')) in row:
283.                output = row['file']
284.                self.lst = output.split("\n")
285.                self.List_3.insert("end", *self.lst)
286.
287.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
288.
289.            self.Button_step5a.grid_remove()
290.            self.Label_whiteline4.grid(row=20, column=0, sticky=N)
291.            self.Button_step5b.grid(row=21, column=0, sticky=N)
292.            self.Label_whiteline5.grid(row=22, column=0, sticky=N)
293.
294.      def selectobject(self):
```

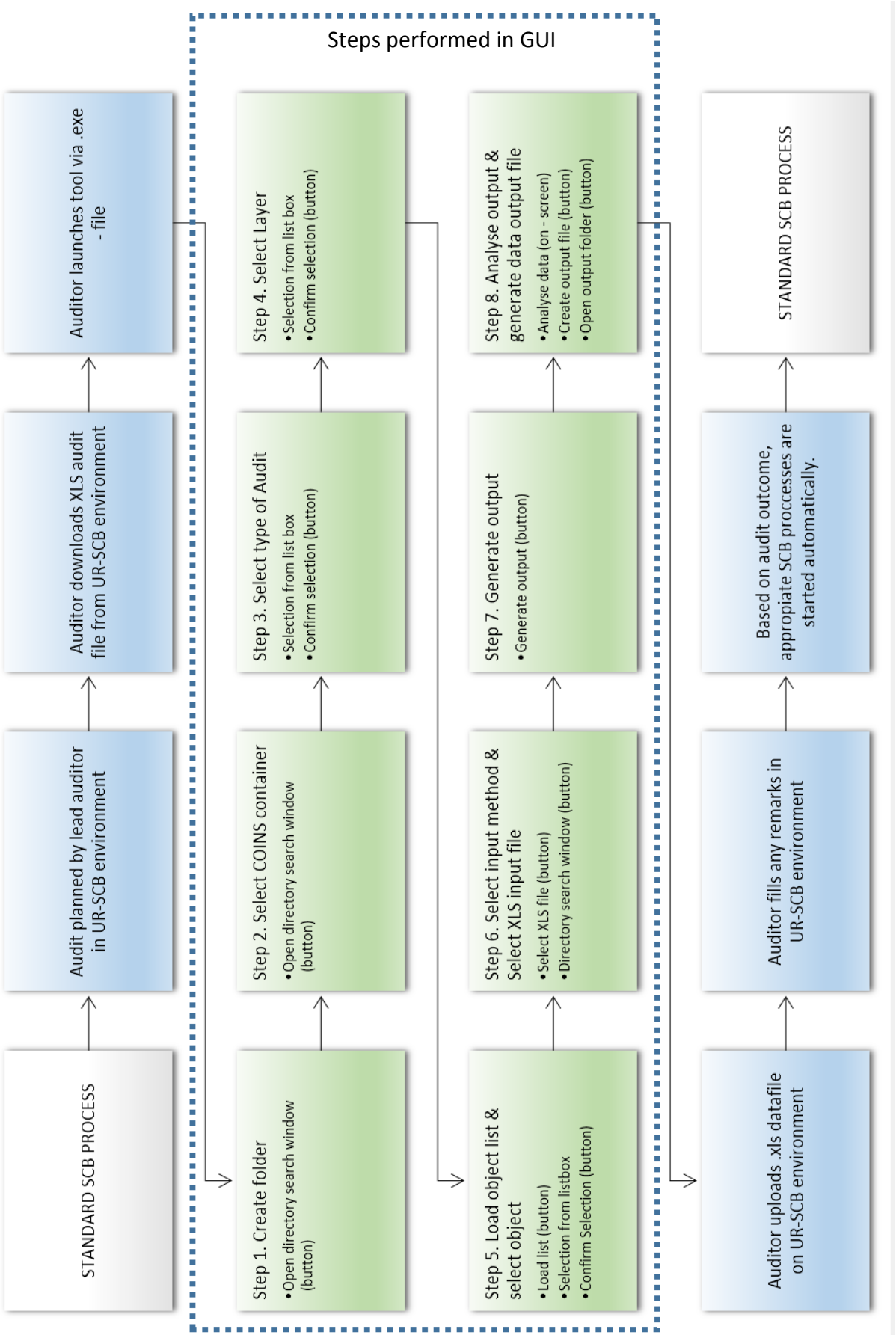
```
295.
296.        self.selectedobject = self.List_3.get('active')
297.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
298.
299.        self.Label_step5.grid_remove()
300.        self.Label_message5.grid_remove()
301.        self.List_3.grid_remove()
302.        self.Button_step5b.grid_remove()
303.        self.Label_step6.grid(row=1, column=0, sticky=W)
304.        self.Label_message6.grid(row=3, column=0, sticky=N)
305.        self.Button_step6a.grid(row=5, column=0, sticky=N)
306.        self.Label_message6b.grid(row=6, column=0, sticky=N)
307.        self.Button_step6b.grid(row=7, column=0, sticky=N)
308.
309.    # Defining the commands for step 6 (automated input via spreadsheet file). Selecting
     the required xls file
310.
311.      def SelectXLSfile(self):
312.
313.        self.XLSfilelocation = tkFileDialog.askopenfilename(**self.file_opt)
314.
315.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
316.
317.        self.Label_step6.grid_remove()
318.        self.Label_message6.grid_remove()
319.        self.Button_step6a.grid_remove()
320.        self.Label_message6b.grid_remove()
321.        self.Button_step6b.grid_remove()
322.        self.Label_7.grid(row=1, column=0, sticky=W)
323.        self.Label_message7xls.grid(row=3, column=0, sticky=N)
324.        self.Button_7xls.grid(row=5, column=0, sticky=N)
325.
326.    # Defining the commands for step 7 (automated input via spreadsheet file). Performin
    g the audit calculations.
327.
328.      def executeauditxls(self):
329.
330.        for row in self.qres:
331.
332.            if (rdflib.term.Literal(self.selectedlayer, datatype=rdflib.term.URIRef(u'ht
    tp://www.w3.org/2001/XMLSchema#string')) in row) and (rdflib.term.Literal(self.selec
    tedobject, datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#string'))
    in row):
333.
334.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
335.
336.              self.obtained_value = row['value']
337.              self.Label_7.grid_remove()
338.              self.Label_message7xls.grid_remove()
339.              self.Button_7xls.grid_remove()
340.              self.Label_whiteline3.grid_remove()
341.              self.Label_whiteline4.grid_remove()
342.
343.              self.workbook = xlrd.open_workbook(self.XLSfilelocation)
344.              self.sheet = self.workbook.sheet_by_index(0)
345.              self.required_value = self.sheet.cell_value(1, 12)
346.              self.required_value_output = StringVar()
347.              self.required_value_output.set(self.required_value)
348.              self.obtained_value_output = StringVar()
349.              self.obtained_value_output.set(self.obtained_value)
350.
351.        # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
```

```
352.
353.             self.Label_8xls.grid(row=1, column=0, sticky=W)
354.             self.Label_message8a.grid(row=4, column=0, sticky=E)
355.             self.Label_message8xlsRV = Label(self, height=2, wraplength=350, justify=L
    EFT, textvariable=self.required_value_output)
356.             self.Label_message8xlsRV.grid(row=4, column=1, sticky=W)
357.             self.Label_message8b.grid(row=5, column=0, sticky=E)
358.             self.Label_message8xlsOV = Label(self, height=2, wraplength=350, justify=L
    EFT, textvariable=self.obtained_value_output)
359.             self.Label_message8xlsOV.grid(row=5, column=1, sticky=W)
360.             self.Label_whiteline1.grid(row=0, column=0, columnspan=2,sticky=W+E)
361.             self.Label_whiteline2.grid(row=3, column=0, columnspan=2, sticky=W+E)
362.             self.Label_whiteline3.grid(row=8, column=0, columnspan=2, sticky=W+E)
363.             self.Label_whiteline4.grid(row=10, column=0,  columnspan=2, sticky=W+E)
364.             self.Label_whiteline5.grid_remove()
365.             self.Label_logo.grid(row=20, column=0, columnspan=2, sticky=W+E+N+S)
366.             self.Label_logo2.grid(row=21, column=0, columnspan=2, sticky=W+E+N+S)
367.             self.Button_8xls.grid(row=9, column=0, sticky=N)
368.
369.             if self.required_value > self.obtained_value:
370.
371.                 self.Label_message8xlsN1.grid(row=6, column=0, sticky=W)
372.                 self.Label_message8xlsN2.grid(row=7, column=0, sticky=W)
373.
374.             if self.required_value <= self.obtained_value:
375.
376.                 self.Label_message8xlsP1.grid(row=6, column=0, sticky=W)
377.                 self.Label_message8xlsP2.grid(row=7, column=0, sticky=W)
378.
379.     def generateoutputxls(self):
380.
381.         tuple_xls_savename = self.folder_location, "/Audit results.xls"
382.         xls_savename = "".join(tuple_xls_savename)
383.         df = self.XLSfilelocation
384.         rb = open_workbook(df)
385.         wb = copy(rb)
386.         s = wb.get_sheet(0)
387.         s.write(0,14, "Are requirements met?")
388.         s.write(0,15, "Selected layer")
389.         s.write(0,16, "Selected object")
390.         s.write(1,15, self.List_2_Selection)
391.         s.write(1,16, self.selectedobject)
392.         s.write(1,13, self.obtained_value)
393.
394.         if self.required_value <= self.obtained_value:
395.
396.                 s.write(1,14, 'Meets the requirements, positive audit outcome.')
397.                 wb.save(xls_savename)
398.                 self.Label_message8xlsP2.grid_remove()
399.
400.         if self.required_value > self.obtained_value:
401.
402.                 s.write(1,14, 'Does not meet requirements, negative audit outcome.')
403.                 wb.save(xls_savename)
404.                 self.Label_message8xlsN2.grid_remove()
405.
406.         # Removing used labels, list boxes, entry boxes, and buttons from grid and place
    ment of new components on grid.
407.
408.         self.Label_whiteline5.grid_remove()
409.         self.Button_8xls.grid_remove()
410.         self.Label_messagestored.grid(row=11, column=0, sticky=N)
411.         self.Label_whiteline4.grid(row=12, column=0, sticky=N)
412.         self.Button_openfilelocation.grid(row=13, column=0, sticky=N)
413.         self.Label_whiteline5.grid(row=14, column=0, sticky=N)
```

```
414.
415.    # Defining the commands for step 7 (manual input via entry box). Performing and pres
       enting the audit calculations.
416.
417.      def openfilelocation(self):
418.
419.        webbrowser.open(self.folder_location)
420.
421.    ##########################################################################################
       ##############################
422.
423.      def manualinput(self):
424.
425.        self.Label_message6.grid_remove()
426.        self.Button_step6a.grid_remove()
427.        self.Label_message6b.grid_remove()
428.        self.Button_step6b.grid_remove()
429.        self.Label_message6m.grid(row=3, column=0, sticky=N)
430.        self.Entry_step6man.grid(row=5, column=0, sticky=N)
431.        self.Button_step6c.grid(row=6, column=0, sticky=N)
432.
433.      def getmanualentry(self):
434.        for row in self.qres:
435.
436.            if (rdflib.term.Literal(self.selectedlayer, datatype=rdflib.term.URIRef(u'ht
       tp://www.w3.org/2001/XMLSchema#string')) in row) and (rdflib.term.Literal(self.selec
       tedobject, datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#string'))
       in row):
437.
438.              self.obtainedvalue = row['value']
439.
440.              self.manualentry = self.manualentrysv.get()
441.              self.Label_step6.grid_remove()
442.              self.Label_message6m.grid_remove()
443.              self.Entry_step6man.grid_remove()
444.              self.Button_step6c.grid_remove()
445.              self.Label_7.grid(row=1, column=0, sticky=W)
446.              self.Label_message7man.grid(row=3, column=0, sticky=W)
447.              self.Button_7man.grid(row=5, column=0, sticky=N)
448.
449.      def executeauditman(self):
450.
451.              self.Label_7.grid_remove()
452.              self.Label_message7man.grid_remove()
453.              self.Button_7man.grid_remove()
454.
455.              self.Label_whiteline1.grid(row=0, column=0, columnspan=2,sticky=W+E)
456.              self.Label_whiteline2.grid(row=3, column=0, columnspan=2, sticky=W+E)
457.              self.Label_whiteline3.grid(row=8, column=0, columnspan=2, sticky=W+E)
458.              self.Label_whiteline4.grid_remove()
459.              self.Label_whiteline5.grid_remove()
460.
461.              self.ObtainedValueManOutput = StringVar()
462.              self.ObtainedValueManOutput.set(self.obtainedvalue)
463.              self.ManuallyEntryOutput = StringVar()
464.              self.ManuallyEntryOutput.set(self.manualentry)
465.
466.              self.Label_8man.grid(row=1, column=0, sticky=W)
467.              self.Label_message8a.grid(row=4, column=0, sticky=E)
468.              self.Label_message8manRV = Label(self, height=2, wraplength=350, justify=L
       EFT, textvariable=self.ManuallyEntryOutput)
469.              self.Label_message8manRV.grid(row=4, column=1, sticky=W)
470.              self.Label_message8b.grid(row=5, column=0, sticky=E)
471.              self.Label_message8manOV = Label(self, height=2, wraplength=350, justify=L
       EFT, textvariable=self.ObtainedValueManOutput)
472.              self.Label_message8manOV.grid(row=5, column=1, sticky=W)
```

```
            self.Label_message8manClose.grid(row=9, column =0, sticky=W)
            self.Label_logo.grid(row=20, column=0, columnspan=2, sticky=W+E+N+S)
            self.Label_logo2.grid(row=21, column=0, columnspan=2, sticky=W+E+N+S)

            self.obtainedvalueman = float(self.obtainedvalue)
            self.manualentryvalue = float(self.manualentry)

            if self.manualentryvalue <= self.obtainedvalueman:

                self.Label_message8manP.grid(row=6, column=0, sticky=W)

            if self.manualentryvalue > self.obtainedvalueman:

                self.Label_message8manN.grid(row=6, column=0, sticky=W)

if __name__=='__main__':

  root = Tkinter.Tk()
  TkFileDialogExample(root).pack()
  gc.collect()
  del gc.garbage[:]
  root.mainloop()
  gc.collect()
  del gc.garbage[:]
```

Appendix 4: The process for an auditor



Appendix 5: The process for an asset manager