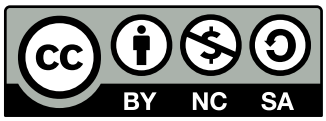
Aalto University
School of Engineering

Master's programme in Master's Programme in Mechanical Engineering

Automated Data Quality Check in Plant Digital Twins

Viktor Ketola

Master's Thesis
2023



Aalto University
School of Engineering

Author Viktor Ketola

Title Automated Data Quality Check in Plant Digital Twins

Degree programme Master's Programme in Mechanical Engineering

Major Mechanical Engineering

Supervisor Prof. Petri Kuosmanen

Advisor Dr-Ing. Christian Binder, M.Sc (Tech.) Kaur Jaakma

Collaborative partner Metso Oyj

Date 30 September 2023 **Number of pages** 61 **Language** English

Abstract

Advances in the fields of Industrial Internet, computational power and artificial intelligence currently happen at a rapid pace. Following this development, the prospect of implementing Digital Twins (DT) in order to leverage these technologies for tangible benefits is more appealing than ever. Building accurate DTs requires substantial amounts of data and documentation, which naturally also need to be correct in order to build a DT for industrial purposes.

This study had the goals of evaluating the current state of the art in the field, as well as investigating how the documentation and metadata of a minerals processing plant DT at Metso Oyj can be checked. A literature review was carried out in order to provide the reader with clear definition of the concepts, and with an overview of the latest developments. For checking purposes, it was decided that the focus of the study should be on checking that the source data created in AVEVA's Engineering and Diagrams packages is correct. Two tools were designed and implemented in the Python programming language, one implemented as web application and the other as an executable file accessible directly in AVEVA Engineering. The tools save approximately 4-8 hours of working time per project.

It was also investigated if equipment metadata in the form of asset tags could be scanned for inaccuracies with fuzzy string matching algorithms. Promising, but not conclusive results were found in this study.

In conclusion, the results of the study indicate that the implemented tools can significantly help with improving the data quality, and thus facilitating better DTs.

Keywords digital twin, data quality, AVEVA, string matching

Aalto-universitetet
Högskolan för
ingenjörsvetenskaper

Författare Viktor Ketola

Titel Automatisk kvalitetsgranskning av data i en fabriksanläggnings digitala tvilling

Utbildningsprogram Master's Programme in Mechanical Engineering

Huvudämne Maskinteknik

Övervakare Prof. Petri Kuosmanen

Handledare TkD Christian Binder, DI Kaur Jaakma

Samarbetspartner Metso Oyj

Datum 30 September 2023 **Sidantal** 61 **Språk** engelska

Sammandrag

I dagsläget sker framsteg i rasande fart inom områdena industriellt internet, beräkningskraft och artificiell intelligens. I kölvattnet av denna utveckling ter sig möjligheterna att implementera digitala tvillingar (DT) för att uppnå påtagliga fördelar mer lovande än någonsin. För att skapa en verklighetstrogen DT behövs substantiella mängder data och dokumentation, vilka naturligtvis också bör vara korrekta för att förverkliga en DT för industriellt bruk. Denna studie hade målsättningarna att utvärdera de senaste teknologiska framstegen inom området, samt att undersöka hur dokumentation och metadata tillhörande en anrikningsanläggnings DT på bolaget Metso Oyj kan kvalitetssäkras. En litteraturöversikt utfördes för att bistå läsaren med klara definitioner av begreppen inom området, samt en översikt över de senaste framstegen. Gällande kvalitetssäkring bestämdes det att studien fokuserar på att granska att källdata som skapas i mjukvaran AVEVAs Engineering- samt Diagrams-moduler är korrekta. Två verktyg utvecklades och implementerades med programmeringsspråket Python, varav ett implementerades som en webbapplikation och det andra som en körbar fil integrerat i AVEVA Engineering. Verktygen sparar ungefär 4-8 timmar arbetstid per projekt.

Det undersöktes också huruvida fabriksutrustningens metadata kan skannas för att upptäcka felaktigheter med algoritmer för oskarp strängsökning. Resultaten var lovande, men inga definitiva resultat upptäcktes i denna studie.

Nyckelord digital tvilling, datakvalitet, AVEVA, strängsökning

Preface

I want to thank my advisor Dr-ing. Christian Binder for the opportunity to write this thesis as a project for Metso in his team with nice and helpful colleagues, and also for his ideas and comments, Professor Petri Kuosmanen and M.Sc (Tech.) Kaur Jaakma for their academic guidance, and B.Tech Shakti Dwivedi for his help with all matters related to the AVEVA software package.

I also want to thank all the friends I have made during my studies at Teknologföreningen and Vasa Nation for all the time spent on course assignments, lunch breaks, extracurricular activities, parties and other good times. Lastly, a word of mention for my family and their support during this endeavour.

Otaniemi, 30 September 2023

Viktor E. K. Ketola

Contents

1 Introduction

1.1 Background

In the highly competitive industry of today, it is important for companies to minimize their time to market and to maximize the performance of development projects. With intense schedules, the probability of errors happening increases which in turn could prove to be a major issue in any project, causing delays and even dangerous products.

Industry 4.0 is accelerating innovation in digitalization and data utilization to address such issues. One such innovation is the concept of the Digital Twin (DT), or a virtual representation of a physical instance with a real-time connection between the instances. A DT can be used for many purposes with the most prevalent ones being prototyping, visualization, simulation, optimization, training and maintenance. To this date, they have been applied in several different industries such as manufacturing, process industries, maritime industry, healthcare, and smart cities. DT has seen a large increase in academic and industrial interest in the past decade and has been regarded as one of the most promising technology concepts being developed.

Leveraging DT in the design stage of a product or a plant comes with several advantages. For example, designers can simulate and conduct what-if scenarios in order to optimize the design and also see how the asset being currently developed would fit in with existing assets. In the design phase, it is highly important that drawings, images, and other types of design data and asset information are verified and validated in order to provide the correct information for the next intended use of the DT.

Metso provides DT solutions with both metallurgical process simulation in the form of the Geminex product, and plant monitoring with a visual interface linked to plant documentation and sensors. The latter is currently implemented utilizing AVEVA and Autodesk solutions. However, currently there are issues with data going lost due to both technical and human errors. The problems are mainly that data tags are named incorrectly, and that documentation is missing or not properly linked. Tags being incorrectly named also break the automated data linkage in the DT solution. This can lead to inferior models and incorrect information being delivered to other teams within the company, or customers. It is thus of importance to ensure that the correct data and documents along with their links are delivered.

This study provided an overview of recent developments in DT and moves on to the development of a tool with the purpose of quality controlling a DT's linked data and documents. The tool is capable of automated quality check and is able to provide the necessary feedback to make corrections or improvement before product delivery.

In Chapter 2, the concept of Digital Twin is expanded and explained. A brief presentation of literature with the current state of the art in the field of plant DT is presented. It is also further clarified what role the terms Computer Aided Design (CAD) and Building Information Modelling (BIM) play in this context. Chapter 3 introduces the reader to the AVEVA software package for process plant design and discusses the development process of a tool for performing the data quality check and generating reports. Requirements are formalized and explained. Furthermore, the

chapter delves into how the data that should be checked is stored, along with different paths that could be taken in order to check the data. The results are presented in Chapter 4, and are discussed in Chapter 5.

1.2 Aim of the study

Quality control is important to safeguard the company's reputation, prevent products from being unreliable, and increase trust on the side of consumers. It ensures that the company looks at evidence-based data and research rather than anecdotal observations to ensure that the services/products live up to the standards.

With current practice, data goes lost during handover between design phases. A common problem is that the tagging has not been done according to the agreed-upon naming convention. This is because currently, the designers tag the objects manually according to Piping and Instrumentation Diagrams (P&ID) which opens for the possibility of human error. Issues with the software itself or connection errors also cause such problems.

Another issue is that the documents themselves have been named improperly or simply do not exist due to human mistakes, which causes problems when they are to be uploaded to the project file vault. The idea is that the tool should be able to detect errors and generate a report after the check. Metso wants to develop some automated way to do the quality check and provide the necessary feedback for the improvement or corrections before the project or data is delivered to the customer, or the next stage of the project. The aim of this study was to answer the following questions:

1. What is the current state of the art in plant Digital Twins?
2. How can the project documentation metadata be checked?
3. What would be the most suitable platform for building a tool?
4. Can this significantly relieve the workload of engineers? How much time and effort can be saved?

1.3 Constraints

Plant design software and its relevant documentation was the point of interest while conducting this study. The scope of this study was to investigate what would be the best way to check the data quality of AVEVA file documentation, decide on how to implement checks and validate by examples. The study explained DT and BIM as concepts, presented the state of the art in the field of DTs, gave insight into the AVEVA software, and presented the development process of a checker tool and the corresponding results.

The study was limited to data coming from AVEVA software and thus does not necessarily encompass a ready-to-use solution for similar plant design software. Since the focus of this study was on the data tags and linked documents, it means that Model Quality Testing (MQT) was out of the scope. MQT refers to checking Computer

Aided Design (CAD) models for morphological errors, which relate to the topological and geometrical correctness of a CAD model. Gonzáles-Lluch et. al [1] defined topological errors as either global errors, which refer to inconsistent orientation and topological noise, or local errors which can be unrealistic shapes, under-connection of an element or over-connection of an element. The same study also defined geometric errors as local or global, with local errors being categorized into punctures (open shells) and overlaps, and global errors which are commonly noise resulting from data corruption which ruins digitized meshes. This type of testing was deemed to be out of the scope, since this study handles plant models, where the point of the visual model is to give the user an interface and not to produce drawings for manufacturing. Such model accuracy is in this case not necessary even if desirable, limiting the need for testing for morphological errors.

1.4 Methodology

A literature review on the current state of DT in the context of process plants has been carried out. In the literature review, an introduction to the historical background and terminology are presented. Challenges in the field have also been addressed. Furthermore, the components and standards that are needed to satisfy the requirements of a DT are explained further. This was to provide an overview of the technology solutions in use.

A deeper dive into the technicalities of the software package at hand, AVEVA, will be presented, along with a presentation about the document management system by studying manuals provided by the developers. Requirements for the checks to be carried out have been formalized and discussed, and the data storage architecture of the software package has been analysed.

The experimental part was based on findings in how the data is stored, and on problem statements from interviews with engineers currently encountering the issues to be solved. Potential software solutions to the issues were investigated, and after analysing the possibilities, development of a tool commenced. The tool was tested on both a test sample and a real-life project, with successful results.

2 Plant Digital Twin Models

Building a Digital Twin means that a part or a process is replicated virtually for the purpose of simulating and observing a physical instance of the same product. The origin of utilizing a "twin" of another asset for practical purposes is usually credited to the NASA rescue mission of Apollo 13 in April 1970. What occurred was that shortly after the launch of Apollo 13, the spacecraft was damaged by an explosion in the oxygen tanks, and a rescue mission commenced. The rescue mission was successful, and this was largely due to the fact that NASA duplicated the spacecraft's physical systems in the form of simulators on earth so that mission control could research and decide on the strategy to bring the astronauts home [2].

The concept of linking real space and virtual space was introduced in 2002 by Dr. Michael Grieves at the University of Michigan in a lecture of his on the topic Product Lifecycle Management (PLM). At this time, a digital representation of a physical system or product was regarded as a new and immature technology. The information being collected was still mostly manually collected and paper-based and not automated and digitized. In the past decades since, there has been an enormous increase in the amount of information that can be collected from both the physical and virtual side, facilitating the development of the concept to what it is today [3]. The term "Digital Twin" was eventually used by NASA in 2010 in their Technology Area 12 report [4]. However, DARPA (Defense Advanced Research Projects Agency) was later credited by NASA for coining the phrase. The use of DT started and evolved in the aerospace industry and was eventually translated into other areas such as automotive engineering, plant engineering and smart cities.

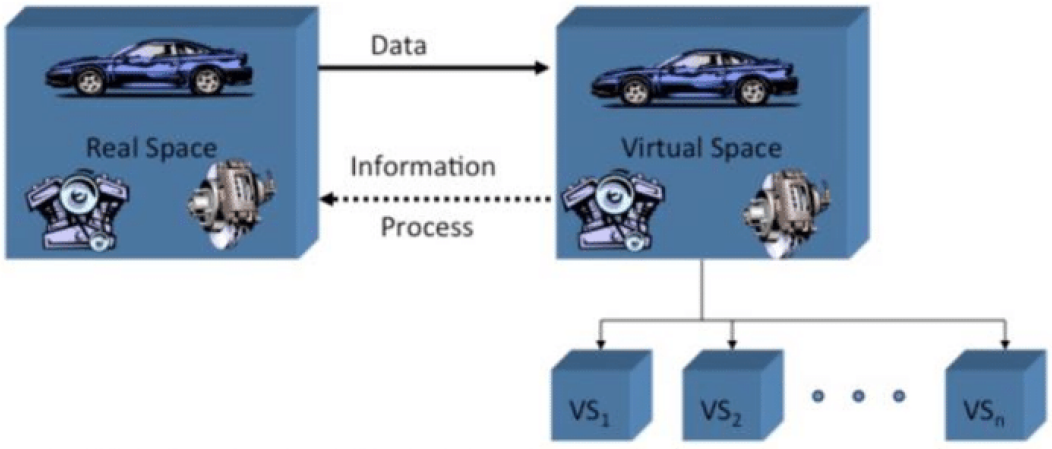


Figure 1: Dr Grieves' original concept for a DT with the slide title "Conceptual ideal for PLM". [5]

The interest in DT has sky-rocketed in the past decade as seen in Figure 2 and Figure 3. However, there is no real consensus at the moment regarding the exact definition of DT. In this chapter, some of the most commonly used definitions will be explained and distinguished.

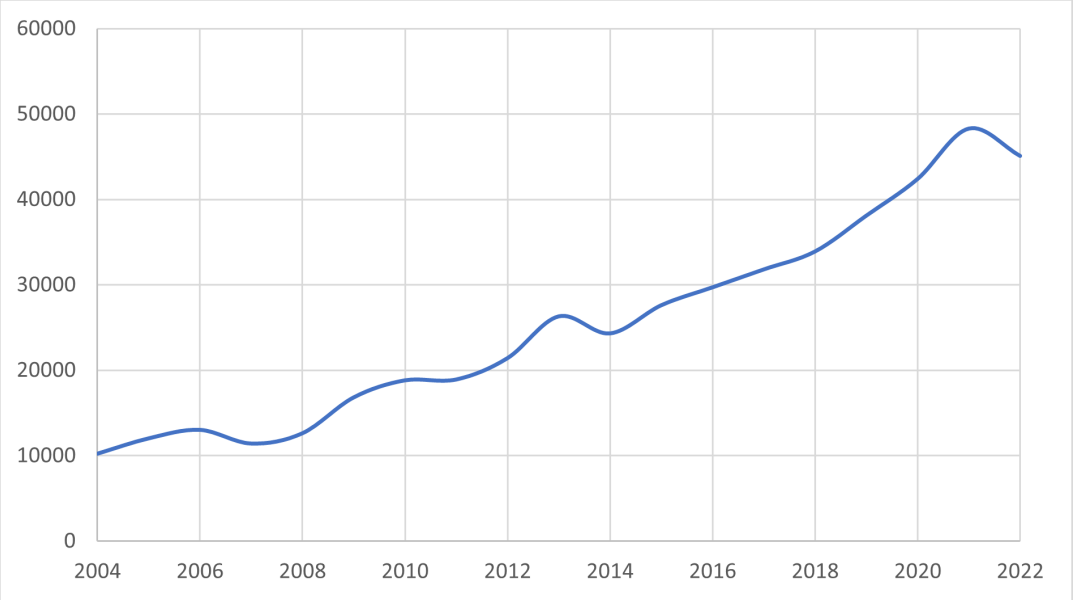


Figure 2: The number of publications yearly where the term 'Digital Twin' occurs. Only articles indexed by Google Scholar are included, meaning that patents and citations are excluded. The data was scraped with V. Strobel's "Academic keyword occurrence" script [6].

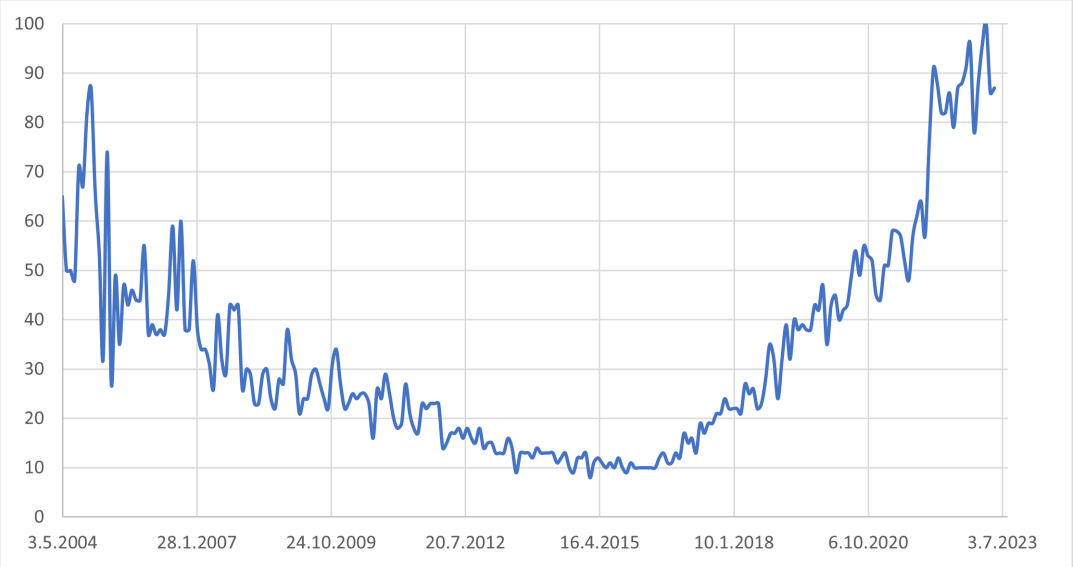


Figure 3: The relative interest for the search term "Digital Twin" in Google Search since 2004. Note that the data has been scaled to an interval of 0-100. [7]

2.1 Definitions of Digital Twin

Present literature

Even though the original concept presented by Dr. Grieves was from a lecture back in 2002, the idea which presents the basic characteristics of DT is still the fundamental concept today - linking "real space" and "virtual space" via information exchange. Grieves' concept with three components can be expanded for more specific definitions depending on the use case. Today, there is no real consensus on the exact definition of a DT, similarly to how the concept of "Mechatronics" can mean slightly different things depending on the opinion of an author or institution.

Grieves' and Vickers' general definition of DT is *"the Digital Twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physical manufactured product can be obtained from its Digital Twin."* [8] The latter can be seen as the final, in practice unreachable goal of the DT concept.

Another widely used definition of DT is NASA's definition from 2010 formulated as: *"A digital twin is an integrated multi-physics, multi-scale, probabilistic simulation of a vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin."* [9]

Stark & Damerau have formulated a definition from the standpoint of manufacturing and production engineering as: *"A digital twin is a digital representation of an active unique product (real device, object, machine, service, or intangible asset) or unique product-service system (a system consisting of a product and a related service) that comprises its selected characteristics, properties, conditions, and behaviors by means of models, information, and data within a single or even across multiple life cycle phases."* [10]

From the definitions found in literature, it is clear that one of the key aspects of a DT is that it must be associated with a physical object or process that exists. It could be compared to a library that contains all the available information about a product or a system. Respectively, there are some requirements on the virtual part of the DT pairing. From the listed definitions, it can be seen that the most important aspects of the virtual part is:

- It needs to contain models that accurately reflect the physical system, this can be both simulations and 3D models where applicable.
- It must be paired with a constantly updating data set related to the physical system coming from e.g. sensors in the physical system.
- The model must be adjustable according to the information exchange in order to more accurately reflect the physical world.
- The information data sets need to contain accurate data, with metadata formulated in a sensible way so that information of interest can be easily accessed.

Digital Twin compared to a Model

With the concepts being closely related to each other, it is necessary to clarify what differs a DT from a simple model. Wright and Davidson [11] state that *"a digital twin without a physical twin is a model"*. To illustrate what this means, functionalities that belong to a DT but not a model are listed in this section. According to Madni, Madni and Lucero , the most important differences between a Computer Aided Design (CAD) or Computer Aided Engineering (CAE) model and a DT are [12]:

(a) DT is a specific instance that reflects the structure, performance, health status, and mission-specific characteristics such as miles flown, malfunctions experienced, and maintenance and repair history of the physical twin.

(b) DT helps determine when to schedule preventive maintenance based on knowledge of the system's maintenance history and observed system behaviour.

(c) DT helps in understanding how the physical twin is performing in the real world, and how it can be expected to perform with timely maintenance in the future.

(d) DT allows developers to observe system performance to understand, for example, how modifications are performing, and to get a better understanding of the operational environment.

(e) DT promotes traceability between life cycle phases through connectivity provided by the digital thread.

(f) DT facilitates refinement of assumptions with predictive analytics-data collected from the physical system and incorporated in the digital twin can be analysed along with other information sources to make predictions about future system performance.

(g) DT enables maintainers to troubleshoot malfunctioning remote equipment and perform remote maintenance.

(h) DT combines data from the IoT with data from the physical system to, for example, optimize service and manufacturing processes and identify needed design improvements (e.g., improved logistics support, improved mission performance).

(i) DT reflects the age of the physical system by incorporating operational and maintenance data from the physical system into its models and simulations.

The key in this whole reasoning is that contrary to a model, a DT utilizes data from the physical part to update the digital version to in turn use it to predict future states of the system.

Terminology at Metso

In the context of Metso's internal terminology, there are several definitions. As per internal documents, they are seen in Table 1.

Table 1: Definitions used internally at the company.

Engineering Digital Twin	Digital version of the physical plant (e.g., BIM), meaning a 3D model with structures etc. - the purpose of this is to give a visualization of locations and setups.
Metallurgical Digital Twin / Process Digital Twin	Digital simulation model of the process, e.g. a flow sheet simulation in HSC Chemistry, Modelica, Apros - benefits in the form of process optimization, increased profitability and higher recoveries.
Asset / Machine / Equipment Digital Twin	DT of a plant component based on engineering data.
Data-based Digital Twin	Based on historical process data only, meaning that no simulations are used in this case.
Training Digital Twin	DT used for training personnel in e.g., operating a process - Purpose is to give higher plant availability and increased safety.

2.2 Digital Twin Types

Furthermore, it is helpful to expand the concept of a DT into parts to more properly describe which stage of development the DT is currently in. The purpose of this is to provide definitions to rely on when discussing a DT development project and thereby provide clarity on exactly what is being discussed. Grieves and Vickers have proposed a distinction to be made in the following way [8]:

- Digital Twin Prototype
- Digital Twin Instance

Digital Twin Prototype according to Grieves, is a description of the prototype of a physical instance. A DTP contains all the information necessary to describe and produce a physical version of the virtual version. In accordance with Madni, Madni and Lucero [12], a DTP is thus just a model and not yet a full-fledged DT. Such information may be, but is not limited to:

- Fully annotated 3D model with geometric dimensioning and tolerancing.

- Requirements
- Bill of Materials with information about all current and past components.
- Bill of Processes that describes all operations performed in order to create the physical product.
- Bill of Services
- Bill of Disposal

Digital Twin Instance is an instance that mirrors a physical product and is meant to remain linked to it for the whole life cycle of the product [8]. A DTI may contain, but is not limited to:

- Fully annotated 3D model with geometric dimensioning and tolerancing.
- Bill of Materials
- Bill of Processes
- Service Record that lists all components replaced and past service and maintenance done to the product.
- Operational States which show actual current, past, and future predicted sensor data.

The main difference here is that a DTI contains measured data about the components of the product that has been obtained with sensors, and can be interrogated for its system state, meaning that all instrumented characteristics can be accessed remotely. A DTP can only predict behavior by using the characteristics stated about the components in data sheets.

2.3 Enabling technologies

Technological advances in the past decades have made DT solutions more realistic and attainable than the situation when Grieves presented his original concept. The field of DT and its related technologies have seen a lot of interest from both industry and academia as previously stated, which has led to considerable amounts of time and money being invested in developing these. The following technologies are important for DT development and have seen significant advances in recent times [13] [14]:

1. **Internet of Things (IoT) & Sensor technology**

   IoT refers to giving "things" a sense of intelligence by fitting them with sensors in order to collect information about their environments and own condition which is then transmitted over a network. IoT is to a significant extent one of the most important building blocks for implementing DT in industry since it enables the information link between the virtual and physical instance. It has been estimated that Industrial IoT could add $14.5 trillion to the global economy by the year 2030 [14].

2. **Artificial Intelligence (AI)**

   AI along with machine and deep learning algorithms is today a leading facilitator in enabling DT systems and is set to become among the most important components in them. Advancements in this field currently happen at a rapid pace and is also an object of study for research organizations worldwide. In the context of DT, AI is useful for analysing data from IoT sensors to provide feedback from the physical instance, and is also particularly useful in predictive maintenance and estimating lifetime of assets based on input data. This may however, require a large set of said input data.

3. **Cloud Platforms**

   Services that provide on-demand computing resources and data storage are essential in maintaining DT. Examples of such service providers are Amazon, Google, and Microsoft. Utilizing on-demand computing is advantageous for the user since the end-point user does not need to upgrade or maintain hardware and since it provides flexibility regarding the usage of computing resources.

4. **Simulation Software**

   Empowered by advances in computing power and data collection, it is now possible to create better simulations than before for testing of various what-if scenarios. Metso for example utilizes its own in-house flow sheet software HSC Chemistry for process simulation. Other examples from the process industry include Aspen Plus, OpenModelica, ChemCAD and SysCAD.

2.4 Challenges

Naturally, possibilities for creating and using DT also come with challenges. These challenges are not related only to technical difficulties, but also to social aspects. The following challenges were identified [13] [14]:

1. **Data Quality**

   Large amounts of data are needed to facilitate a useful implementation of a DT. It must be made sure that the data collected from the physical instance is of high quality and noise-free. Quality of the metadata in the virtual model is also of paramount importance as the DT progresses through its development stages.

2. **Data Security**

   All data collection and transmission of data comes with an increased vulnerability to security hazards, like cyber-attacks. Such issues must be addressed.

3. **Gaining trust from industry**

   A quite significant part of implementing new technology is establishing trust among potential users of the technology. If potential customers cannot be convinced of the benefits that the utilization of DT will bring, it will naturally

hamper the development of DT. Measuring and presenting the performance and capabilities of delivered DTs is key in establishing trust in the market.

4. **Expectations**

   Tying tightly to 3), it must be acknowledged that inflated expectations on DT can be an issue. While it is important to establish trust by ensuring that DT gives benefits and performs accordingly, too high expectations on the benefits is detrimental. This may lead to market observers thinking that DT is nothing more than an overhyped bubble. Keeping expectations on a reasonable level is essential for reaching "the plateau of productiveness" as indicated in the Gartner Hype Cycle in Figure 4.

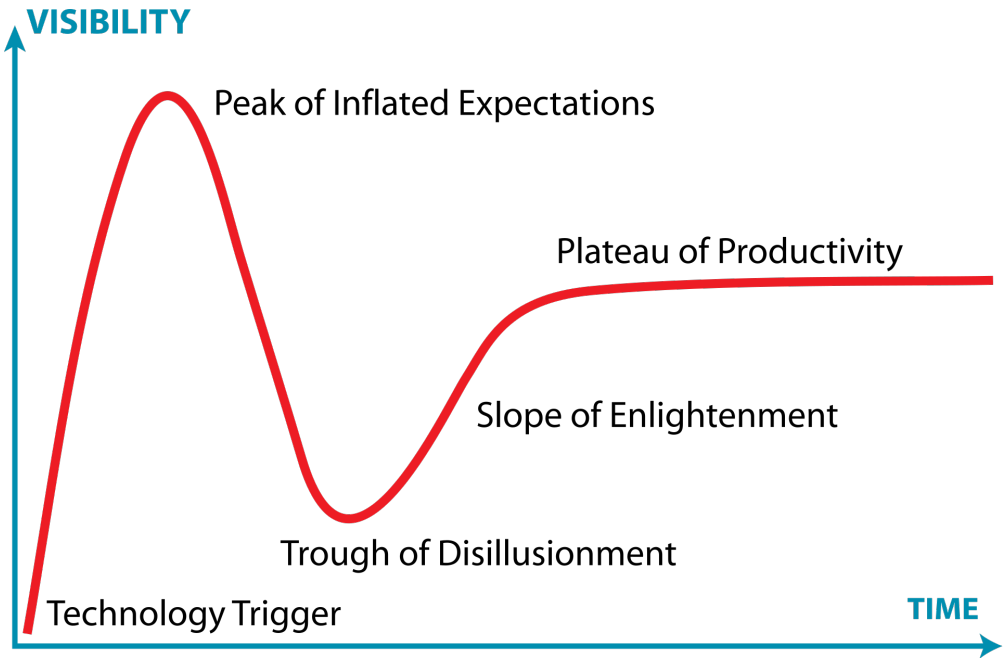


Figure 4: The Gartner hype cycle which describes the maturity, adoption, and social application of new technology [15].

The main challenge addressed in this study is related to assuring data quality in the form of equipment documentation in the DT. As the amount of data grows, quality issues become more apparent. Finding faulty data is also an issue with a large data set, as they are not feasible for a human to manually evaluate. Unless an automated solution is taken into use, the most meaningful way to measure data and metadata quality is to manually evaluate a statistically significant sample of data [16], or in this case documents. This comes with several disadvantages:

- Manual quality estimation is only valid at the sampling time. If the data set is updated the result is no longer accurate and must be reviewed again, taking away working hours.

- Stressing the last point, human labour is expensive and should arguably be used for other tasks than mundane data checking.
- In this way, outliers that are not an issue in the reviewed sample are impossible to find.

2.5 Modelling plants for a Digital Twin

Designing plant models for a DT application comes with several steps. The DT ideally has a 3D model for visualizing the plant itself and for highlighting things, such as equipment with a value out of bounds for instance as seen in Figure 5. A model that shows the connectivity of equipment within the plant is called a topology-based model or process connectivity model, which is the ultimate goal of the modelling [17].

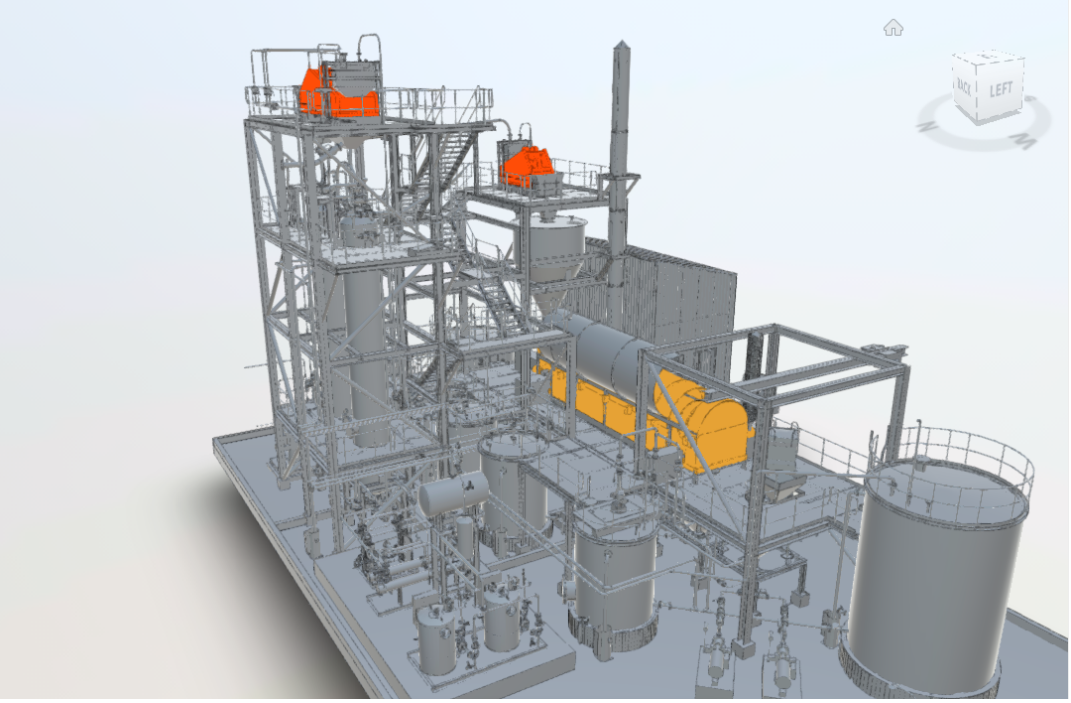


Figure 5: Example of a plant model where equipment is highlighted due to detected issues. [18]

The technology for visualizing plants is explained with a history review and a presentation of recent developments. Setting up a working information exchange is essential for building a DT, for which there are established standards and procedures that are showcased in this section. The basic principles of simulating a plant are also explained along with examples of research in the field.

2.6 Computer Aided Design

Computer Aided Design (CAD) refers to the use of computers for creation, modification, analysis, and optimization of a design. Previously, drafting and design was done on

paper, which eventually evolved into CAD software with the advent of computers in industry. The advantages of CAD include increased productivity of engineers, improved documentation and convenience when modifying an existing design. CAD is today a mature technology and already has been for a considerable amount of time, the term being coined in 1959 by MIT researcher Douglas T. Ross and the invention of 3D CAD being attributed to Renault engineer Pierre Bézier in the late 60s with his invention of the Bézier curve and the UNISURF CAD software [19] [20].

CAD has many applications in industry, facilitating the creation of high-quality drawings and documentation in engineering e.g., in mechanical parts, full product assemblies, infrastructure projects and large industrial plants.

Plant Design Software

When considering the plant model in the DT, the most important required features for the software to be used are different from the requirements on CAD software intended for producing drawings of parts to be manufactured.

The focus is generally not on manufacturing drawings, but more on providing an interactive overview of a large project. Examples of such software packages are AVEVA and Bentley.

2.7 Building Information Modeling

2.7.1 BIM models

BIM is a technology for supporting the Architecture, Engineering, and Construction (AEC) industries. BIM refers to accurate virtual models of a building being constructed digitally with the purpose of supporting design, construction and procurement activities through all phases of a project [21]. The equivalent of this for the mechanical engineering of products and components could be said to be Product Lifecycle Management (PLM).

Traditionally, facility delivery projects have been dependant on paper-based communication. This is a rather slow way of communicating and also causes significant delays and extra costs to the project if and when errors occur. Efforts were made to improve these procedures by employing web-based solutions and 3D CAD. However, even if the information is exchanged more swiftly the issue of conflicts caused by documents persists.

BIM addresses these issues by proposing a way of integrating multi-disciplinary data in an information model that contains all the necessary information about a building in a format that can be used by all involved parties throughout the whole life cycle of the building. Fundamentally, BIM differs from the traditional ways of creating, sharing, and using life cycle data of a building and is arguably a completely different approach.

2.7.2 BIM and CAD

CAD systems have since a long time transcended the stage of just generating files consisting of vectors, lines, and layer identification. Current CAD systems can generate files that also contain associated data and text. As users of the systems became more interested in sharing the data associated with a design in addition to drawings and 3D models, the vision of BIM was born.

The borderline between a BIM model and a CAD model might not be totally clear, which facilitates the need to also clarify what is not BIM technology [21]:

1. **Models containing only 3D data and no object attributes**

   Such models are used only for visualization and have no support for design analysis or data integration. These are for example, models produced in Google SketchUp which are only indicative of the design but contain no information necessary for analyzing the modelled structure.

2. **Models without parametric intelligence**

   This refers to models that cannot have their position or proportions adjusted due to the lack of parametric functions. Not having parametric functionality means that changing the model is labor-intensive and that there is no protection against creating incorrect views of the model.

3. **Models where dimension changes in one view are not automatically reflected in others**

   This allows for errors that can be difficult to detect.

Since BIM modelling is cloud-based, it means that engineers, architects, and stakeholders can collaborate virtually and communicate clearly over the whole life cycle of the project. All this facilitates a more efficient and less error-prone workflow. BIM modelling allows for many variables to be taken into consideration, such as sustainability and risk detection.

2.8 Standards and models for data processing in process plants

Standardization in industry is an effort to reduce confusion and costs that stem from unnecessary and undesirable differences in equipment, systems, materials, and procedures. Within standards, industry practice in areas such as safety, testing and installation can be documented. Official standards are generally published by professional societies, trade organizations and committees and strive to adhere to general consensus by experts in the field. Standards improve cooperation and workflow greatly on all levels, between engineers in an organization and between organizations and countries. To protect employees and the general public, it is common for standards to be incorporated into laws and regulations as a code. A code is a standard that has been legally accepted by a government agency and has been done so in order to ensure safety in a technical activity for both employees and the general public. Quite often, the case is that organizations that develop standards also develop codes [22].

There are several proposals for how to process and store data in process plants. Four such models are GPM (Generic Product Model) and the three international standards ISO 10303, ISO 15926, and IEC 62424. GPM is however developed specifically for nuclear power plants and has limitations in terms of interoperability [23]. ISO 10303 is a standard for exchanging product model data and ISO 15926 is a standard for representing and exchanging plant information with IEC 62424 filling a similar purpose. Due to the limitations of GPM, the international standards will be the focus in this section.

ISO 10303

ISO 10303: "Automation systems and integration — Product data representation and exchange", informally known as STEP (STandard for the Exchange of Product model data), is an international standard for exchanging electronic product data between different product life-cycle systems. STEP can be seen as a replacement for the American CAD data exchange format IGES (Initial Graphics Exchange Specification), even though the scope of STEP is not limited to solely CAD data. STEP is capable of handling a wide range of product types and life-cycle stages, and practically all CAD vendors have a built in STEP translator in order to ensure smooth transitions of models between different software [24].

ISO 15926

ISO 15926: "Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities" is a set of standards for sharing and integration of all information during the life cycle of a process plant. The standard consists of a generic description of a data model, information for geometry and topology, implementation methods for integration of distributed systems and methods for validation and usage of data [23]. ISO 15926 strives to serve as a bridge between computer systems by integrating information provided by these systems.

A common problem when utilizing different types of software in various life-cycle activities is that sometimes information is modelled implicitly, meaning that the software or the user is capable of understanding the information without details that have been left out. When bringing together different information sources, it is usually necessary to make this information explicit in order to avoid misunderstandings and provide the full picture. For example, we consider a system where an "impeller diameter" is represented as a primitive property of a centrifugal pump. The "impeller diameter" is quite clearly a property of the impeller, and not of the pump itself. The same maintenance cycles and performance metrics may not apply to both the impeller and the whole pump, so in that case, having the "impeller diameter" as a primitive function is flawed [25].

However, explicit data modelling might lead to a complexity which is too large. ISO 15926 attempts to solve the issue by utilizing data templates where each row represents the semantics of a statement like "Object has a property of X number of

units." For example, we consider a pump again by stating "Pump A has a design pressure of which unit is bar." In this case we have [26]:

- Physical object, which is Pump A.
- Class of indirect property, which is design pressure.
- Floating point number, which is the number of units in question.
- Unit, which is bar.

In order to account for changes in a plant over its life-cycle, ISO 15926 uses a four dimensional approach for representing life-cycle data. This means that objects are considered as existing in three dimensional space, and time. Physical objects having both spatial and temporal parts means that it is possible to present data about the object that was valid at any point in time. In this way, a historical record of the object is created [27]. ISO 15926 has a generic data model that is created by utilizing basic data entity types. The basic entity types are then ordered in a hierarchy.

IEC 62424

IEC 62424 - "Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools" is a standard that defines a neutral data format for exchanging plant information between CAE tools. The standard is also called Computer Aided Engineering eXchange (CAEX). CAEX works as a basis for a vendor-independent data format that is useful when transferring data within the engineering process, and especially useful with different P&ID creation tools [27].

CAEX is object-oriented and based on XML, and one of the key features is that it allows predefining patterns in the form of classes which can then be replicated several times. It was originally developed in a cooperation project between RWTH Aachen and ABB and is well suited for exchanging P&ID data as well as production monitoring and control system information [28].

Like ISO 15926, CAEX can be used for exchanging engineering data in plant projects. The differences between the two mainly come down to [27]:

- History: The ISO 15926 standard is based on STEP, while CAEX is a more modern approach.
- Life cycle: CAEX does not consider the complete life cycle of a plant, while ISO 15926 does consider this.

2.9 Simulation

Mathematical abstractions, or models, of a process can be used to gain understanding of a process at hand by simulating it. A process model is built by a set of equations that permits the user to predict the dynamics of a chemical process, assuming that

the model is accurate enough. Basically, all process models are built on the general conservation principle:

$$[Accumulation] = [Input] - [Output] + [Internal production]$$

By applying the conservation principle, a basic model structure of a chemical process can be developed by balancing the fundamental quantities mass, energy and momentum. A mass balance equation is developed with respect to total mass or the mass of individual components in a mixture. Under non-relativistic conditions, mass can never be created or destroyed, but only appear as other components with the example of reactants reappearing as products. Energy balance is thus simply governed by the first law of thermodynamics, meaning that total energy in a system always remains constant. Chemical processes are modelled by so called heat or enthalpy balance equations. Momentum balance is governed by Newton's law of motion and the idea is thus that generation of momentum is the result of the forces acting on a volume element [29]. Process models can be used for:

- Improving understanding of the process: Investigating how the process reacts to different inputs by running simulations. In a DT context, this means that experiments can be conducted on a plant or a product without modifying the physical counterpart.
- Training operating personnel: Simulations can be used to train plant operators on typical problem situations.
- Designing a process controller: The process model can be used for testing different controller configurations and tuning techniques [29].

Flowsheet simulation

Flowsheeting is the use of computer tools to perform heat and mass balancing, sizing and cost calculations for a chemical process. The name stems from that the initial embodiment of the concept of a chemical process is called a "flow sheet". Modelling integrated production processes cannot be done based on the information of single process units. Since there are recycle streams and applications of process control and optimization, each single unit can strongly influence the entire process. This facilitates the need for flowsheeting. Generally speaking, flowsheet simulation can be done in both steady-state and dynamic state systems. Dynamic simulations are however more challenging and computing intense [30].

Several software systems for flowsheet simulation exist on the market, such as Metso's HSC Chemistry, Aspen Plus, gPROMS Formulated Products and OpenModelica. In Figure 6 an example of a flow sheet in HSC Chemistry is shown.

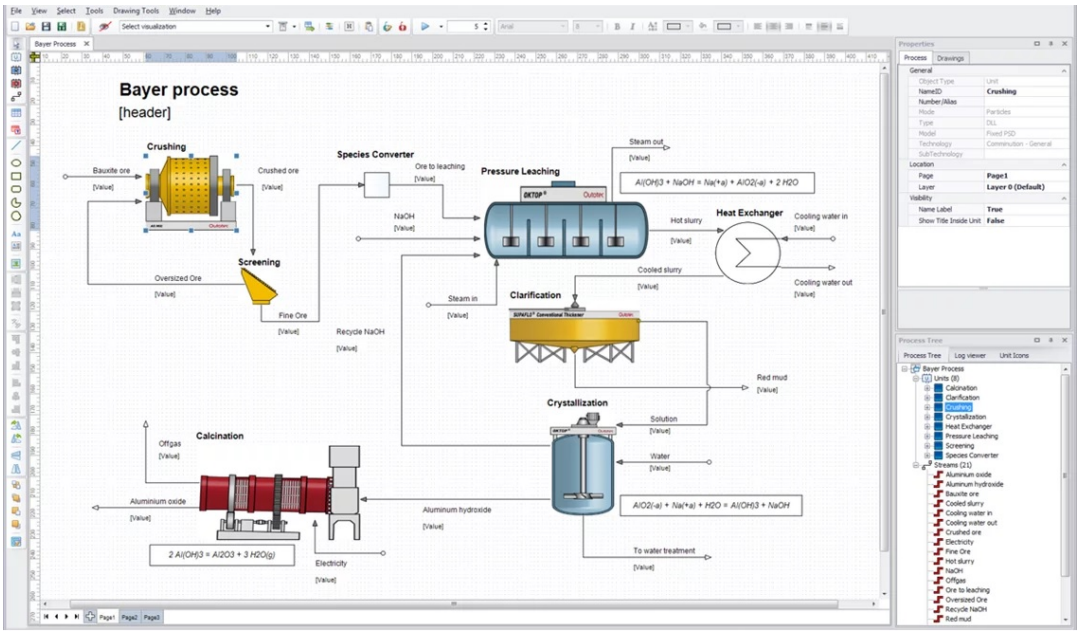


Figure 6: HSC Chemistry Sim interface while viewing a process flowsheet. [31]

2.10 Documentation management

Documenting all the modelled assets is vital for both the end-point user of a DT and the party building it. Without proper documentation, it is impossible to know when something was modified or who did it for instance. Typically, documents are uploaded to a Engineering Document Management System (EDMS) such as Meridian. Like the name suggests, the purpose of EDMS software is to keep track of documents in an engineering project, and changes to them. Another approach is for instance, to build an app on a SQL relational database, and provide a user interface in the form of a web application that stores data in XML documents. These concepts are briefly explained in Sections 2.10.1 and 2.10.2

In order to keep track of each asset, asset tags are assigned to them. What kind of information that is directly conveyed by the tag, and the tag naming system depends on the standard for the plant in question [32]. In the context of plant DTs, an asset tag is the means to identify the location, the type of item and the name of an entity present in the plant. This is not only limited to physical pieces of equipment or piping, documents related to the equipment also carry an individual tag.

2.10.1 SQL

Structured Query Language (SQL) is a computer sublanguage for storing, manipulating, and retrieving data in a relational database. A relational database stores data in the form of tables, with rows and columns representing data attributes and the relationships between data values. The user can interact with an SQL database by performing a *query*. Queries are in principle questions for the database, and if there is a valid response to the query in a database, SQL retrieves that data and displays it to the user.

In addition to retrieving data, the other mentioned functionalities of SQL are also accessed by sending queries.

SQL was invented in the 1970s as Standard English Query Language (SEQUEL), and is still in popular use among developers today because it integrates well with different programming languages, such as e.g., Java. The language is also quite user friendly since it uses simple English vocabulary as keywords, making the logic behind the queries intuitive [33]. Its advantages over older read-write APIs are that it can access many data entries with a single command, and it can access data with or without indices.

2.10.2 Extensible Markup Language

Extensible Markup Language (XML) is a standard for document markup. Basically, it is a generic syntax for marking up data with human-readable tags. This standard format can be used for a myriad of applications, such as web sites, vector graphics, genealogy, voice mail systems etc. XML documents can be interacted with by writing own programs in languages such as Python where libraries and modules can be leveraged to simplify parsing of the document. It should be clarified that XML is not a programming language. By itself, an XML does not do anything but can act as a format for instructions for a computer program.

Data is stored in XML documents as strings of text. The data is surrounded by a text markup that describes the data in question. XML's basic unit of data is called an element. The most important property of XML is that it is a meta markup language. This means that there is not a fixed set of tags and elements that are used globally in the language. XML allows developers to invent elements as needed, such as molecules and atoms in chemistry, or apartments and rent in a real estate application.

2.11 State of the art in process plant Digital Twins

In this section, the current state of the art in the field is examined and briefly explained. This is to provide an impression of the maturity of the technology and an example of applications.

Modelling brownfield plants

When discussing the topic of process plant DTs, one should distinguish between *brownfield* project and *greenfield* projects. A brownfield is defined as a currently operating plant, which has physical structures and software systems in place for which it is not always possible to obtain the information in digital format. Old plants e.g., pulp and paper mills might have machinery drawings and P&ID existing only on paper, which in turn requires many labour hours if such information is to be digitized [34]. Because of industrial practice it is also common that the existing information is not always up to date in for example, the case of minor components being changed without updating the plant documentation. This can mean that e.g., P&IDs are not up to date which obviously complicates things.

Greenfield sites in turn are defined as sites where no previous infrastructure exists, meaning that no such constraints exist. They are thus more flexible when it comes to design of the facilities and their systems. It should however also be mentioned that the fact that building infrastructure from scratch constitutes costs and extra financial risk. Unexpected on-site conditions can also contribute to delays and extra costs for a project.

Contemporary research on brownfield plants has examined the feasibility of utilizing technologies such as LiDAR 3D scanning and 2D image recognition to generate a plant 3D model and digitize drawings respectively. Even if several challenges were encountered this approach was deemed as promising. There have also been attempts at generating simulations models for existing plants automatically that have been regarded as successful [34] [35]. Such simulation models have been generated by utilizing automation control systems, the configuration of fieldbus systems and human machine interfaces (HMI) in use at the plants. This is possible since these structures are object oriented, meaning that all objects can be linked to functions. These data sources enable the creation of a reference topology model for the plant to serve as a base for generating the simulation model. With an established reference model, it is then possible to generate a low-fidelity simulation model with a modelling environment such as Modelica. A simulation like this is able to simulate actuators, sensors and simple equipment [35]. At this stage it can be connected to a Process Control System (PCS) to form a virtual plant.

A proposed roadmap for the workflow in such a process can be seen in Figure 7.

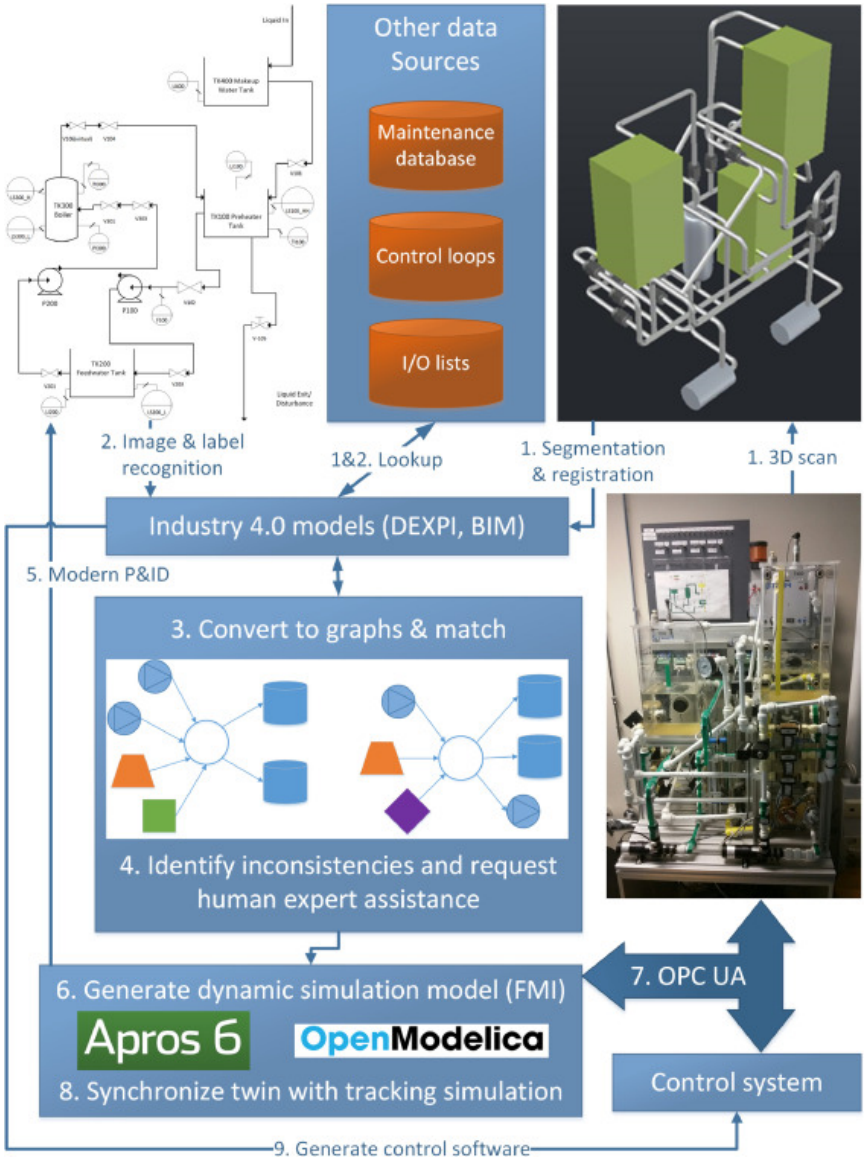


Figure 7: The workflow for generating a DT of a brownfield process plant proposed by Sierla et. al [34].

3 Material and Methods

3.1 AVEVA E3D Design

One example of software used for 3D Design of process plants is the AVEVA package. It offers the modules E3D Design, Engineering and Diagrams. E3D Design is a solution for the process plant, marine and power industries and is the successor to AVEVA PDMS, which is currently being phased out of the market. It is used for producing three-dimensional plant design models and is supported by AVEVA Diagrams where two-dimensional schematics such as Process Flow Diagrams are created. All information about a project, both 3D and 2D, are stored in hierarchical databases. The Engineering module provides utility to manage the data. A user can access all of the databases for the purpose of attribute queries, however, editing and writing data requires the user to have access rights for the specific database in question.

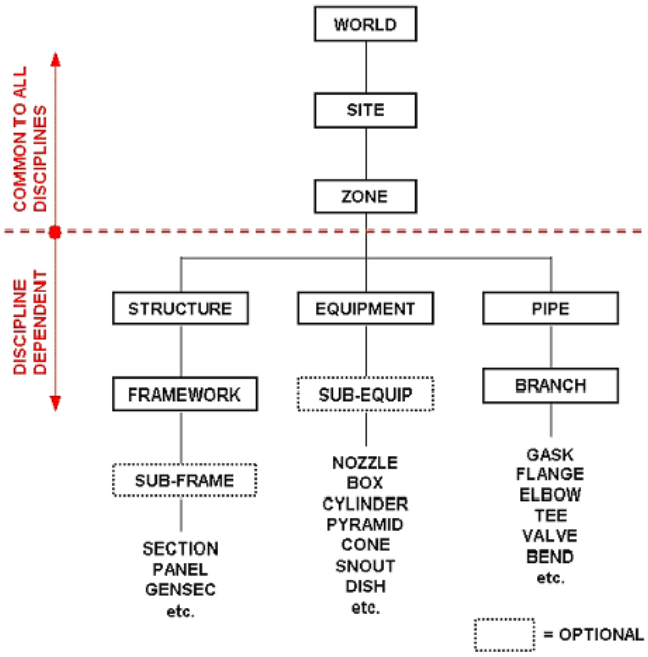


Figure 8: A simplified MODEL database hierarchy. Note that all database elements are owned by other elements, except for the WORLD element [36].

As seen in Figure 8, a project is built on a single WORLD element. A WORLD element can have any number of subordinate SITE elements, which is essentially a collection of project areas. One SITE element may contain the whole project, or just one part of a large project. SITE elements in turn contain ZONE elements, which store similar types of items for sensible referencing. An example of this would be for instance, a piping system.

Equipment (EQUI) modelling in E3D is built up by primitives, meaning that pieces of equipment are modelled with primitive shapes such as nozzles, cylinders, and boxes.

Sub-equipment(SUBE) elements are optional, and these are used to further divide an equipment element for practical reasons. SUBE elements can have their own

primitive elements.

Structure (STRU) elements are administrative elements, meaning that their purpose is to own FRAMEWORK (FRMW) elements. This is for practical reasons regarding modelling and reporting. FRMW elements store structural components in the model.

Sub-Framework (SBFR) elements are optional and can also own structural components similarly to how SUBE elements function. Structural components in turn are elements from a catalogue containing for instance profiles and panels.

PIPE elements are administrative elements similarly to STRU. The purpose of PIPE elements is to own BRANCH (BRAN) elements. BRANCH elements in turn are sections of a pipe, which is defined as having a known start and finish point. In E3D, the start point s named Head and the finish point is named Tail. BRAN elements can own components such as GASK, FLANGE, ELBOW etc. as seen in 8 which forms piping branches in the model.

E3D consists of two modules for design, one for modelling and one for pipework spooling, and two more modules for producing their respective drawings and piping isometrics [36].

3.2 AVEVA Diagrams

Diagrams is AVEVA's solution for producing and managing Process Flow Diagrams (PFD), Piping and Instrumentation Diagrams (P&ID) and Heating, Ventilation and Air Conditioning (HVAC) drawings. AVEVA's database communication philosophy is designed so that whenever a diagram is designed, it is stored in a schematic model database in the cloud. All data created in Diagrams is thus also connected to the other AVEVA software packages in use and accessible in the same way as the E3D file related to the same project.

Like similar software, Diagrams comes with the most common diagram symbols used in industry, also with the option of creating own symbols. The package's strength lies in its ability to assist users to create consistent diagrams by intelligent features such as rules and automatic actions, and in its integration with other AVEVA software. The latter facilitates the possibility to do consistency checks between a 3D model and a P&ID.

3.3 AVEVA Engineering and Documentation Transfer

AVEVA Engineering is a platform for different teams working on a project to access and reference each other's data, while still maintaining full control of their own data. It also has built-in tools for data consistency check between disciplines, such as P&IDs and 3D models. Customization is available with AVEVA'S own programming language PML and .NET APIs.

Equipment tags can be created in AVEVA Engineering, which can then be linked to AVEVA Diagrams and AVEVA E3D. Tags created in Engineering have to be correct, since they are the basis for everything else. The Engineering module is thus an essential tool for assuring that the site project has proper documentation. The

documentation for the plant equipment is then exported to a file vault for later use when the same documentation is linked to a DT implementation.

Information transfer process

AVEVA Engineering is accessed through a digital workspace in the case of this study, meaning that the software is not locally installed on user PCs. This greatly simplifies troubleshooting for support and also adds flexibility for users, since it eliminates the need for an employee's own workstation to have high-end hardware. Several users can also access the Engineering module and modify project databases simultaneously. Because of this, a "Get Work" function exists for updating the user's current session with the changes made by other users, as well as saving the user's own changes for others to see.

The Engineering module is linked to the Diagrams module by the Compare/Update function. When Compare/Update is used on a chosen set of documents, the user is showed a report of if and how the selected documents are matched to a corresponding item in the Diagrams module as seen in Figure 9. In the example shown, the visible tag names are indicated to be linked to and matched against a corresponding item in AVEVA Diagrams.

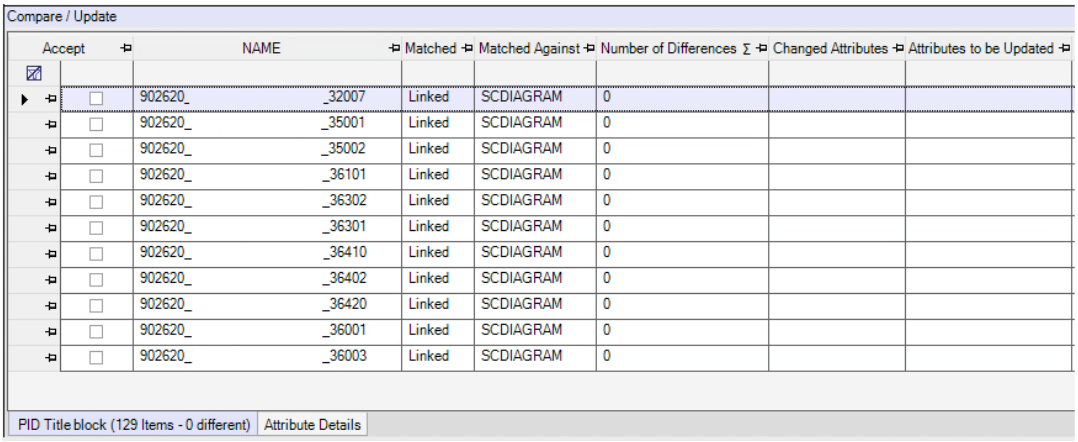


Figure 9: Example of a Compare/Update report on a database used for training purposes. [37]

From this view, it can also be selected which items should be re-linked and/or updated to the other modules. All-in all, the Compare/Update tool is one of the most commonly used functions in AVEVA Engineering projects and is especially important to use regularly when several engineers and/or teams work on the same project.

Meridian Document Import Tool

Currently, a partly automated and mostly manual work procedure is in place for handling the creation and uploading of AVEVA Engineering documentation to other platforms such as Meridian EDMS. Once all the documents are deemed to be up to

date, it can be proceeded to creating a Meridian Import source document. Meridian's Document Import Tool uses this source file as an import template to read certain documents from the same directory as the source file, which are then stored in a Meridian Document Vault. The quality of this source file is largely important for a document batch import to be successful.

Uploading and importing documents to a Meridian Vault requires a set of minimum metadata for each document, namely [38]:

- Path to the file that is be imported, relative to the computer running the Document Import Tool.
- File name to specify what the document is to be called in the vault, this can be generated in AVEVA Engineering.
- Revision number for assigning the first revision number to a document in Meridian, this is essential to make sure that the documents are up to date and accurate.
- Document type, however, this one is less important since it can be changed later.
- Document status, e.g., Released, In Review etc.
- Created by/Modified by, it is obviously beneficial to know who has modified a file in case questions arise.

Creation of the Document Import Source file

For the purpose of creating an export list, or the import source file, a Meridian Export List view is accessible as an option in AVEVA Engineering. This is a preset that contains the relevant columns for a Meridian DIT template. The template in question was created as a customized in-house solution at the company and was not provided by a software vendor.

The source file itself is created by importing the equipment tags to a "Title Block" sheet from AVEVA Diagrams with the Compare/Update function with an example in Figure 10. This list contains the attributes needed for creating a Meridian Export List and is edited at this stage. In this sheet, other necessary data that is not automatically imported by Compare/Update is added manually by employees. Such data is for instance, Document Type, Project ID, Plant Code, Revision Notes, Document Status and different types of titles relevant for making Equipment Data sheets at a later stage.

Name	Docum...	Document Title 2	Document Title 3	Document Title 4	Project Number	Plant code	Plant Unit Code
Document Name 1	P&I DIAGRAM	Document Title 2	Document Title 3	Document Title 4	123456	ABC1	PUC1
Document Name 2	P&I DIAGRAM	Document Title 2	Document Title 3	Document Title 4	123456	ABC2	PUC2

Figure 10: An example of a "Title Block" sheet containing a part of data that is to be added to a Meridian Export List [37].

As mentioned, this is manual labour with employees filling out cells by hand. However, the AVEVA interface is structured like a spreadsheet and comes with

functionalities found in e.g., MS Excel for speeding up the process. For instance, it is possible to fill in a Plant Code e.g., "HLP01" by typing or pasting it in once and then making use of a "fill handle" or "drag and drop" type of functionality in the same way as MS Excel. In this way, filling out the full sheet can be done significantly quicker. However, this also raises the concern that employees might fill in the value in rows that should not contain it. If the original cell which is then used to fill the rest is typed incorrectly, it naturally causes issues in the documentation.

In a later stage, equipment data sheets are generated with AVEVA Engineering according to different templates depending on the type of equipment. If the data in the data sheets is not consistent with the Import source file for any reason, it is a matter that needs to be investigated and corrected accordingly. This is further expanded upon in Section 3.10.

3.4 Overview of AVEVA NET

AVEVA provides Information Management products, where the information of a project is centralized in a repository called AVEVA NET WorkHub. The user interface used to access this information is called AVEVA NET Dashboard, from where all asset information about a process can be accessed.

AVEVA NET WorkHub is the environment where all the data in a project is stored. A key feature is that related data is linked to each other to provide context. AVEVA NET Dashboard is a web-based solution from where the user can access and interact with the information stored in WorkHub. The main purpose of Dashboard is to support users in organising, validating, and collaborating on asset data and documents. All-in-all, AVEVA NET acts as a platform for creating a DT of a considered asset during its whole life cycle and also supports ISO 15926 [39].

The data behind the AVEVA NET interface is stored in an SQL relational database, and the website is built from XML documents that store the data. This means that it is necessary to examine which is the most efficient way to check this data.

Linked data

As previously stated, a DT represents a physical counterpart with data from the physical counterpart. In order to improve operations in industry, DT applications for accessing metadata about a part or a plant are in use. By the click of a button, the user can access all documentation about an asset, such as name tag, function, drawings, procurement information, maintenance data and sensor data in a context. AVEVA NET provides this kind of functionality as seen in Figure 11.

A problem that needs to be addressed with regard to plant projects, is that tags are sometimes incorrectly named, not linked and that data and documents are sometimes missing. The reasons for this are mainly human errors, but sometimes also technical errors.

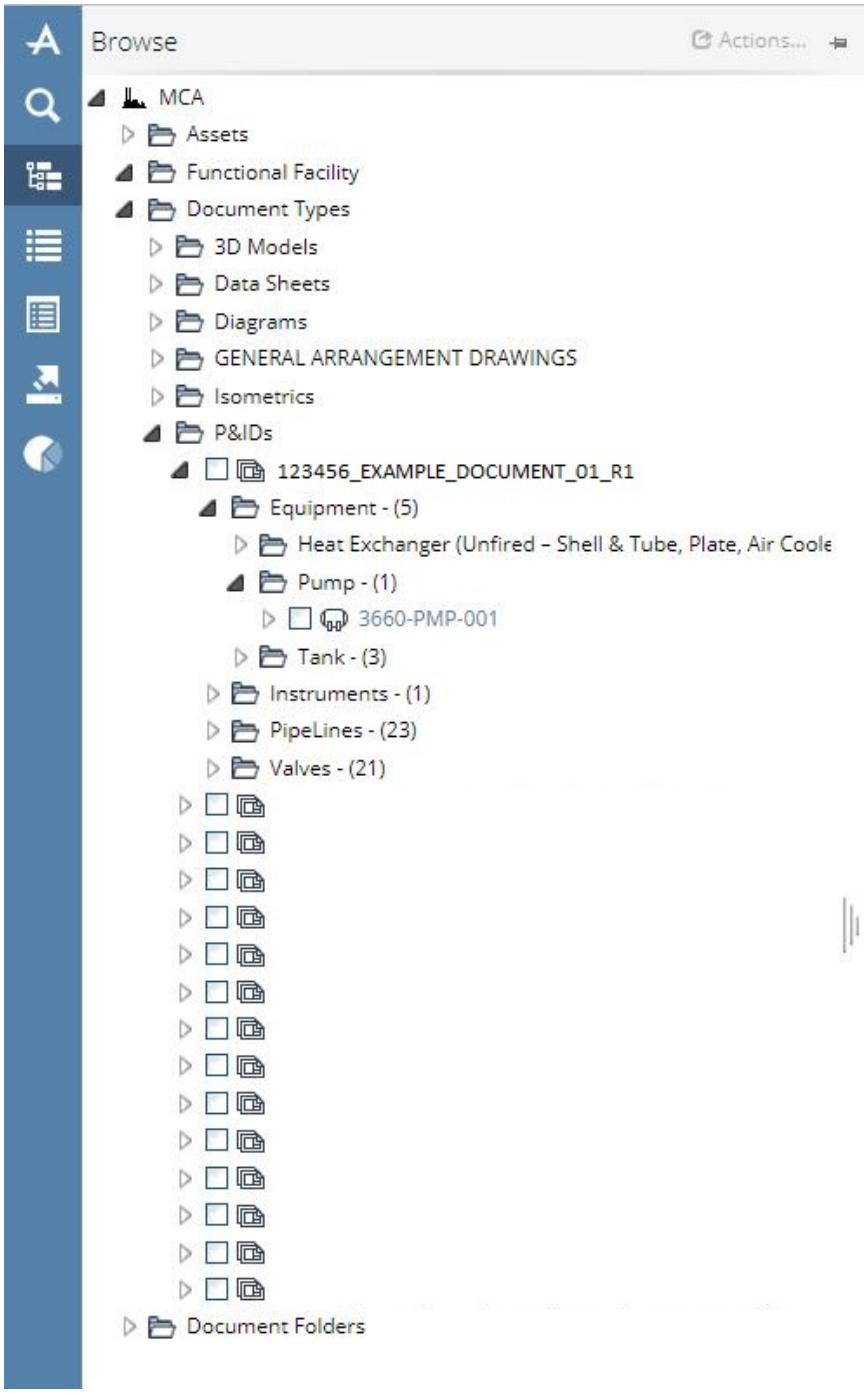


Figure 11: View from AVEVA NET's hierarchy tree browser, where the path of a P&ID has been opened. As seen in the figure, the equipment present in the P&ID can be accessed. [39]

AVEVA NET offers the possibility of generating reports of the data, drawings and equipment present in the database. Administrator rights are needed to access the "Export" functionality. An AVEVA NET demonstration project was used in this study in order to explore possibilities of fetching data.

3.5 Requirements Engineering

3.5.1 Background

Before starting a development task, it is necessary to clarify the objectives of the task at hand. The client or proposer of the task must answer the following questions as clearly as possible [40]:

- Which criteria is the intended solution expected to satisfy?
- What properties must it have?
- What properties must it not have?

After receiving answers to these questions, they are refined into requirements. A requirement was defined according to IEEE standard 610.12-1990 [41] as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in (1) or (2).

Even if the standard cited is a glossary for software engineering, an argument can be made that these definitions are general enough to use also in a non-software context.

When formulating requirements, it is necessary to clarify the goals of the project, and under which circumstances they have to be met. Requirements can be either *demands* or *wishes* which should be distinguished between.

Demands are requirements that must be satisfied in any case; the project is a failure and the proposed solutions deemed unacceptable if they are not met. These should preferably have clearly stated limits, e.g., "the side length cannot exceed 400mm".

Wishes are requirements that are of lower priority, but preferable to include in the solution if possible. They are thus desirable but may be omitted if it is deemed unfeasible to implement them in the final revision. Wishes should be distinguished between being of minor, medium, or major importance. [40]

3.5.2 Requirements Document

The purpose of the requirements engineering stage of a project is to produce a requirements document, which describes *what* a designed system needs to do, but not *how*. In other words, the required performance is described, but not the solution [42].

Both larger and smaller engineering projects start out with one or more likely several customer requirements. The first step is then as mentioned to distinguish between the demands and wishes in these requirements, and then produce requirement specifications.

The requirements will then later be used to *verify* and *validate* the solutions produced during the development process. Requirements must be thoroughly understood, meaning that an understanding of what is needed and what is possible is required. Simply translating the requirements into measurable quantities will not suffice [42]. The following questions must be asked:

- What is technically feasible within the time frame?
- What can the available workforce handle?
- Will the design actually address the identified issue?

Verification and validation

Verification is the term in use for checking performance against the set requirements utilizing one or several methods. The goal of a product/service/system development process is to provide a customer with something which is also accepted because it solves the problem(s) stated by the customer. Verification determines if the product, service, or system has been designed according to the set requirements. Examples of some common verification methods are [42]:

- reviews
- experiments, for example with functional models
- comparison to existing products
- simulations and analysis

Validation is in turn a specific case of verification that refers to the process of investigating if the product, service, or system that has been produced satisfies the set requirements by gathering objective evidence. The validation phase consists of testing the actual product and is carried out after the verification. Validation only cares about the output of the tested product or system, which is the main difference from verification [43] [42].

3.6 Tool requirements

Several problem statements were specified by the in-house personnel working with the AVEVA software package and by the document handling team. It was requested that ways of checking data both from AVEVA NET and from exported AVEVA Engineering documents is investigated. In this section, the requirements gathered for these tools are presented.

3.6.1 AVEVA Engineering document checker requirements

Discussions about the document handling led to the following demands for such a tool:

- The tools must check that the master data sheet matches with the filled-out documents.
- The tool needs to handle different templates.
- The tool should tell the user which relevant documents are found in a target folder.
- The tool must produce a report for the user that states the result of the check.

These requirements are further formalized and compiled for a better overview in Table 2.

Table 2: Requirements gathered for an AVEVA Engineering checker tool.

Requirement	Importance	Reasoning
Verify that the master data sheet columns match with their corresponding entries in the filled out equipment data sheets	Demand	This is the main purpose of the tool
Dynamic scanning to handle different templates	Demand	The tool is more useful in this case
Tool should alert the user if the master data sheet is incomplete	Demand	Useful information
A report should be generated for the user	Demand	Vital feature for analysing results
Tool should alert the user about files in a folder that cannot be found in the master data sheet	Wish	Useful information

3.6.2 AVEVA NET checker requirements

For automated processing of documents, the following demands were determined to be required for an AVEVA NET checker tool:

- All the elements (Especially process equipment, pipelines, valves, instruments) in a 3D model should be tagged properly, meaning that the element tags adhere to the naming convention of the current project.
- All the tagged items should be correctly linked with their respective documents. A kind of report where it registers the tags and their respective documents should be generated.

The following were determined as wishes:

- Metadata of the tags should be checked.
- Check if the correct sensor data is getting read.
- Find faulty tags and report them.

These requirements and wishes have been compiled into Table 3.

Table 3: Requirements gathered for the AVEVA NET checker tool.

Requirement	**Importance**	**Reasoning**
All elements should be correctly tagged	Demand	It is essential for the documentation that each element has a data tag that is consistent with the naming convention.
Tagged items must be correctly linked to their respective documents	Demand	It is essential that the correct documents can be accessed.
A report with a register of each tag and their respective documents should be generated	Demand	Reporting is essential to let the user know that everything is correctly linked.
Metadata of the assets should be checked	Wish	This is of secondary importance at this stage.
It should be checked if the correct sensor data is getting read	Wish	This is of secondary importance at this stage.

3.7 Checking stored data

This section provides an overview of current solutions which could satisfy the requirements stated in the previous section. AVEVA Engineering is a sheet-based way of handling information, and has built-in capability for exporting information to Excel (.xlsx). This opens up for building tools based on Excel read and write libraries.

AVEVA NET is built on an SQL relational database and the user interface fetches data from XML documents, which have been created with AVEVA Engineering. AVEVA NET offers the possibility to export data in Comma Separated Values (.csv) format and Microsoft Excel format.

3.8 Built-in AVEVA checks

AVEVA has inbuilt functionalities for linking data tags between AVEVA Engineering, Diagrams, and AVEVA E3D. The portal to access for such functionality is AVEVA Engineering.

For this purpose, AVEVA Engineering has the function Compare & Update. Compare & Update utilities enable AVEVA Engineering data to be compared against data created in the same project with other AVEVA products and/or external systems, and updates to be selectively applied as required. This makes the alignment with a new revision of data from another engineering department faster and more accurate. While these tools greatly improve the workflow as explained in 3.3, human mistakes can and do happen with fields that need to be manually filled, facilitating the need for checking that everything is consistent. Another problem is lag in the software, which can cause the data to be inconsistent.

An example of XML code below shows a simple, yet complete XML document.

```
<person>
    Thesis Author
</person>
```

This document has a start-tag called "<person>" and an end-tag called "</person>". The string contained within is called character data. Start-tags always begin with < and end-tags with </ and they are both closed with >. The tag names generally just describe the content inside and have nothing to do with formatting.

Now, below is a new version of the same XML document where more complexity has been added.

```
<person>
    <name>
        <first_name>Thesis</first_name>
        <last_name>Author</last_name>
    </name>
    <profession>thesis writer</profession>
</person>
```

There is still only one person element, which is the parent element, but there are also two child elements: a "name" element, and a "profession" element. The name element also contains two child elements of its own, namely "first_name" and ´last_name". The "name" and "profession" elements are siblings, the same also stands for the "first_name" and "last_name" elements. An important feature of XML is that a parent element can have multiple child elements, but a child element can only have exactly one parent element. In practice this means that a child element is completely enclosed by a parent element. This hierarchical structure has an element called "root element" at the top. The root is the one element in an XML document that does not have a parent. XML documents for a structure that is commonly referred to as a "tree", with all the other elements branching from the root element [44].

3.9 Checking the AVEVA NET data

As set by the requirements, several aspects of the data need to be checked. For this, several methods are in place, and some are already used extensively. Regular Expressions (RegEx) are used for checking tag naming conventions, however, this comes with some issues in its current implementation; strings that happen to fit the RegEx pattern are caught, even though they are not proper equipment tags.

Other issues include lack of documentation linked to the equipment. This means more precisely that it is needed to check whether an equipment tag is referred in any document at all. In a later stage, it will also be examined whether it can be checked if the tags are referred in likely the correct documents.

Regular Expressions

Regex is the main workhorse for matching character string pattern in a text with a target pattern. They have been so already for a long time, entering popular use in 1968 after being conceptualized by computer science and mathematics researchers in the early 1950s. The simplicity of the syntax made Regex very popular for such applications, even if it does not come without problems and difficulties with implementation [45]. Below is a simple example of a Regex pattern that finds equipment tags adhering to the naming convention of a four-digit location code, dash, a three letter equipment code, dash and a three digit serial number, e.g. "3000-PMP-001":

```
\d\d\d\d-[A-Za-z][A-Za-z][A-Za-z]-\d\d\d
,where \d is an unspecified digit and [A-Za-z]
any letter A-Z (both capitalized and small)
```

Difficulties with Regex typically include interpreting if the search pattern is indeed correct, which is not always easy for a human even though the syntax is relatively simple. Below is an example of a subjectively complicated string pattern that is no longer a trivial case such as the one above [46]:

```
<\s*[aA]\s+[hH][rR][eE][fF]=f\s*>\s* <\s*[iI][mM][gG]\s+[sS]
[rR][cC] =f\s*>[^<>]*<\s*/[iI][mM][gG]\s*>\s*<\s*/[aA]\s*>
```

Even if there are Regex checker tools available, it is commonly far from trivial to read for a human and mistakes can and do easily happen. Its simplicity and straight forward effectiveness on the other hand keeps it popular for use in all sorts of applications.

String Matching Algorithms

It was decided that other methods than Regex should be tested for determining if a tag is correct, and for this, the concept of fuzzy/approximate string matching in the context is examined. This refers to algorithms that are used for finding similarities between two strings, or sets of strings. The fundamental idea here is to find strings that approximately match the pattern of dictionary strings, or finding which strings are similar to the dictionary strings. This is different from semantic matching, which matches strings by their meaning even though they may look very different. The hypothesis that was tested is: Can a set of correctly named tags be used for training a model to recognize whether a tag is correct, meaning that it adheres to the naming convention without containing errors in the form of incorrect or missing characters.

In order to detect correct equipment tags from the tag list, Python tools for fuzzy string matching were examined. For this purpose, the PolyFuzz Python library by M. Grootendorst is leveraged [47].

The tested methods in this study were:

- Levenshtein Edit Distance, which is a technique that calculates the number of changes that has to be made to go from one string to another. The input tags can be matched with the reference list of generated "fake" tags, which generates a similarity index. Levenshtein distance is commonly used as a synonym for Edit Distance since it is the most common metric. Levenshtein allows insertion, deletion and substitution of characters.

- With Damerau-Levenshtein (DL) distance, the possible edit operations are: substitution, insertion, deletion and transposition. The difference between Levenshtein and DL is thus that DL allows transposition, namely swaps of adjacent characters [48].

- Jaro Winkler Similarity, which is another Edit Distance measure that only allows transposition of characters.

As mentioned, a script was made for generating data sets of different sizes consisting of correct tags. This was realized by obtaining a list of three letter equipment abbreviations, and then randomizing equipment numbers. In this way, sets of 1000, 10000 and 100000 tags in the correct style were generated for testing purposes. The reasoning for this was to use these correctly named tags as a reference to try and fit a real data set on. A real set of 5213 tags containing both proper and faulty tags was used to evaluate the success of the tests. Results are showcased in Section 4.1.

Autodesk solutions

As current in-house DT development is built on Autodesk Platform Services, contact was established with Autodesk sales personnel regarding potential solutions for data checking. Different Autodesk solutions were proposed, such as readily made model checkers. These were for the most part however focused on checking BIM models, and were not necessarily compatible with the AVEVA software package. A model information checker that could potentially solve the issues was proposed, however, Autodesk personnel claimed that for this a consultation would have to be ordered and paid for. Thus, it was decided to move on and neglect Autodesk's solutions as this would surely prove to be time-consuming to an unknown extent and possibly costly.

Aspose.3D for Python

The package Aspose.3D for the Python programming language was investigated in order to find out its capabilities for model checking. Aspose.3D for Python via .NET is a Gameware and Computer-Aided-Designing (CAD) API to manipulate documents without any 3D modelling and rendering software dependencies. API supports several file formats such as FBX, STL, OBJ, 3DS, U3D, DAE, glTF, DRC, RVM, PDF, AMF, and PLY. It allows users to create, read, convert, modify and control the substance of the mentioned 3D document formats [49]. The only useful functionality in Aspose3D found that was relevant to this study is that it can be used to view AVEVA RVM files with its data tree in for example a PDF editor. However, there seemed to be compatibility issues with the AVEVA RVM files used and nothing useful could be extracted with this approach. Thus, the package did not seem promising and was abandoned for this study.

3.10 Checking AVEVA Engineering data sheets

When providing equipment documentation for a customer, document templates are filled out by loading data from AVEVA Engineering by utilizing the Import source data file. What is important to consider at this stage is that the data coming from AVEVA Engineering needs to be correct and consistent, since it is the Master data and everything else will be based on it. This facilitates the need for checking that nothing went wrong when generating this. The documents are loaded into Meridian with a developed import script as explained in Section 3.3. The script finds the correct documents listed in the ImportData file, and then uploads them to Meridian.

There is a need to make sure that the data in the ImportData sheet and the corresponding data in the equipment data sheet to be submitted match, typically these can differ due to a human error at some point in the work process, or server lag especially when multiple users are attempting to update the database simultaneously.

4 Results

4.1 Tag Checking

Tests were run with sets of 1000, 10000 and 100000 generated tags. In this section, the results are presented and analysed. A set of 5213 tags from an AVEVA Engineering project was used for the trials, since the AVEVA NET demo project that was used for other tests was deemed to be too small to run proper tests. The data set used contained a lot of noise, and far from every tag was matching the target pattern set for these trials. There were a lot of placeholders and copies, namely tags in the form of "Copy-(1)-of-TAG" which should not be identified as a correct tag. The real tags were matched against the generated tags and received similarity ratings. For validating which tags are correct, a customized RegEx matching macro was implemented in Excel which returns "match" or "no match", depending on if the pattern matches or not. This made it possible to find a cut-off ratio where the checked tag can be confidently assumed as correct. As seen in the results below, several RegEx non-matches are present alongside matching tags, and this typically means that they were very similar to the matching ones, likely having a typo due to a human error. In the plots, the tags have been sorted by their similiarity ratio before plotting.

Levenshtein and Damerau-Levenshtein Edit Distance

For identifying if the strings are identical, similar or completely different, a scale between 0 and 1 is used, where 0 indicates that the strings are identical and 1 indicates no similarity at all. Thus, a lower rating is better here. In this case, it was clearly visible at which threshold ratio the tags lined up with a RegEx check as seen in Figures 12 and 13. These two models produced identical results. The results were not noticeably different depending on how large the fitted data set was.

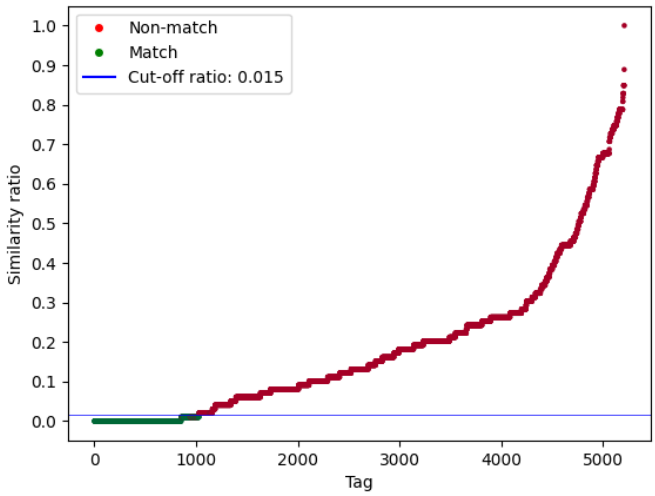


Figure 12: Levenshtein Edit Distance result.

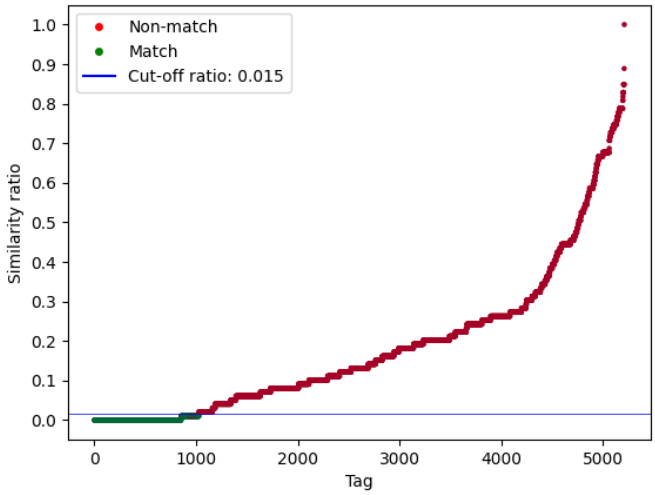
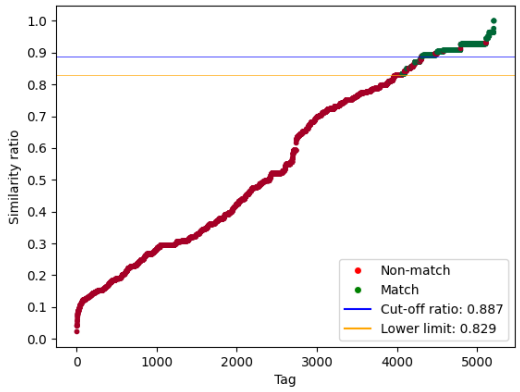


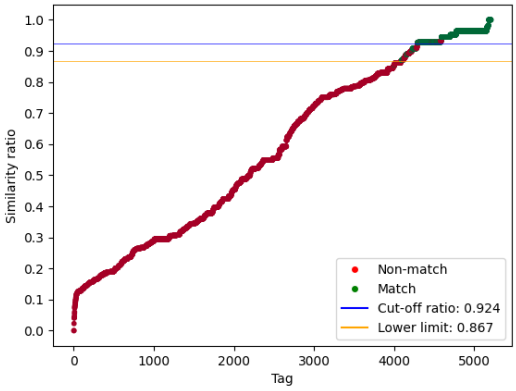
Figure 13: Damerau-Levenshtein Edit Distance result.

Jaro Winkler Similarity Edit Distance

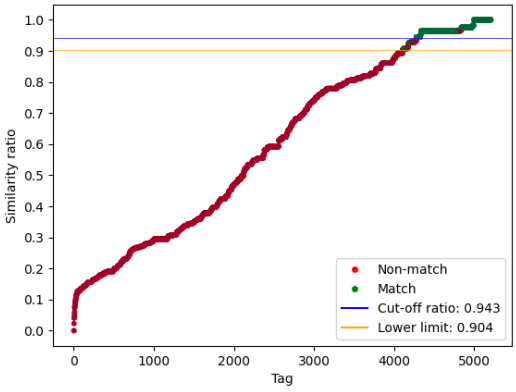
In this section, the results from the trials with the Jaro Winkler model are showed. The tags were first sorted according to their similarity, starting from the lowest and ascending. After this, they have been scatter plotted with the similarity ratio on the y-axis. In this case, the similarity scale is opposite compared to the ones showed in Figures 12 and 13, meaning that 1 implies a perfect match and 0 implies no similiarity.



(a) Trial with Jaro Winkler Similarity and a dataset with 1000 generated tags.



(b) Trial with Jaro Winkler Similiarity and a dataset with 10000 generated tags.



(c) Trial with Jaro Winkler Similiarity and a dataset with 100000 generated tags.

Figure 14: Jaro Winkler Similarity trials

As seen in Figures 14a, 14b and 14c, the certainty with which it can be assumed that a tag is correct increases with a larger set of data that the real tags are fitted on. A lower limit has been pointed out, which is the similarity ratio at which the first matching tags are found. It is also possible to define a more specific cut-off ratio with a larger data set, above which tags can be assumed to be correct albeit with a few isolated outliers. Above the specified cut-off ratio, tags can mostly be assumed to be correct and the few outliers can be examined. Typically, the problem with the outliers is that they have an extra character at the end, such as "_".

Summary

The verdict of these trials was that this could certainly be possible, however, a conclusive solution that definitely works with a high confidence interval was not found with these trials. It was decided that this is left for further research with a recommendation of acquiring a more robust theoretical background of the topic before conducting new experiments.

4.2 AVEVA NET Export checker tool

For checking the data, it was decided to use the Python programming language to build an app that carries out the data checks. The decision was made based on Python's vast library resources and its relatively simple syntax and readability. For realizing the set requirements, possible solutions were investigated. The tool was developed using Python version 3.11 (latest release at the time) [50]. The complete repository will be uploaded to Github and made available upon request.

4.2.1 Exporting data from AVEVA NET

In order to export information from AVEVA NET, an Export Definition (ED) needs to be created. A Default ED is in place, but in order to obtain exactly the desired information it was more practical to create custom EDs since the default one includes irrelevant information. In the first trial, documents were fetched with an ED that searches for ID tags and any related document as seen in Figure 15. After defining the export file's contents, it is possible to do a wild card search for all Equipment for instance, and then export it as either a .csv or .xlsx file. This particular ED returns a .csv file with two data columns, one with the Equipment ID Tags and one with the linked documents where the ID Tags are referenced.

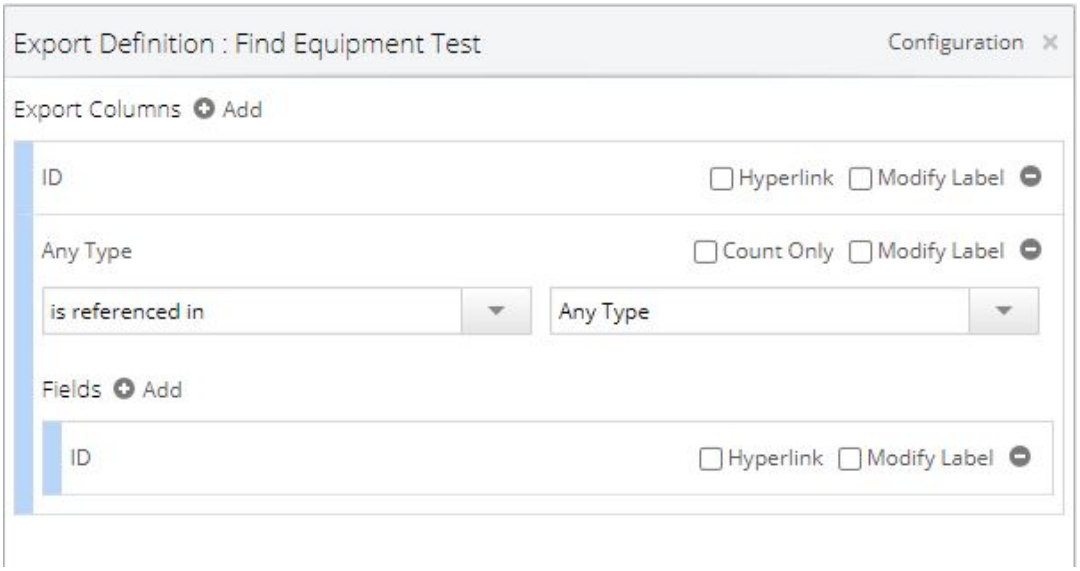


Figure 15: An Export Definition in AVEVA NET that fetches ID tags and any kind of document that is related to that tag [39].

4.2.2 Reading CSV reports

AVEVA NET offers the possibility of exporting selected tags and related data as either CSV files or MS Excel files. After trying both approaches, it was noted that CSV files are considerably faster to process especially when handling large files. Based on this fact, it was decided that the application was to be built as a CSV processor. The idea behind the procedure was to:

- Read the csv file which contains equipment tags and their respective linked documents
- Checking the RegEx pattern for each tag and appending found faulty tags to and adding them to a separate report sheet
- Checking if the tags are not linked to any documents at all, and adding them to a separate report sheet

For reading .csv files, Python 3 comes with a built in module called "csv" and it is thus not necessary to install separately. The same thing was true for the "re" or Regular Expressions module. In Figure 16, the process of coding the checker in Python is visualized.

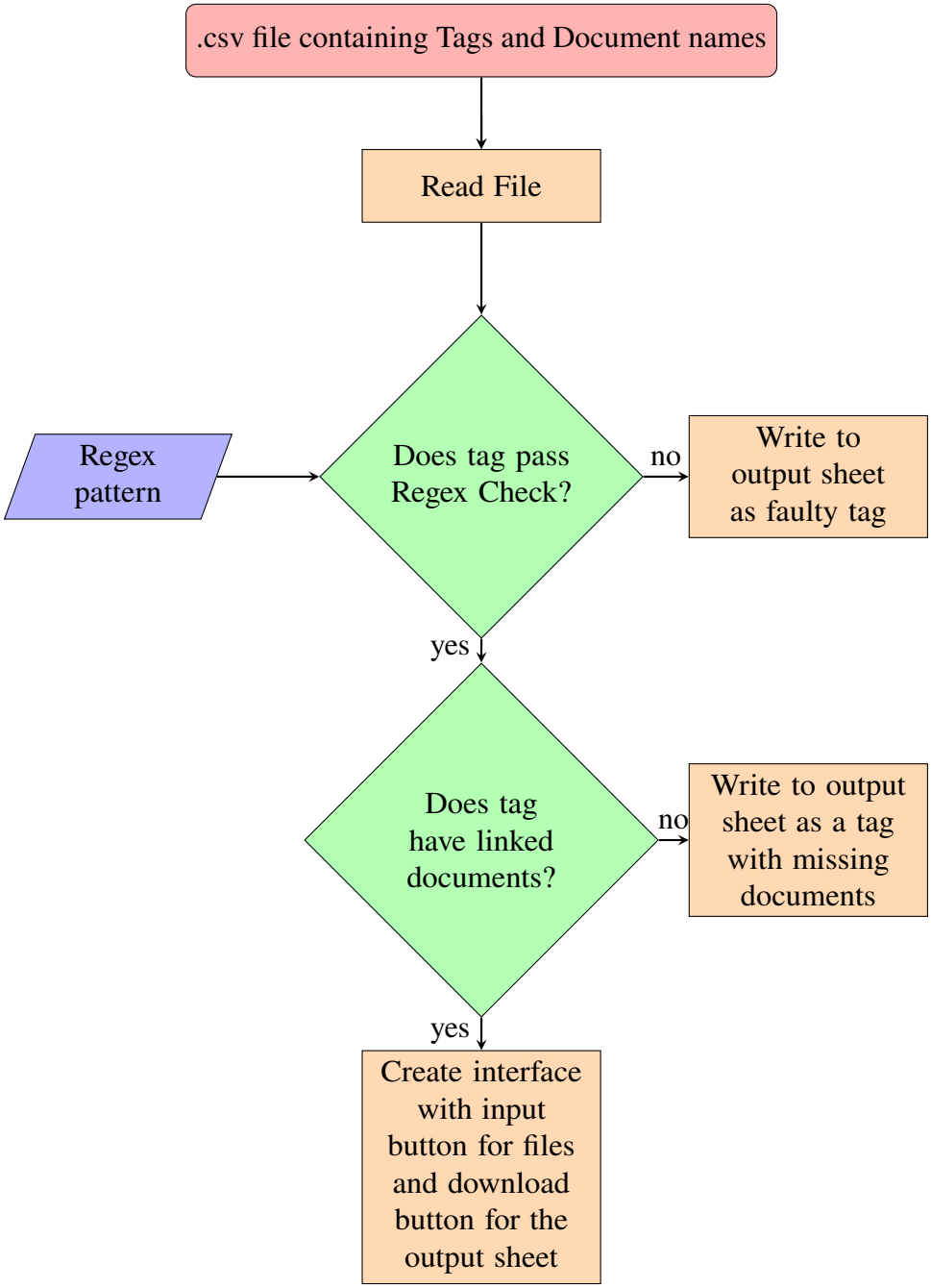


Figure 16: Flow chart of the program for checking AVEVA .NET exports.

4.2.3 Web application

For the purpose of testing and demonstrating the Python AVEVA NET checker application, a web application was developed to make it simple and straightforward to input files for checking. The web-based application was realized by utilizing Flask API for Python. In Figure 17 the user interface is described. The web-based application allows the user to upload a .csv file that has been exported according to the previously

defined ED and shows a confirmation when the file has been uploaded. When this is done, the web page allows the user to run the Python script checks and download a report file in Excel format. The decision to produce an Excel report was for the purpose of allowing the user to analyse the report data with a convenient toolbox, such as MS Excel or Python modules. The interface was made to be quite simple, with buttons for browsing the file explorer for the correct file, uploading the file and opening a new window to see if the correct file was indeed selected.

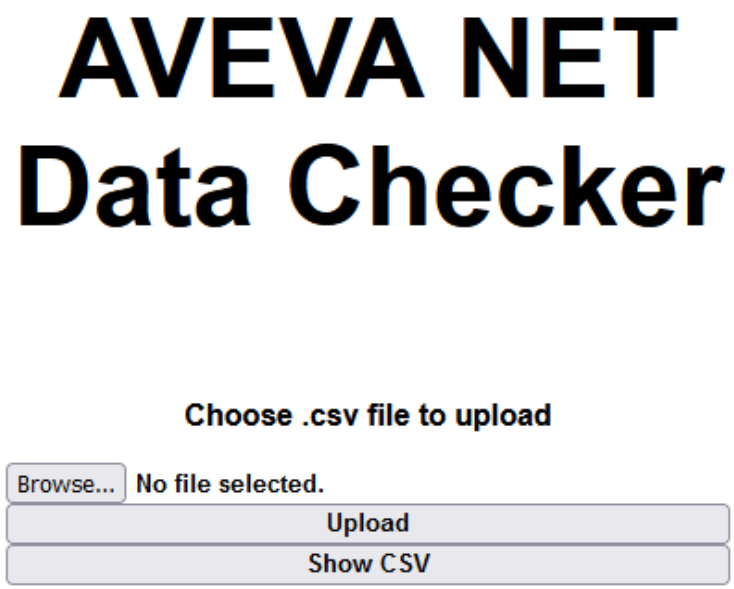


Figure 17: Web application interface.

4.3 Document checker tool

In this section, the procedure for creating the checker tool is explained. Firstly, the train of thought is presented in text, and further down flowcharts are shown to better clarify the process. The tool has been tested with two different templates to validate that it works dynamically, and not only with one test template. The result indicated that the checker indeed works dynamically and can thus be used with different projects.

4.3.1 Processing the files

For reading and processing the files, the Python modules Xlsxwriter, Pandas and Openpyxl for reading and writing to Excel files were used. Otherwise, built-in Python functions were leveraged for checking text strings in Excel cells. The aforementioned modules have slightly different advantages depending on which operation that needs to be done to a file, such as the case of importing the values from a column with a certain header, where the Pandas module is convenient. Xlsxwriter in turn is convenient when

writing to a file, while Openpyxl has more advanced features and useful functions for analytics.

While attempting to read the files from a sample project, it was noted that the sheets are protected after being created with AVEVA Engineering. In this case, it was impossible for the created program to open and read the files. A workaround had to be developed for this to work, and a functioning solution was found. It was noted that after manually opening an Excel file and saving and closing it, the program could access the files. As a solution, a PowerShell (.ps1) script was written to open, save and close every Excel (.xlsx only) file in a specified target directory which proved to solve the issue. Below is a pseudocode version of the script:

```
folderPath = "path"
initiate Excel app
for each file in path -Filter ".xlsx":
    Create workbook
    workbook.Save()
    workbook.Close()
close Excel app
```

4.3.2 Application structure

After formalizing the required functionality for the checker tool, the process went on to develop functions that perform the required tasks, and a user interface. In this subsection, the design flowcharts for these implementations are shown. Explanations are provided for each specific function in their own subsections. The functional layout for the tool as a whole is shown in Figure 18.

To make the application accessible without having Python installed on the machine where it is running, it was converted to an executable (.exe) file with the "Auto PY to EXE" converter, which is in turn based on PyInstaller [51]. This converter offers the choice of converting the Python script either to a single directory containing all the dependencies, or a single .exe file. Mainly, the difference is that a single .exe file will be somewhat slower on startup, since the executable file needs to create a temporary directory for all the dependencies. When running the app in a shared desktop environment, it was noted that using an output directory contrary to a single .exe file was considerably faster. A series of trials concluded that the single executable file starts up with about a factor of 2 to 3 times slower as compared to the executable file in a one-folder bundle.

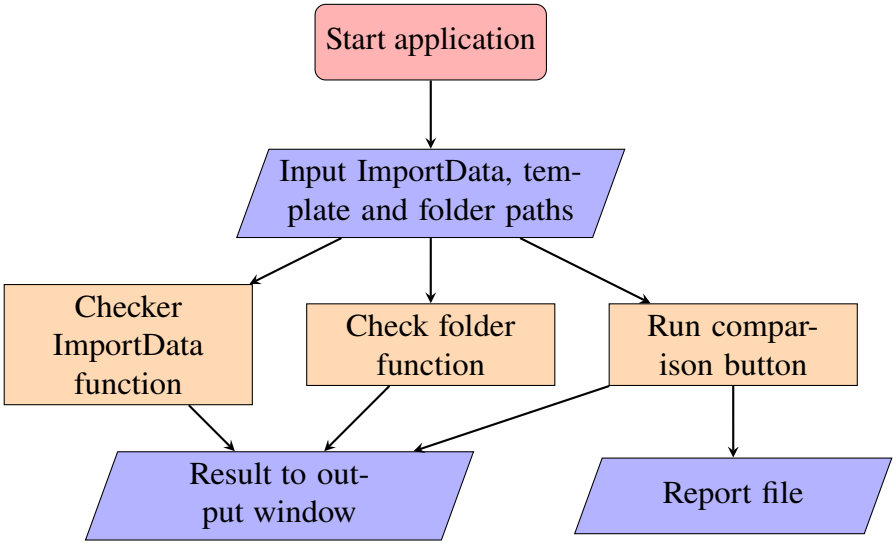


Figure 18: Structure of the developed tool.

Check ImportData

For checking the AVEVA Engineering master data file generated for Meridian Export, the logic was the following:

- Define which columns in the file that need to have a value, meaning that there cannot be empty cells in these columns.
- Compare the UniqueName column values to the Excel and PDF file names on the same row in order to make sure that the file names match. The names of the Excel files listed in the ImportData sheet needs to match with documents in the folder containing documents that are to be uploaded to the Meridian vault.
- Display discrepancies to the user in an output window.

Below in Figure 19, the procedure for the function structure can be seen. The ImportData source file was parsed and handled with the Openpyxl module. File name comparisons were done by Python's built in string handling. The function made for this checks that the UniqueName, Excel file name and PDF file name for each equipment tag is consistent. It was requested that the function returns output to the user only if there is a mismatch. For reading the mandatory columns, the Pandas module was used because of its handy functions for such purposes. This check will only alert the user in case cells in mandatory columns contain no data at all, as this was requested by end-point users.

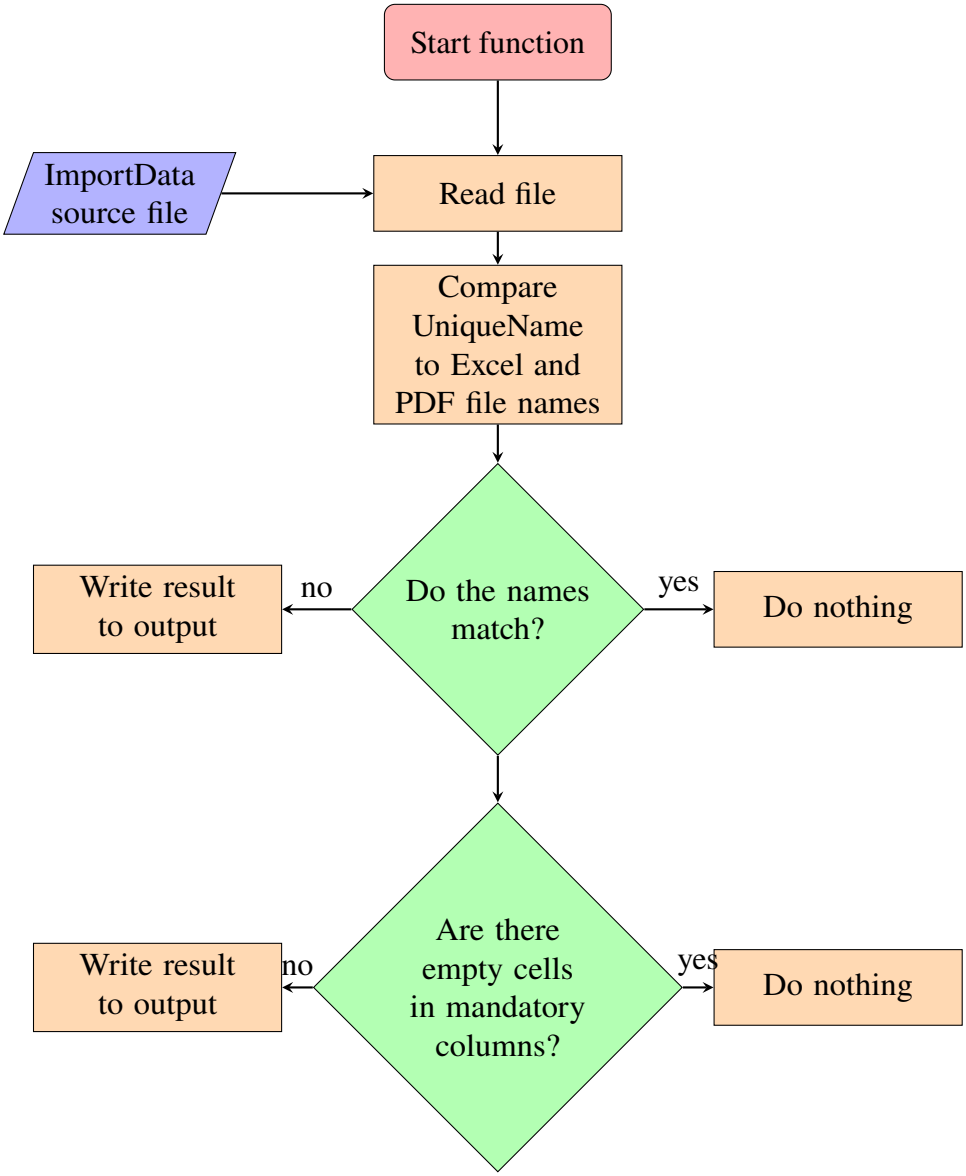


Figure 19: ImportData check flowchart.

Folder check

It is also necessary to validate that the documents in the given folder can be imported with the selected ImportData sheet. This is validated by the following procedure:

- Reading the ImportData file and importing the column containing Excel file names to a list.
- Accessing the folder path and doing a simple comparison to the aforementioned list to find out if paths to the files in the list exist.
- Reporting if there are files that are not listed in the ImportData file.

For checking the folders according to requirement set in Figure 3.6.1, the logic illustrated in 20 was applied. To examine which files do exist in the folder but are not present in the ImportData sheet, the document names listed in the ImportData sheet are first imported to a list with the Pandas module. After this, the target directory path is accessed, and a comparison is run to see if the file names have a path in the target directory. During the check, files that exist in the folder but not in the ImportData sheet have their filenames printed to the output window to notify the user. The purpose of this alert is to confirm that all the files that are to be uploaded to Meridian indeed exist.

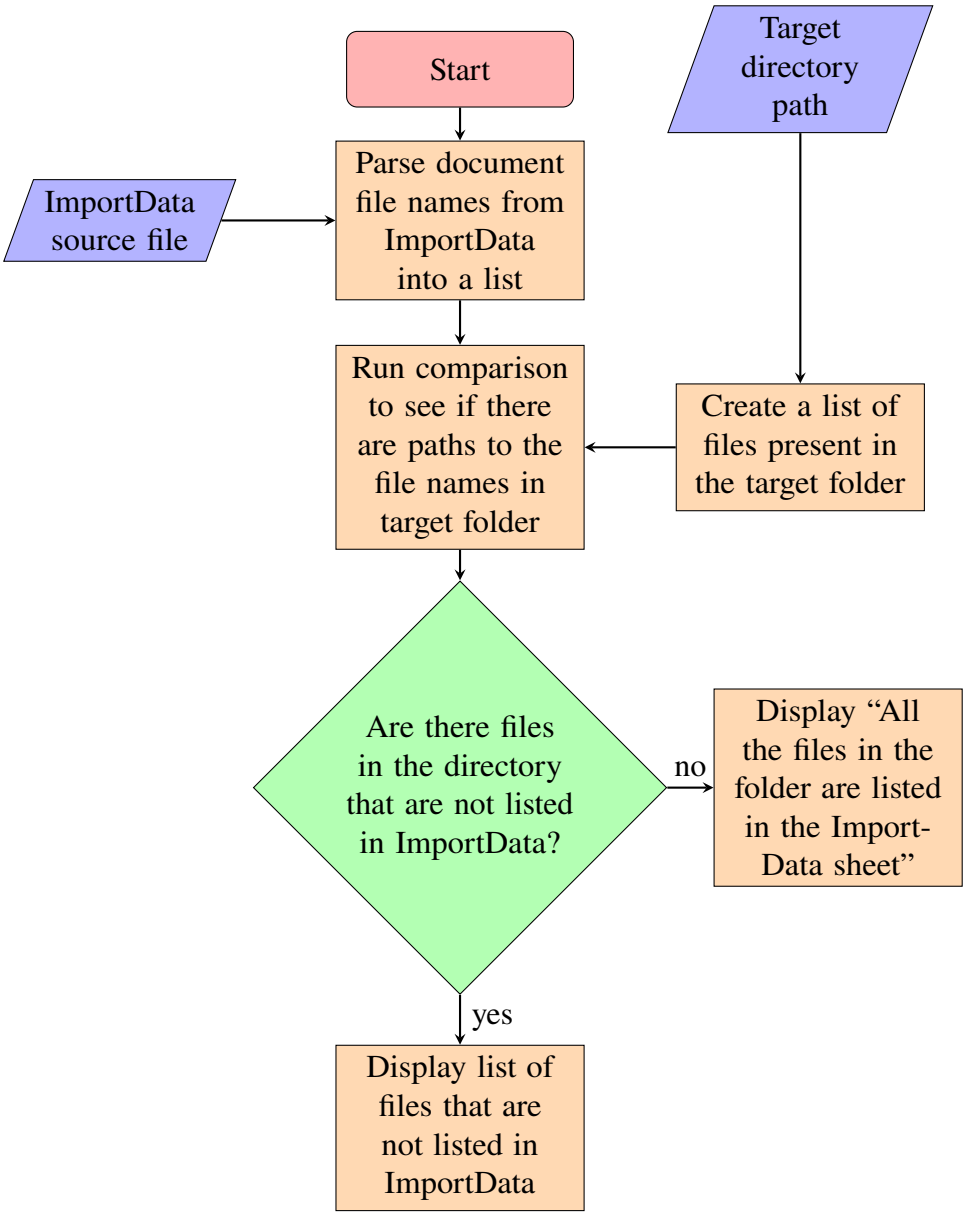


Figure 20: Folder check flowchart.

Compare to template

The third and last function designed for the tool in this iteration is the "Compare to template" function. The logic behind this tool has been illustrated in Figure 22. Firstly, the structure of the templates is explained before the logic flow chart is shown. Since the documents are generated in accordance with a given template, the train of thought for solving the issue was the following:

- Create a function that finds the cell locations for each cell beginning with the '#' key and stores them as a list.
- Create a function that fetches the relevant filled templates (Excel files) by utilizing the file names from the Master data sheet.
- Use the field names present in the template to locate the appropriate columns that use the same field names as headers.
- Compare the '#'-field values to each corresponding cell value for the document in question, meaning that for each file name, the list of cell locations is used to fetch the values for the currently checked file and is then compared to the correct cell in the Master data sheet.
- Log the complete comparison to a file accessible later, as well as a report spreadsheet that highlights the discrepancies found while running the check.
- Implement a link directly in AVEVA Engineering so that the checker application can be accessed inside the program.

In Figure 21, an example of a data sheet template is shown. The fields that are to be filled with data from AVEVA Engineering are marked with a '#' sign as the first character in the cell. All the fields marked with '#' in the template correspond to a column header in the Master ImportData file. By matching the column header with the correct row for the document name in question, the cell that should contain the same value in both the template and the Master file is found. A separate document is created for each piece of equipment as both .xlsx and .pdf files, which contain exactly the same information in the same layout.

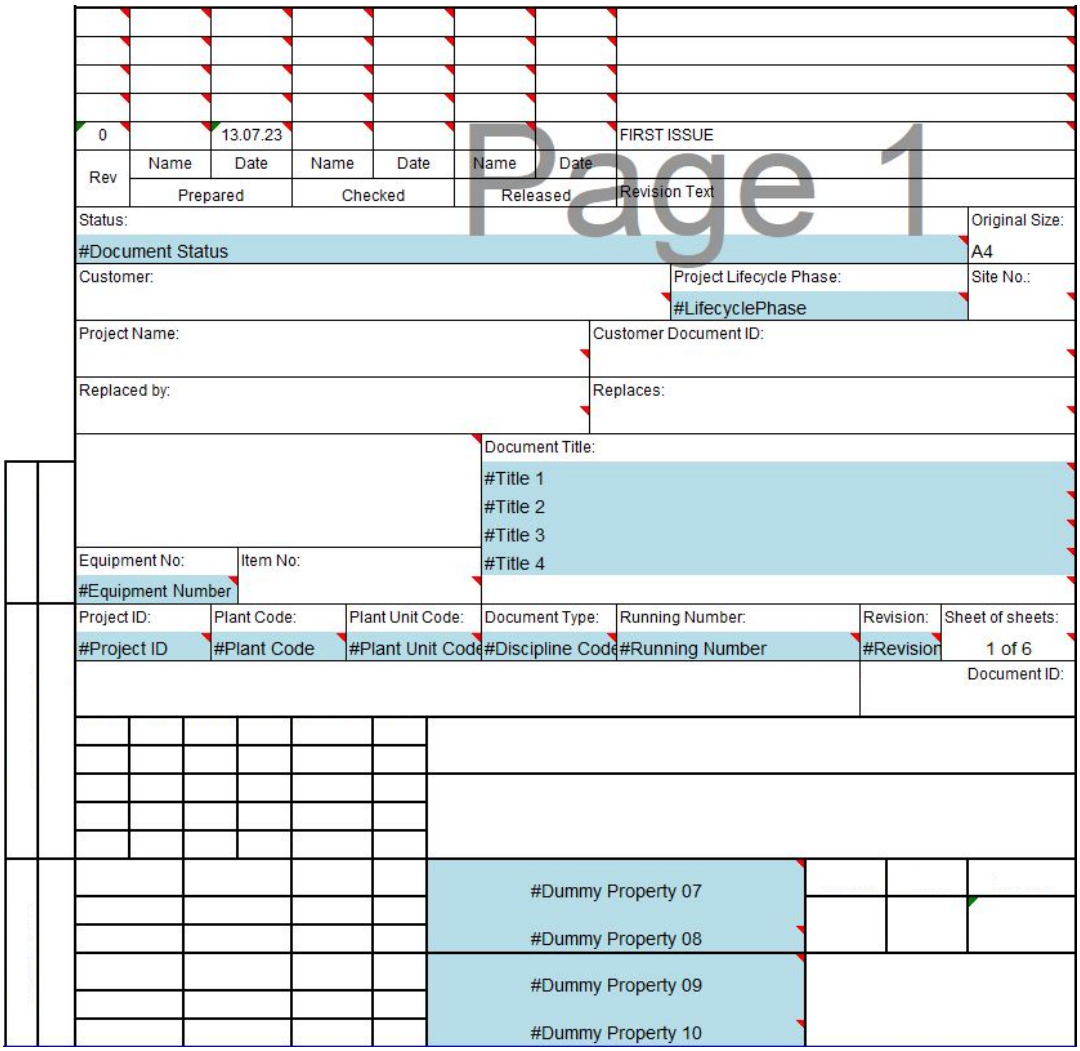


Figure 21: Example of a real document template [52].

It was deemed that the easiest way of checking these sheets was to use this cover sheet in Excel format, as the important data cells are marked with the '#' character. Excel files are used since it was easier to implement in programming, it was deemed that a PDF version of this checker is highly possible but requires another approach, which would take too much time. Using the '#' character as a marker makes it possible to have the checker be dynamic, since it finds those cells and not statically defined cell coordinates.

The workflow for the function is as mentioned shown in Figure 22, and starts out with checking if a directory for the report file exists, and creates one if not. The user is notified if a new directory is created, otherwise the program will proceed without writing anything to output. Firstly, the script scans the target directory for the files listed in the ImportData source file. This is independent from the "Check folder" function. The reason behind this is that the "Check folder" script can be executed very quickly compared to this script, and thus it makes more sense to check if all the files can be read using "Check folder" before executing this script.

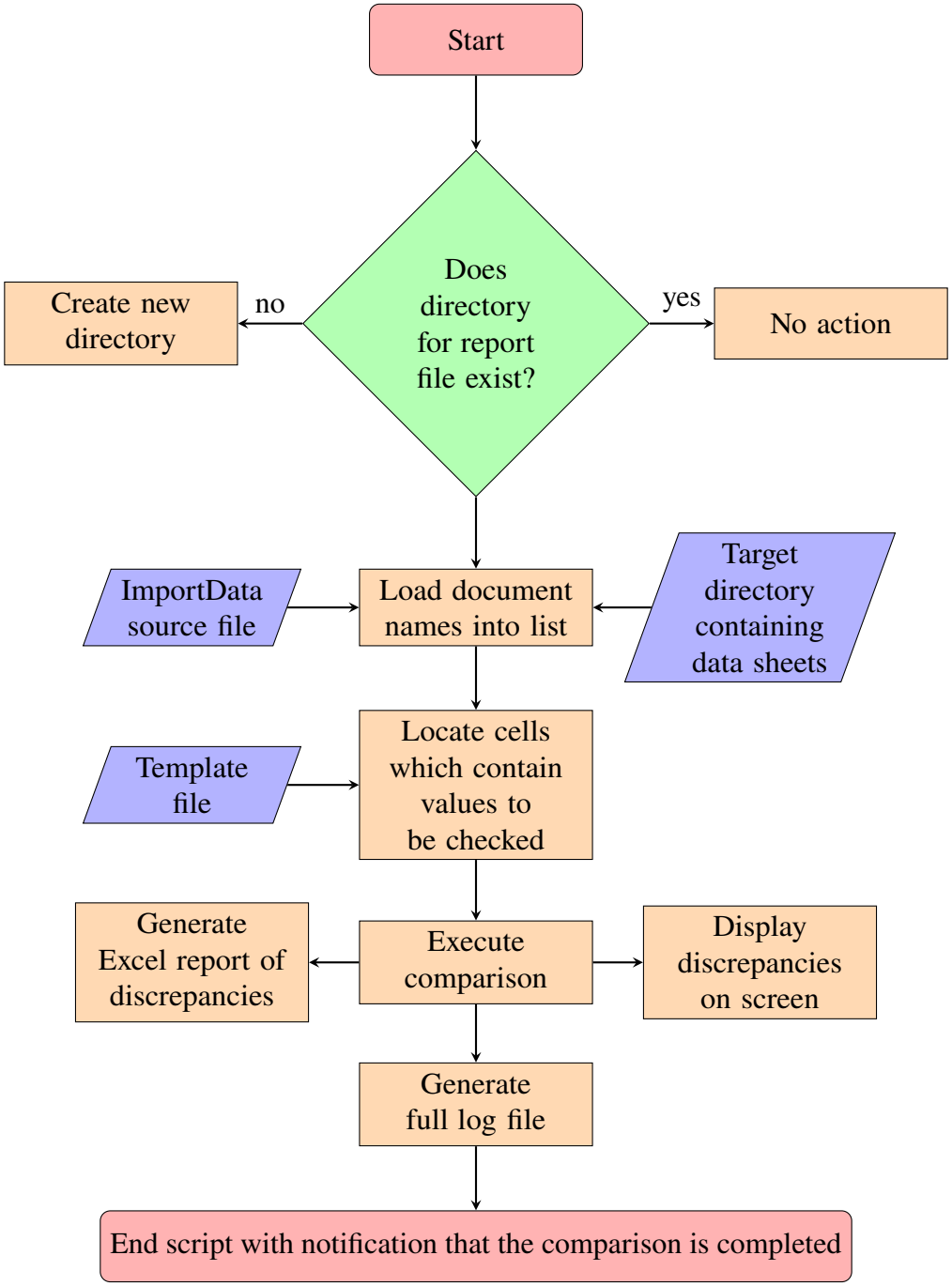


Figure 22: Flowchart for the comparison script.

When the document names are loaded, the template is scanned for the '#' marked cells, and their coordinates are stored in a list. The values after the '#' are also stored, since they are the same as a corresponding column header in the ImportData file that contains the data supposed to go there.

With this information in hand, the coordinate and header lists can be looped simultaneously for each document to find the proper values from the source file, and comparisons to the data sheets can be executed. If discrepancies are found, meaning

that the data sheets do not match up with the ImportData source file, they are written to the output directly in the tool as well as written to a separate Excel report file. The Excel report file provides the user with information about:

- Name of the file that is not consistent with the ImportData sheet.
- Which field that contains the discrepancy.
- Which row in the ImportData sheet that contains the data of said file.

A full log file in text (.txt) format is also written to the same folder as the Excel report file where the full result of every check can be seen.

4.3.3 Visualization

A visual interface was created with the WX module for Python seen in Figure 23. Requirements for the user interface were discussed separately. The discussed requirements comprised:

- Tool tips that clarify what file or path that needs to be selected as inputs.
- Buttons for accessing the file browser through a dialogue box.
- Link to the output report.
- Output window for letting the user see the output and thus know if the scripts ran successfully.
- Link to a User Manual.

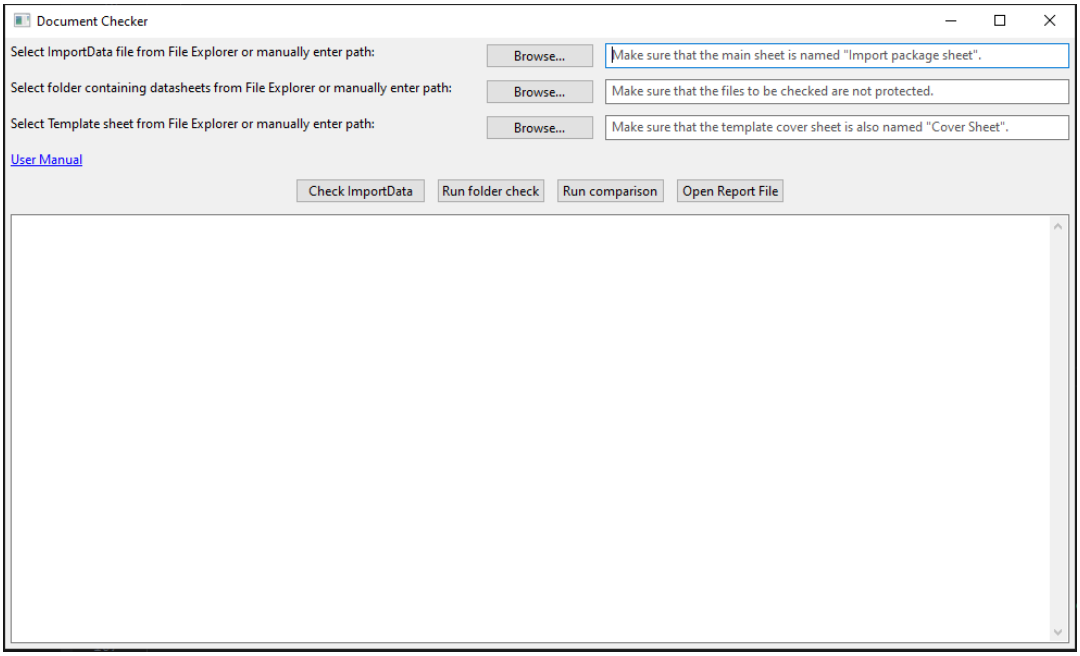


Figure 23: The created interface for the checker tool.

4.3.4 Implementation in AVEVA Engineering

As per request by an end-point user, the tool is implemented as a button in the toolbar of AVEVA Engineering for increased convenience. In practice, this means that the button is a link to an executable file version of the Python script. In AVEVA Engineering, this kind of implementation is done by using its internal Programmable Macro Language (PML). The implementation was realized by converting the tool into an executable file. A separate tab already exists in the software's toolbar for the purpose of containing in-house developed tools. AVEVA offers a "Customize" option, which allows the user to add more buttons to the toolbar in this tab. After a new button has been declared with the name "Datasheet_Checker" in this case, a macro is added for accessing and running the executable file. The macro is as following:

```
syscom "file path"
```

After this is saved, the button for executing this macro appears in the ribbon under a separate tab in AVEVA Engineering.

4.3.5 Improved work process

Without a document checker tool, the current quality assurance procedure is to do sample checks which obviously comes with flaws. In a batch of say, 250 documents, only 5-10 are manually checked by personnel, and if everything seems to be in order it is assumed that the rest are also correct. There is realistically only a very small chance to spot an isolated error in this manner and is thus quite clearly inferior to an automated process where every document is checked. With a a full project sample of 263 files to be checked, a series of test runs were conducted on both a local machine, and a virtual workspace (Citrix Workspace). The local machine used for the runs is a Dell Precision 7530, which has an Intel i7-8850H CPU and 32 gigabytes of Random Access Memory.

Table 4: Run time test results

Run	**Local machine (s)**	**Virtual workspace (s)**
1	262.37	417.93
2	278.67	394.41
3	295.59	435.87
4	266.53	445.01
5	260.31	411.00
6	275.12	413.95
7	244.00	432.87
8	267.34	408.00
9	247.38	415.88
10	263.71	421.04
AVG	**266.10**	**419.6**

From Table 4, it can be concluded that running the script in the virtual workspace is considerably slower than on a local machine. On average, it was found that the run time on the local machine was 266.1 seconds for processing 263 files, and 419.6 seconds on the virtual workspace. Calculating the run times per file:

$$\frac{266.1}{263} = 1.02 \text{ seconds per file}$$

and

$$\frac{419.6}{263} = 1.60 \text{ seconds per file}$$

meaning that on average, the virtual workspace is

$$1.60 - 1.02 = 0.58 \text{ seconds}$$

slower per file.

4.3.6 Comparison to manual checking

If these checks are to be carried out manually, they are rather time consuming. Making sure that all the necessary data exists in the ImportData sheet can be done quite trivially by utilizing MS Excel built-in functions. For example, utilizing the 'COUNTA' function, which returns the number of filled cells in a range, to find out if there are empty columns. Manually making sure that an ImportData sheet contains all the necessary information is still a process that takes several minutes, as compared to a second or two with the developed tool.

Checking the target directory manually is not the easiest task either, since it can be difficult to spot small errors in the file names. The developed tool addresses this issue by performing string comparisons.

In order to do the checks manually, each equipment data sheet must be opened, and the appropriate cells must be found in the ImportData source sheet. This process is tedious and easily takes 1-2 minutes per document to do properly. As mentioned, only a set of random samples can be checked as a result of this being a very time-consuming process. Even if done quickly with each document taking about 1 minute to check, which was deemed as a reasonable estimate, it would still amount to about half a working day to a full day (4-8 hours) to check a full project like the one used for testing (263 files). According to the Finnish Engineers' and Architects' trade union TEK [53], the median salary for Finnish engineers is 5420€/month. Assuming that an average of 100 projects are worked on per year and a 7.5 hour working day, this amounts to an interval of

$$\binom{400}{800} h/y \approx \binom{53}{107} \text{ working days/year}$$

for one engineer going lost if everything is thoroughly checked manually, which is quite a significant number considering that there are about 220 workdays in a calendar

year accounting for holidays. Rounding TEK's stated average salary into salary per day:

$$€5420 * 12\text{months} = €65040/\text{year}$$

$$\frac{€65040}{220 \text{ days}} \approx €296/\text{day}$$

This implies that with time spent being in the mentioned range of 53-107 days, the monetary deficit of manually checking is in the interval of 15700€-31700€ on a yearly basis, not accounting for that the time used on this could be used for working on something else. Thus, the tool allows the user to conveniently check every document massively faster and more comprehensively than manually, saving time and effort that is better used elsewhere, while minimizing errors that could potentially cause major issues further down the road. Such quality errors could potentially be damaging in regard to both company reputation and monetary losses larger than the calculations stated in this section.

4.3.7 Implementation results

The tool was implemented successfully as a shortcut to an executable file in Citrix Workspace. The executable file was saved in said workspace and implemented as explained in Section 4.3.4. The implementation was done successfully, namely, the file can be started through a shortcut in AVEVA Engineering to be used for checking data sheets in projects.

5 Discussion

5.1 Summary

In this study, Digital Twin as a concept has been thoroughly defined. The current state of the art in the field of designing plant DTs was presented. Particularly the means of storing and managing documents related to plant DTs were examined. The study focused on data created with the AVEVA software package, and thus the proposed solutions are tailored to that software. Two separate tools were developed with different approaches to automating data quality checking in the example plant's documentation. One tool was created as a web application that fetches data from AVEVA NET, and the other one was implemented as an executable file accessed with a button macro in the AVEVA Engineering software module. The automated data checks work as intended and enable every project's documentation repository to be thoroughly checked, which has not been feasible to do manually due to the time-consuming and tedious nature of such work. Fully checking the project data means that data quality can be improved, and less errors will make their way to the final product. Implementation of the tools was successful as a whole, constituting proofs-of-concept that can be further developed in the future.

String matching algorithms were tested for scanning asset tags in order to find out if they are named wrongly according to the naming convention in use. Promising, but not conclusive results were found in this matter. Calculations were made in order to illustrate the time and monetary savings enabled by employing automated checks.

5.2 Research questions

In regard to the defined research questions, they have been answered to an extent. Re-iterating the questions:

1. What is the current state of the art in plant Digital Twins?
2. How can project documentation metadata be checked?
3. What would be the most suitable platform for building a tool?
4. Can this significantly relieve the workload of engineers? How much time and effort can be saved?

Addressing the questions in the same order:

1. The current state of Digital Twins in the field of plants has been reported based on recent articles in the field. Digital Twins in general are indicated to have an upwards trajectory in importance and market value. Recent developments in the fields of computing power, machine learning, artificial intelligence and IoT are all greatly contributing to improvements. Challenges such as data quality, data security, gaining trust in industry and overly high expectations in industry were acknowledged. Insights from recent research on retrofitting DT on

brownfield projects as well as implementing new DTs were shown. Advantages of applying DT in industry were recognized. Especially when linking data from different sources, the correctness of the links needs to be verified qualitatively and quantitatively. In DT's, data from different sources can be linked to enhance the transparency and increase the visibility of issues to be addressed in a faster manner.

2. Several ways of checking metadata were examined. The possibilities of utilizing the AVEVA NET portal were examined, as well as algorithms for checking data quality. An AVEVA NET checker tool was implemented as a web application, however it was noted that not all interesting data is accessible through the AVEVA NET portal. Eventually, the decision was taken to design a tool for checking Digital Twin equipment tag metadata coming directly from AVEVA Engineering with the tool being integrated in the software.

3. The Python programming language was selected as the platform for designing a tool. This was due to its many libraries, functionalities, readability, and previous knowledge. Both the possibilities of realizing a tool as a web application and an executable file were explored, since they come with different advantages and disadvantages.

4. The document checker tool evidently greatly reduces time and effort. For each project, it is estimated that a half to one full workday could be saved by implementing such a tool if the documents were currently fully checked. However, this is not the case, meaning that the value added is that the data is being thoroughly checked which it was not before. The developed AVEVA NET tool and the algorithm research cannot be quantified in the same way at this stage.

An overview about the history and current developments in the field of DT and specifically plant DTs was presented to provide context and motivation for the work carried out. It was found that DT as a concept has been on the rise for the past decades and is only set to grow, being accelerated by recent developments in ML, AI and computing power. With this background, it makes sense for modern companies to expand their capabilities in the field of digitization in order to keep up with competition. The possibilities for leveraging the advantages of DT to increase efficiency and sustainability of a plant should definitely also be mentioned as a key motivation for this.

This study project resulted in a couple of tools that help engineers during the creation of a plant DT. The results were a document checker tool directly built into AVEVA Engineering and a web application AVEVA NET checker tool.

5.3 Results compared to Requirements

In this subsection, the results presented in Chapter 4 are compared to the requirements stated in Section 3.5.

5.3.1 AVEVA Engineering document checker

During the development process, a new requirement was added. The updated requirement list is seen in Table 5:

Table 5: Updated requirements list for the document checker.

Requirement	Importance	Reasoning
Check that the master data matches with equipment data sheets	Demand	This is the main purpose of the tool
Dynamic scanning to handle different templates	Demand	The tool is more useful in this case
Tool should alert the user if the master data sheet is incomplete	Demand	Useful information
A report should be generated for the user	Demand	Vital feature for analysing results
Tool should alert the user about files in a folder that cannot be found in the master data sheet	Wish	Useful information
Tool should contain a link to a user manual	Wish	Useful functionality

We can state that the designed tool fulfils all these requirements in a manner that was deemed to be satisfying. A wish that occurred later in the development process was that a user manual should be created, along with a link to the manual included in the tool. This additional wish was also implemented without major issues, and the manual itself received positive feedback from the in-house AVEVA team.

5.3.2 AVEVA NET checker tool

Regarding the AVEVA NET checker, the requirements stayed the same for the duration of the process according to Table 3. All the requirements except the last one were met in some regard, and this was because the project deviated away from the possibility of accessing that kind of data. In the first stages, the idea was that a DT built in APS should be examined and checked. However, this was a project that was being undertaken parallel to this thesis and unfortunately encountered obstacles, which made this project proceed in another direction.

The generated report states if tagged items are linked and to which document, but it does not state if this is correct or not. This is because it was difficult to judge if a tagged item is "correctly" linked to a respective document. Further research is necessary for this specific matter.

5.4 Proposed further research

Regarding the research carried out for this study, there are some issues yet to be investigated. The document checker tool could most likely be further optimized to run faster, and it could be researched whether it would be faster to check PDF renders instead of Excel sheets. It could also be investigated if there are computationally faster ways of handling the sheets within Python. Alternatively, the feasibility of translating the code into a programming language more suited for applications, such as C# could be examined.

The AVEVA NET checker could include more functionality, but it was decided that the document checker was to be prioritized, since it has more practical value at this point. An improved iteration in the future should also judge if documents are linked correctly.

The next steps for this project would be to realize the APS DT that has been under development for the duration of this thesis project. As the APS DT is built with Nodejs, it is highly possible to create a tool tailored to checking e.g., sensor readings and CAD model properties in it.

Machine learning tools could be explored more in order to find tags that are incorrect or incorrectly linked. In this study, some trials were performed in this direction, however, additional time is needed to investigate the matter on both a literature review and experimental basis. The results in this study imply that it could be an approach to solving the problem.

References

[1] C. González-Lluch, P. Company, M. Contero, J. D. Camba, and R. Plumed, "A survey on 3D CAD model quality assurance and testing tools," *Computer-Aided Design*, vol. 83, pp. 64–79, Feb. 2017.

[2] E. Ferko, A. Bucaioni, and M. Behnam, "Architecting Digital Twins," *IEEE Access*, vol. 10, pp. 50335–50350, 2022.

[3] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," 03 2015.

[4] R. Piascik, J. Vickers, D. Lowry, S. Scotti, J. Stewart, and A. Calomino, "Technology area 12: Materials, structures, mechanical systems, and manufacturing road map," *NASA Office of Chief Technologist*, pp. 15–88, 2010.

[5] M. Grieves, "Origins of the Digital Twin Concept," 08 2016.

[6] V. Strobel, "Pold87/academic-keyword-occurrence: First release," Apr. 2018.

[7] trends.google.com, "Digital Twin," 2023.

[8] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," *Transdisciplinary perspectives on complex systems: New findings and approaches*, pp. 85–113, 2017.

[9] NASA, "Draft modeling, simulation, information technology & processing roadmap—technology area 11," 2010.

[10] R. Stark and T. Damerau, *Digital Twin*, pp. 1–8. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019.

[11] L. Wright and S. Davidson, "How to tell the difference between a model and a digital twin," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7, p. 13, Dec. 2020.

[12] A. M. Madni, C. C. Madni, and S. D. Lucero, "Leveraging digital twin technology in model-based systems engineering," *Systems*, vol. 7, no. 1, p. 7, 2019.

[13] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 2405–2415, Apr. 2019.

[14] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital Twin: Enabling Technologies, Challenges and Open Research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020.

[15] Wikipedia, "Gartner hype cycle — Wikipedia, the free encyclopedia." http://en.wikipedia.org/w/index.php?title=Gartner%20hype%20cycle&oldid=1156200393, 2023. [Online; accessed 25-May-2023].

[16] X. Ochoa and E. Duval, "Automatic evaluation of metadata quality in digital repositories," *International journal on digital libraries*, vol. 10, pp. 67–91, 2009.

[17] G. Di Geronimo Gil, D. Alabi, O. Iyun, and N. Thornhill, "Merging process models and plant topology," in *2011 International Symposium on Advanced Control of Industrial Processes (ADCONIP)*, pp. 15–21, 2011.

[18] Autodesk, Inc., "Autodesk Platform Services implementation for internal use."

[19] P. E. Bézier, "Example of an existing system in the motor industry: the unisurf system," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 321, no. 1545, pp. 207–218, 1971.

[20] D. T. Ross, *Computer-aided design: a statement of objectives*. MIT Electronic Systems Laboratory, 1960.

[21] C. M. Eastman, C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[22] P. R. Smith and J. V. Thomas, *Piping and pipe support systems*. McGraw Hill Book Co., New York, NY, 1987.

[23] B. C. Kim, Y. Jeon, S. Park, H. Teijgeler, D. Leal, and D. Mun, "Toward standardized exchange of plant 3D CAD models using ISO 15926," *Computer-Aided Design*, vol. 83, pp. 80–95, 2017.

[24] M. J. Pratt *et al.*, "Introduction to ISO 10303—the STEP standard for product data exchange," *Journal of Computing and Information Science in Engineering*, vol. 1, no. 1, pp. 102–103, 2001.

[25] J. W. Klüwer, M. G. Skjæveland, and M. Valen-Sendstad, "ISO 15926 templates and the Semantic Web," in *Position paper for W3C Workshop on Semantic Web in Energy Industries; Part I: Oil and Gas*, 2008.

[26] B. Kim, H. Teijgeler, D. Mun, D. Sun, J. Hwang, and S. Han, "A Representation and Implementation of Process Plant Models using OWL and ISO 15926," in *PLM08 Conference*, 2008.

[27] T. Holm, L. Christiansen, M. Göring, T. Jäger, and A. Fay, "ISO 15926 vs. IEC 62424 — Comparison of plant structure modeling concepts," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pp. 1–8, 2012.

[28] M. Schleipen, R. Drath, and O. Sauer, "The system-independent data exchange format CAEX for supporting an automatic configuration of a production monitoring and control system," in *2008 IEEE International Symposium on Industrial Electronics*, pp. 1786–1791, 2008.

[29] A. K. Jana, *Chemical process modelling and computer simulation*. PHI Learning Pvt. Ltd., 2018.

[30] M. Dosta, J. D. Litster, and S. Heinrich, "Flowsheet simulation of solids processes: Current status and future trends," *Advanced Powder Technology*, vol. 31, no. 3, pp. 947–953, 2020.

[31] "SIM - Process Simulation Module)." https://www.metso.com/portfolio/hsc-chemistry/process-simulation-module/. Accessed: 2023-06-14.

[32] G. J. Gilman and J. Gilman, *Boiler control systems engineering*. Isa, 2010.

[33] "What Is SQL (Structured Query Language)?." https://aws.amazon.com/what-is/sql/. Accessed: 2023-05-16.

[34] S. Sierla, M. Azangoo, K. Rainio, N. Papakonstantinou, A. Fay, P. Honkamaa, and V. Vyatkin, "Roadmap to semi-automatic generation of digital twins for brownfield process plants," *Journal of Industrial Information Integration*, vol. 27, p. 100282, 2022.

[35] M. Hoernicke, A. Fay, and M. Barth, "Virtual plants for brown-field projects," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pp. 1–8, 2015.

[36] AVEVA, "Getting Started with AVEVA E3D." https://help.aveva.com/AVEVA_Everything3D/2.1.0.3/wwhelp/wwhimpl/js/html/wwhelp.htm.

[37] "AVEVA Engineering." https://www.aveva.com/en/products/aveva-engineering/. Accessed: 25.04.2023.

[38] BlueCielo, "BlueCielo Meridian Enterprise 2018 User's Guide: Creating an import data source file," Jul 2018.

[39] " AVEVA NET Workhub and Dashboard)." https://www.aveva.com/content/dam/aveva/documents/brochures/AVEVA-NET-Workhub-Dashboard.pdf.coredownload.inline.pdf. Accessed: 2023-05-19.

[40] G. Pahl, W. Beitz, J. Feldhusen, and K. Grote, "Engineering design: A systematic approach third edition," *Berlin, Springer Science+ Business Media Deutschland GmbH, 2007. 632*, 2007.

[41] "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 610.12-1990*, pp. 1–84, 1990.

[42] G. M. Bonnema, K. T. Veenvliet, and J. F. Broenink, *Systems design and engineering: facilitating multidisciplinary development projects*. CRC press, 2016.

[43] "ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, 2017.

[44] E. R. Harold and W. S. Means, *XML in a nutshell: a desktop quick reference*. " O'Reilly Media, Inc.", 2004.

[45] K. Thompson, "Programming techniques: Regular expression search algorithm," *Communications of the ACM*, vol. 11, no. 6, pp. 419–422, 1968.

[46] M. Erwig and R. Gopinath, "Explanations for regular expressions," in *International Conference on Fundamental Approaches to Software Engineering*, pp. 394–408, Springer, 2012.

[47] M. Grootendorst, "Polyfuzz: Fuzzy string matching, grouping, and evaluation.," 2020.

[48] C. Zhao and S. Sahni, "String correction using the Damerau-Levenshtein distance," *BMC bioinformatics*, vol. 20, no. 11, pp. 1–28, 2019.

[49] Aspose, "Python 3D File Manipulation APIs."

[50] "Download Python | Python.org)." https://www.python.org/downloads/. Accessed: 2023-07-05.

[51] B. Vollebregt, "GitHub: auto-py-to-exe," 2023.

[52] Metso Oyj, "Damper template," 2023. Unpublished internal company document.

[53] TEK, "Työmarkkinatutkimus," 2022.