Miika Nurmi

Data Management Solution for Customer-Specific Configurations

School of Technology and Innovations
Master's Thesis in Electrical Engineering
Energy and Information Technology, M.Sc. (Tech.)

Vaasa 2026

UNIVERSITY OF VAASA
School of Technology and Innovations
Author: Miika Nurmi
Title of the thesis: Data Management Solution for Customer-Specific Configurations
Degree: Master of Science in Technology
Degree Programme: Electrical Engineering
Supervisor: Kimmo Kauhaniemi
Instructor: Mikael Snickars
Evaluator: Hannu Laaksonen
Year: 2026 **Pages:** 78

ABSTRACT:
The aim of the thesis was to develop a more unified and traceable approach for managing customer-specific configurations in the case company. The need for the study was based on the current configuration management environment, where configuration files are managed in several locations. This creates challenges in identifying the correct version, managing changes, finding the correct information and maintaining consistent practices. The thesis examined the current state of configuration management, identified development needs and defined requirements for an improved solution. The work was based on literature related to configuration data management, a current state analysis, stakeholder interviews and a review of potential solution alternatives. The results showed that the current way of working supports daily work reasonably well, but configuration management is partly dependent on project-specific practices and personal knowledge. The most important development needs were related to version control, change management and standardizing configuration management processes. Based on these findings, user, process and technical requirements were defined for an improved configuration data management solution. SharePoint, M-Files and DocuWare were reviewed as potential solution alternatives. SharePoint and M-Files were compared in more detail because they were considered the most relevant alternatives for the case company's environment. M-Files was selected as the basis for the solution concept and pilot implementation because its functionalities supported the identified requirements. The thesis concludes that improving configuration management requires a suitable solution and common operating practices. The proposed concept provides a basis for further development towards a unified, traceable and less person-dependent configuration data management.

KEYWORDS: system requirements, information management system, system development, solution concept, data management

UNIVERSITY OF VAASA
School of Technology and Innovations
Tekijä: Miika Nurmi
Tutkielman nimi: Toimintamalli asiakaskohtaisten konfiguraatioiden tiedonhallintaan
Tutkinto: Diplomi-insinööri
Koulutusohjelma: Sähkö- ja energiatekniikan maisteriohjelma
Opintosuunta: Sähkötekniikka
Valvoja: Kimmo Kauhaniemi
Ohjaaja: Mikael Snickars
Tarkastaja: Hannu Laaksonen
Valmistumisvuosi: 2026 **Sivumäärä:** 78

TIIVISTELMÄ:
Tämän diplomityön tavoitteena oli kehittää yhtenäisempi ja jäljitettävämpi toimintamalli asiakaskohtaisten konfiguraatioiden hallintaan kohdeyrityksessä. Työn tarve perustui nykyiseen konfiguraatioiden hallintaympäristöön, jossa konfiguraatiotiedostoja hallitaan useissa eri sijainneissa. Tämä aiheuttaa haasteita oikean version tunnistamisessa, muutosten hallinnassa, oikean tiedon löytämisessä ja yhtenäisten käytäntöjen ylläpitämisessä. Työssä tarkasteltiin konfiguraatioiden hallinnan nykytilaa, tunnistettiin kehitystarpeita ja määriteltiin vaatimukset parannetulle ratkaisulle. Työ perustui konfiguraatiotiedon hallintaa käsittelevään kirjallisuuteen, nykytila-analyysiin, sidosryhmähaastatteluihin sekä mahdollisten ratkaisuvaihtoehtojen tarkasteluun. Tulosten perusteella nykyinen toimintatapa tukee päivittäistä työtä kohtuullisen hyvin, mutta konfiguraatioiden hallinta on osittain riippuvainen projektikohtaisista käytännöistä ja henkilösidonnaisesta tiedosta. Keskeisimmät kehitystarpeet liittyivät versionhallintaan, muutosten hallintaan ja konfiguraatioiden hallintaprosessin standardointiin. Näiden havaintojen perusteella määriteltiin käyttäjä-, prosessi- ja tekniset vaatimukset parannetulle konfiguraatiotiedon hallintaratkaisulle. Mahdollisina ratkaisuvaihtoehtoina tarkasteltiin SharePointia, M-Filesia ja DocuWarea. SharePointia ja M-Filesia vertailtiin tarkemmin, koska niitä pidettiin kohdeyrityksen toimintaympäristön kannalta merkittävimpinä vaihtoehtoina. Ratkaisukonseptin ja pilottitoteutuksen pohjaksi valittiin M-Files, koska sen toiminnallisuudet tukivat tunnistettuja vaatimuksia. Työn johtopäätöksenä on, että konfiguraatioiden hallinnan parantaminen edellyttää soveltuvaa ratkaisua ja yhteisiä toimintakäytäntöjä. Ehdotettu konsepti tarjoaa pohjan jatkokehitykselle kohti yhtenäisempää, jäljitettävämpää ja vähemmän henkilösidonnaista konfiguraatiotiedon hallintaa.

AVAINSANAT: järjestelmävaatimukset, tiedonhallintajärjestelmä, järjestelmäkehitys, ratkaisukonsepti, datanhallinta

Table of contents

Table of tables

Table of figures

1 Introduction

1.1 Background

Companies generate increasing amounts of data as digitalization advances and different technologies and solutions become more integrated. The large amount of data, its diversity, and segmentation have become key challenges for industrial data management. Data-oriented knowledge is a significant part of the company's operational data, and its management can be diversified and dependent on different systems, processes, and individual data user preferences (Zhang & Yang, 2025). Information fragmentation is a dire problem regarding the usage of data as there are several different possibilities for the fragmentation. Data from the same process or activities may scatter across different locations, solutions, models and documents, which can make it more difficult to form a good understanding of the data and accessibility to the data when it is needed (Van Der Aa et al., 2015).

The amount of data is increasing which adds the need of systemic data version control. Version control of the data is crucial to ensure information is transparent, trustworthy and coherent throughout the entire life cycle which will ease to notify the changes on the documents or if there is a problem with the data. When the data is edited frequently and there are multiple users for the data the version control is crucial (Pulicharla, 2024).

1.2 Purpose of the study and research questions

This thesis was conducted for a commissioning company, hereafter referred to as the case company. In this thesis, a configuration refers to a set of data that defines system behaviour in a customer-specific application. The term customer-specific configuration is used to refer to configurations created for specific customer projects. The purpose of

the study is to analyse how configuration data and related documentation can be managed in a more unified, traceable, and systematic way through the data lifecycle. The current solution for this matter is outdated and not highly scalable. Currently, the configuration data and related documents are stored across several different systems, making it difficult to maintain consistent version control, ensure the data is accessible to all stakeholders and support collaboration between all the data users. The study aims to analyse current solutions, practices and processes, identify needs for the development and explore potential alternative solutions that could support a more coherent approach to data management.

The study seeks to define the requirements for a unified solution for all the data by analysing existing configuration management practices, conducting data analysis and gathering insights from interviews with the stakeholders and relevant project teams. This study will provide essential information about the needs for the next proposal for the solution.

To fulfil the purpose of the study, the following research questions are addressed:

1. What methods, systems and practices are currently used for managing customer-specific configurations?
2. How well do the existing systems support the needs of different stakeholders throughout the data lifecycle?
3. Is it feasible to manage configurations through a unified system?
4. What internal and external solution options are available for improving the configuration data management?

1.3 Scope and limitations

This thesis focuses on data management for customer-specific configurations within the case company. The management of the configurations as an object of study is specifically focused on configuration storage, version control and sharing of the configurations. The

study is limited to reviewing data that is already available on the customer-specific configurations portfolio. The market review of different kind of data management solutions will be delimited based on literature and preliminary investigation of the solutions used and which would be suitable for this use case. The study is also limited in extent, not to create a fully ready data management solution, but to create a set of system requirements based on the interviews and data analysis and create a solution concept and pilot a suitable solution. The study focuses specific department of the case company and their processes regarding the customer-specific configuration management.

2 Configuration Data Management

2.1 Principles of configuration data management

Configuration management can be understood as a systematic approach aimed at maintaining the integrity of a system and its associated information during changes. It is not limited to preservation of documentation or information but is essentially related to the management of changes in relation to defined baselines or approval baselines. In this case, the task of configuration management is to ensure that the information describing the system remains controlled, consistent and usable throughout the project and subsequent life cycle (Whyte et al., 2016).

The need for systematic management is particularly important when there is a lot of configuration related information, the information has a variable structure and is used by several actors at different stages of the life cycle. In complex environments, information does not consist of individual files or documents, but of larger interconnected data sets that are produced, updated and utilized at different stages. In such cases, configuration management also requires common processes and management practices that can be used to coordinate changes, dependencies and information structure between different actors. Without such practice, information and changes to it can become dispersed in a way that weakens the manageability and subsequent use of the information (Müller, 2013; Whyte et al., 2016).

Key principles for configuration management include data consistency, change traceability, and the ability to utilize data throughout the system's life cycle. In long-lived complex systems, data must remain understandable and usable even during later phases even when the data is being under maintenance and changes. This emphasizes the need to manage configuration related data in a way that supports change visibility, data traceability and subsequent reuse (Müller, 2013).

2.2 Characteristics of configuration data

Configuration data is not just a single file or a discrete data element, but a larger set of related information, documents and relationships. In practice, this set can include documents, drawings, models, plans, requirements and other information describing the system. Therefore, configuration data can be viewed as structured and relational information, the meaning of which is not only based only on a single document, but also on how the information relates to other parts of the set (Whyte et al., 2016).

A key feature of configuration information is also its temporal nature. The information does not remain unchanged but is updated during the life cycle of the system and the project, in which case changes can affect both the technical object and the information describing it. The visibility of changes, traceability, management of approved states and connection to previous versions and states are critical in configuration management (Müller, 2013; Whyte et al., 2016).

Configuration information is strongly linked to different phases of the life cycle. The same information may be necessary in planning, implementation, deployment, maintenance, modification work and decommissioning. A long-life cycle increases the risk that information will become more difficult to trace, maintain or interpret unless it is managed systematically. Simply storing of the information is not enough, but information must be kept consistent, identifiable and usable at different phases and from different user perspectives (Müller, 2013; Whyte et al., 2016).

2.3 Key practices in configuration data management

One of the key practices in configuration management is the identification of managed objects. In practice, this means that the objects included in the configuration and the

information describing them are defined in such a way that they can be managed consistently in different phases. Identification applies to both the configurations themselves and the documentation describing them, because without the identification, later change management, version separation and status tracking are not feasible. Without sufficient identification, subsequent change management, traceability and state management can become difficult, especially when the same object is related to multiple views, data structures or life cycle stages (Müller, 2013; Whyte et al., 2016).

Another key practice can be seen as a version and revision control. In configuration management, it is not only about that information exists, but also how different versions, revisions and variants are distinguished from each other and how they are maintained as coherent entities. This requires practices and processes that allow previous states to be preserved or reconstructed, new versions to be created in a controlled manner and different variations to be handled so that inconsistent combinations do not end up in use. Version is therefore not just about storing files, but also about the rules and structures that govern the relationship between versions (Conradi & Westfechtel, 1998).

Another key practice is change control. Configuration management is based on the principle that changes are not made uncontrolled but are related to an approved baseline and handled through defined processes. This is evident, for example, in the change proposal, approval and release practices, and in the fact that the effects of changes are assessed before their approval. From a management perspective, it is not only important to add new information, but also that the basis, effect and approval of the change can be included as part of the controlled whole (Müller, 2013; Whyte et al., 2016).

Figure 1 illustrates the general logic of change control in configuration management. A change is not introduced directly into the controlled configuration, but is first proposed, assessed and approved before the revised configuration is released.

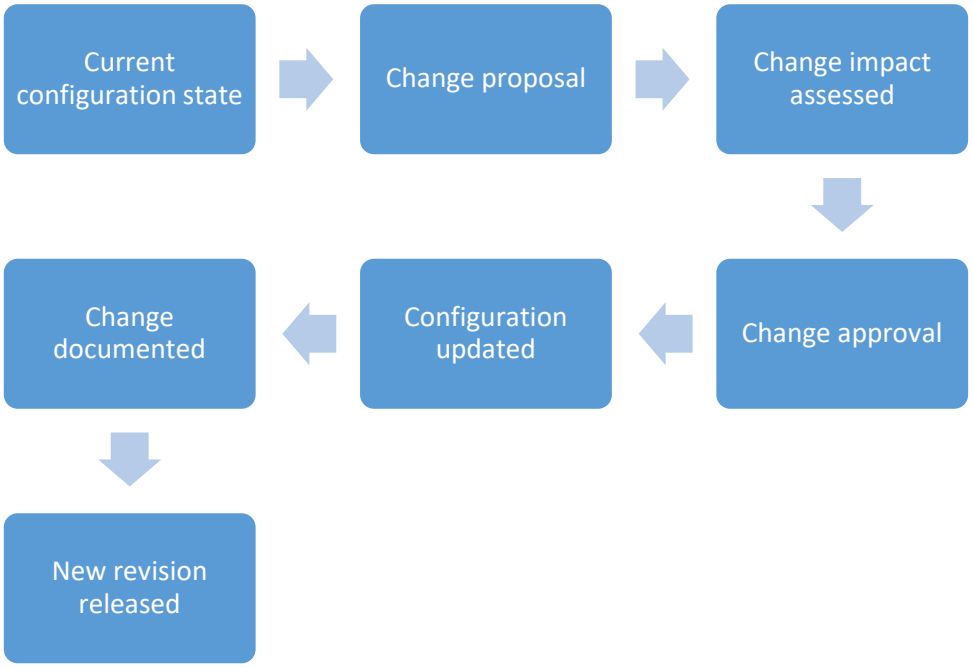

Figure 1. Simplified change control process in configuration management, adapted from Whyte et al. (2016).

2.4 Information management systems for configuration data

Configuration information management does not necessarily take place in a separate system intended for configuration management, but configuration related information can be part of a broader document and information management environment. In such environments, configuration files, technical documents and other project-related material can be located in the same entity, in which case configuration information management is linked to more general document management and information retrieval practices (Whyte et al., 2016; Zantout & Marir, 1999).

The key functions of document management systems are document storing, indexing, search, sharing, collaboration and supporting document related workflows. From the perspective of how easy it is to find the data, the value of a system is not only based on storing of documents, but also on how documents can be classified, described, searched and presented to the user. Especially in specialized technical environments, effective

document management often requires solutions that supports search and structure such as classification, tagging, metadata or other ways of organizing information (Kim & Compton, 2004; Liu et al., 2008; Zantout & Marir, 1999).

The suitability of a document management system cannot be assessed solely on the basis of whether it can store files. What is essential is how well the system supports identification, retrieval, version control, access control and connections to other documents and data sets. When configuration information is part of a broader document environment, the same system can support both general document management and configuration management needs, but in this case the requirements for configuration information may be stricter than for regular project documentation (Liu et al., 2008; Whyte et al., 2016; Zantout & Marir, 1999).

2.5 Foundations for Solution concept development

Developing a solution concept requires an understanding of the work environment in which the system will be used and the information needs of the users. A knowledge management system cannot be evaluated solely on the basis of technical features, but must support how users search, store, update and utilize information in their work. Therefore, the starting point of the solution concept must be an understanding of the user's work tasks, information usage patterns and operating environment (Lowe et al., 2004).

Creating a solution concept also requires a structured way to translate the identified problems and user needs into requirements for the system. If the solution is only viewed through individual features, the overall picture can remain fragmented, and comparison of different options becomes difficult. Therefore, a requirements based approach is needed, in which user needs, system requirements and the features of potential solutions can be systematically linked (Coronado M et al., 2002).

In a complex system environment, developing a solution concept is not a single selection decision, but step-by-step process in which needs are collected, requirements are specified and the starting points of the solution are formed in a commonly understandable form. In such process, the choice of technology is only one part of the whole. The solution concept is formed gradually as the problem, goals, user needs and acceptable requirements are specified in relation to each other (Ratchev et al., 2003).

It is essential to maintain a connection between the needs of the users and the system solution to be implemented. If this connection is not established, the system may not respond to the real work situations and practical needs of the users. In this context, the definition of requirements acts as a bridge between the needs of the users and system solution, as it can be used to ensure that the solution addresses the right problem and that its suitability can be assessed later (Saiedian & Dale, 2000).

3 Current State Analysis

3.1 Interview material and current state assessment

The current state analysis is based on interviews with stakeholders and review of the current solution of the configuration data management. The aim of the interviews was to form a comprehensive picture of how configurations are currently created, stored, shared and managed across different roles and work phases. The interviews also aimed to identify challenges related to the current way of working and development needs for the future proposal.

The interviews were conducted as semi-structured thematic interviews. A pre-prepared question framework was used in the interviews, the themes of which were related to, among other things, current systems and tools, storage and sharing of configurations, version controlling, change tracking, challenges encountered in practice and desired features of future solution. The semi-structured interview format was chosen because it allowed the discussion to be focused on themes central to the research while at the same time deepening the answers and additional questions.

The interviewees were selected based on their roles and experience so that the data would cover as many perspectives as possible related to current solution to configurations. The interviewees worked in different roles, and their experiences within the company varied. The interviewees were anonymized in the study, and they are referred to by identifiers such as A,B,C. The interviewees are summarized in Table 1.

The interviews were conducted with the consent of the interviewees and recorded for later review. The interviewees were informed before the interview began that the recordings would only be used for the analysis of this thesis. They were informed about the confidential treatment of the material, anonymization and that the audio recording would be deleted after the work has completed.

Table 1. Interviewees.

Identification letter	Role	Years in the company
A	Project manager	4
B	Project manager	1,5
C	Project manager	3
D	Project engineer	5
E	Senior project engineer	20
F	Project engineer	3
G	Project engineer	4
H	Application engineer	20
I	Test specialist	9,5

3.2 Existing systems and tools

Based on the current state analysis, configuration management as where the data is located and managed is part of broader project documentation and data environment. Configurations appear to be one file type among other project files, and their management is linked to the same environments and structures as the management of other project related materials. This observation was also supported by the interviewees, in which SharePoint was repeatedly described as the main shared environment for project documentation and also for configuration files and related documentation. Several interviewees described that project documentation and documentation distributed to the customer are stored in SharePoint, although individual file processing can also take place elsewhere. Configuration file takes mostly place on SharePoint after they are made by the engineer.

Based on the current state the role of SharePoint seems to be particularly related to shared storage, project-specific folder structure and data sharing. At the same time, it is clear that it does not cover the entire work chain, but files are also processed in other environments during the work phase. The interviews clarified this picture in particular by stating that unfinished files produced by tools are stored locally on the author's own computer or shared storage within the case company folder during the design and development of the configurations. According to the interviewees, a version that is intended to be stored in a shared environment or delivered to the customer is typically transferred to SharePoint at a later stage, when the configuration is finished.

The current system environment also does not seem to consist of just one tool, but several parallel tools that have different tasks during the work process. Based on observation, this is reflected in the fact that the shared document environment, personal work files and materials delivered to the customer are not fully managed in the same place or at the same stage. The interviewees supported this observation. In addition to SharePoint, some interviewees mentioned Teams, customer portals, old network drives. The current environment appears to be an environment consisting of several data management locations, where one tool does not cover the entire data lifecycle. Interviewees B,C,D and F referred to additional environments such as Teams, customer portals, network drives or other project-specific storage locations.

It has been noted that folder structures, file names and other practical arrangements play a major role in a data management. Since the configurations are not in a configuration management system, the findability and usability of data seem to depend partly on how the project-specific structure has been implemented. The interviewees clarified this picture by showing that practices are not completely consistent. Some interviewees described the folder structure and storage method as already partially standardized, while others pointed out that practices vary more in older projects. In addition, the interviewees mentioned templates and the use of previous projects as part of the work, more informal practices also supporting the handling of configurations.

Interviewees B,F,H and I raised folder structures, templates and project-specific conventions as important part of how configuration related information is currently managed.

Overall, the current system appears to be an environment where SharePoint serves as the main common management location for configuration files and project documentation. But the actual work phase for configuration management also takes partly in local files and other parallel environments. This overall picture was supported across multiple interviews, particularly by interviewees A,B,C,D,F,G and I.

3.3 Structure, quality and usability of configuration data

This topic is dependent on the whole environment where the configuration data is being stored, as the configuration data is a part of larger environment where the data structure consists of folders, file names and project specific solutions. This means that the quality and usability of the data do not only depend on the content of the file itself, but also on how the data is named, where it is placed and how well the configurations connection to its related material is understood. This observation was presented by several interviewees. Interviewees B, E, G and H described that the folder structure, file names and project specific practices have a great impact on how easily the configurations can be found and used later.

From a structural perspective, both standardization and variation can be seen in the current state. Some interviewees described that a more unified folder structure has been created in SharePoint, to make it easier to find information and to make the solution between projects more consistent and unified. At the same time, the interviewees revealed that practices vary more in older projects and that not all projects follow exactly the same structure. Interviewees B, G and H highlighted the standardized folder structure as positive development, but interviewees E and F described at the same time

that the structure does not always remain controlled or that project specific solutions may still differ from each other.

In terms of data quality, the interviewees particularly highlighted the importance of version control, visibility of changes and naming conventions. In most of the cases, versioning is based on the date, file name, version number, or that old versions are moved to a separate archive folder. This produces at least a basic level of order, but at the same time the content of the changes and the differences between versions are not always clearly visible directly from the file or its location. Interviewees B, E, G and H described different naming and versioning conventions, where the latest version is identified based on the file name, date and the location in the structure. At the same time interviewees A and D pointed out that systematic version control is not implemented to the same extent in all projects. The experiences of interviewees are not completely in uniform regarding the practices because some interviewees felt that the current practices are sufficient for the current need even as the practices are not common for all projects.

From the usability perspective the key question is how easily the right information can be found and how well its meaning is understood in a later use situation. The usability of the data seems to be dependent strongly on whether the user is already familiar with the project structure or not. The interviews revealed that the information is usually found better when the person has been involved in the project themselves or is familiar with the structure used, but at a later stage or when transferred to another person, finding the right information can be clearly more difficult. Interviewees B, E and I highlighted situations where finding a file, identifying the right document or locating templates was not completely straightforward. Interviewee D described that the actual search function is not really used, but information is found by clicking on the folder structure. This suggests that usability is based more on knowing the structure than on the search functions offered by SharePoint.

The usability of the data is also affected by the connection of configurations to other data, such as software versions, templates, drawings, signal lists or project specific background documents. If these connections to the configurations are not visible or if they are only based on the knowledge of experienced person where to look, the use of the data becomes difficult, especially when the project is old or the responsibility is transferred to another person. Interviewees C, G and H highlighted the importance of software versions, templates and other background data for correctly understanding the content and usage environment for configurations.

Overall, the current structure, data quality and usability appear to support basic operations reasonably well, but are still partly based on varying practices, personal knowledge and project specific logic. Using the configurations is easier when the author knows the history and structure of the project, but more challenging when the configurations are searched later by another person or among several projects. This makes the structure, naming, version control, visibility and connections to other documents essential for the configurations.

3.4 Current processes and practices

Configuration management does not appear to have a strictly defined process that would be followed in the same way in all projects. However, the interviewees revealed that there are various established working methods used to handle the configurations during projects. Some interviewees described that some projects use a clear practices which are aligned within the project, while others described that the practices are more author or project specific. Interviewee D stated that at least in bigger and newer projects there has been some kind of process around version control, but in most projects, there were no actual uniform process according to interviewee D. Interviewee I described that when the group of authors is small enough it has been possible to operate without strict processes around this topic.

A typical working method is that configurations are first created in a separate application, after that they are transferred to SharePoint when the version is ready to be passed on. This came up in several interviews. Interviewee D described that incomplete backups are normally kept locally and that the latest configuration is stored in SharePoint in the highest level of the folder tree.

Version control is largely based on the file name as it can present project name and version number. The most common usage of version control in configuration files is date on the file name. The content of the changes done in the configuration files are not always visible between versions. This varies a lot between authors and projects. Some leave change log on the description of the configuration file, some create a readme within the same folder of the file. Most interviewees agreed that there are usually an archive or old named folder in the SharePoint which holds on the old versions of the configurations. On other hand, interviewee I said that from his perspective, a newer file is often identified only by the date or by asking the author.

The interviewees do not assess the need for the version control in the exact same way. For some, version control appears to be inadequate in its current state, while others emphasize that continuous versioning of unfinished configurations during work is not necessarily appropriate. Interviewee B stated that in unfinished configurations, actual version control does not support work in the same way as in example shared templates where versioning is more important. Interviewee H described in similar way that the importance of version control varies depending on whether we are talking about project specific configurations or reusable templates. This suggests that current practices are not only inconsistent, but also that their necessity is seen differently in different roles.
Regarding the documentation of changes, the interviewees highlighted the need for more common and visible practices. Interviewee D described that it would be a good idea to record changes in a common readme, Excel, or other standardized file format. Interviewee A, on other hand suggested a more centralized change log type solution via Teams, OneNote, or a SharePoint, instead of documenting the changes in different

folders. These observations do not yet describe the current unified process, but they show that current practices are considered at least partly too informal and author based. The practices related to the sharing of the configurations also seem to vary depending on the project. Internally, SharePoint acts as a common sharing and storage environment, but delivery to the customer can take place in different ways depending on the case. Interviewee C described that in some projects the customer has its own portal where all the material and configurations are to be delivered, while others described SharePoint link to the shared folder or via an email.

Overall, current processes and practices appear to be working in a way that configuration management works in practice but has many challenges that can be improved. Current processes are more as established ways of working than strictly defined and uniform processes. The interviews show that there are common features, but the practical implementation still varies depending on the projects, authors and usage needs.

3.5 Identified challenges and development needs

Important challenges of the current state are related to version control, visibility of the changes and the fact that identifying the latest version of the configuration files is not always easily identified. Several interviewees described that in practice the latest version of a file is determined by location, file name or by asking the author who made the configuration. This is sufficient in some situations but does not very well support the fact that changes, their content and the history of the configuration would also be visible in later use. Interviewee D described version control and documentation of the backups and their changes the most important improvements, as the changes made in the versions of the configurations are mostly visible in the configuration description field in the tool where the configuration is made. Interviewee I described that the latest file is often identified only by the date in the file name or by asking the author, which according to him is not a very effective method.

Another strong theme is related to the findability of the configurations, naming and inconsistency of the folder structures in different projects. Configurations are usually found better when a person is familiar with the project or knows who made the configurations. Interviewee E raised a situation where the same configuration was in two different branches, which made the whole situation confusing. Interviewee B emphasized that a more uniform structure would make it easier to find the configurations and other related data and reduce the need to reinvent the wheel.

The interviewees also highlighted that configuration management is partly person-dependent. Finding the right file, understanding the changes and understanding the history of the project may require that the original author or a person who knows the project well is still within reach. This is particularly evident when the system itself does not sufficiently tell you what has been changed in the file, what is it related to or which version is correct for a particular use case. Interviewee I stated directly that he often asks the author which file belongs to which and which is the latest version. Interviewee H, in turn described that there is currently no actual uniform guidance, but that practices are largely determined case-by-case basis.

Not all interviewees found the situation to be equally problematic. Experiences vary depending on the project type, role and context of use. In some more stable or long-term projects, the practices were felt to be sufficient for their own needs, even though they were not considered a common model for all projects. Interviewee F described that in his own large project the situation is quite clear, and no significant problems have been encountered, although the estimates that the situation may be more difficult for those who handle many small projects. Interviewee G stated that from his perspective there have been no major problems, although sometimes the latest backup may have remained on the development side and not immediately ended up in SharePoint after it was done. This shows that development needs are not completely the same for all users, but at the same time the practices need to be less varied.

Templates and standardization of the configuration management seem to be useful improvement especially for those projects that can use repetitive material or when a more standardized approach is sought. Interviewees noted that the benefit of templates also depends on how well their versioning, maintenance and suitability for different projects are managed. Interviewee H emphasized that as templates become more common, their version control should also be in place so that it is known which template is being used and why. Interviewee B considered common templates to be an important future development direction, especially from the perspective of standardized solutions. Interviewee D noted that some projects use readymade templates, but the same practices are not yet uniform across all projects.

Regarding the development needs, interviewees repeatedly highlighted the need for more uniform practices, clear version control and more visible changes to the configurations.

3.6 Summary of current state

Based on current state analysis, configuration management works reasonably well in daily work, but it is partly based on project specific practices, folder structures and personal knowledge. SharePoint currently serves as the main common storage and sharing environment for configuration files and project documentation. However, the overall process is not limited to SharePoint, but files are also processed in local folders, Teams, customer portals and other project specific locations during work. As a result, the configuration management environment is not completely uniform throughout the entire data life cycle.

The main challenges identified in the current state can be summarized as follows:

- Distributed storage environment: configurations and related data can be located in several different locations depending on the project phase, user and delivery method

- Project specific variation: folder structures, naming conventions and version control methods are not uniform across all projects
- Identifying the correct version: the latest or usable version is often identified based on the file name, date, location or author information
- Lack of visibility of changes: the existence of different versions may be visible, but the content, rationale, and impact of changes are not always clear.
- Need for consistent practices and templates: The interviews highlighted the need for clearer structures, common practices and controlled templates.
- Variability in sharing: sharing configurations and related documents works in practice, but the method of sharing varies depending on the project and the customer.

Based on the current state, the development need is not only regarding to where configuration files are stored. It is also important to consider how configurations are identified, versioned, combined with related material, documented for changes and distributed to different users in controlled manner.

4 Definition of System Requirements

4.1 User requirements

The user requirements describe what the improved solution should offer to the configuration management for the users of the configurations in their daily work. The solution should offer support to how each users search, update and share configurations. Therefore, the user requirements focus specifically on how to find the correct configurations, clarity, visibility and usability of the configurations.

It is necessary that the correct configurations, version and location of the configuration should be easily identified. The user should not have to infer the correct version based on the file name, date or unspoken knowledge of other people. The solution should support the presentation and structuring in a such a way that the user can find the right information quickly and understand which configuration is appropriate in a given a certain situation, this same applies to the files related to the configuration files.
Another important requirement from a user point of view is related to the visibility of versions and changes to the configurations. The user should be able to identify which file is latest, what has been changed in it and how the current version relates to previous versions. This does not necessarily mean a heavy or complex process, but above all that the system reduces uncertainty and makes changes more visible than in the current environment. The solution should support users' confidence that the configuration in use is correct and up to date.

The solution should also be usable in a such a way that it supports daily work and does not unnecessarily increase manual workload, as it should be even easier to use than solution before. The solution should be logical and easy enough to use from the perspective of different user groups. Usability also means that the solution should fit the existing operating environment and practical work situations. If the system is perceived as too heavy, pricey, difficult or slow to use, user can circumvent it with informal practices which weakens the unity of the management.

The solution should have smooth and controlled sharing of the configurations. The user should be able easily to share the configurations or any other document with the right people, both internally and externally if necessary. At the same time the system should support the fact that configurations are available when it's needed. In this regard the access rights, and accessibility of the configurations should support the progress of the work and do not constitute and additional obstacle or extra work.

In summary, it can be stated that the key requirements of an improved solution are related to easy discovery of the configurations, identification of correct version, visibility of changes, smooth usability and controlled sharing.

4.2 Process requirements

Process requirements describe how the improved solution should support the working methods and vice versa. The key challenge is that the current processes are varying between different stakeholders and projects as the needs differ a bit. Therefore, the improved solution should support configuration management in a way that it makes the managing consistent and more easily repeatable.

The solution should support common and sufficiently clear working methods between different projects, the goal is not necessarily a complete detailed process, but a common operating logic that reduces unnecessary variation and makes configuration management process more consistent. The configuration management should not be based solely on the authors or projects own working methods.

The process requirements focus specifically on standardizing the practices, clarifying the version controlling and change management of the configurations. This will ensure that the configuration management environment is not only a repository of the individual files but as a whole management of the configurations after they are created.

4.3 Technical and architectural requirements

The technical requirements describe what kind of structural and technical features are required from the improved solution in order to be suitable for configuration management in this current situation the case company is operating. As in what kind of features and compatibility the system must have in order to support the identified needs in practice.

The solution must be a clear and controlled main environment for storing and managing the configurations. The configurations should not be scattered uncontrollably across multiple locations, but the system must provide an unambiguous place where official and usable configurations are located. This does not necessarily mean that all the work to the configurations takes place in the same environment, but after they are created the configurations should be managed in the same system.

The solution should support a data structure that enables the consistent organization of the configurations, identification and findability of the configurations. The structure should be based on a folder tree structure, metadata or a combination of those, as long as the end result supports the logical management of the configurations. It is essential that the system is able to present the configurations in a structure that it does not make it unnecessarily difficult to find the configurations or understand what other files are related to the configurations. The system also should also support the management of multiple projects, document types and configurations in the same environment.

The solution should support version controlling or workflow and the preservation of the version history in a technically reliable way. The user must be able to clearly identify the latest version, and older versions should be available and visible if needed. Version control should not only be based on naming conventions, but the system should support some kind of metadata or naming conventions need to be unified in the system. The

solution should be able to show how the versions differ between configurations and their changes in a structural way.
The system should also support template management, this is crucial for projects that can use templates in a great manner. The templates can be stored in a controlled manner, separated from each other, versioned and templates can be used consistently across different projects. If the use of the templates is essential for the project, the system must also support users so that users can identify which template is correct, up to date and appropriate for the current use case.

The solution should also provide sufficient support for identifying and searching functions for the configurations. This means that the user can use search function for the configurations and the configurations are easily findable based on different properties on the search function so that the correct configurations can be easily found. The system should support the identifiability and discoverability of the configurations, so they are not dependent on the users' own practices only.

The solution should support controlled sharing and support for access right management. The solution must enable the use of configurations by different user groups in a such a way that the configurations are available when needed, but the visibility of the configurations can be restricted, the system must therefore support both internal use and if necessary, external sharing in a way that does not compromise the integrity of the configurations.

The solution should be technically suitable for the case company's operating environment. This means that it must be compatible with the case company's system environment in a bigger picture. The system must meet requirements for the information security and cybersecurity.

In summary, the technical requirements for the system are mainly aimed at the system's ability to handle the management of the configurations based on the needs of the case company.

4.4 Summary of requirements

The challenges identified in the current state analysis form the basis for the system requirements. In Chapter 3, the key problems emerged as fragmentation of knowledge, varying project practices, inconsistency in version control, poor visibility of the changes and the fact that finding the right information often depends on the user's own project knowledge. Based on these observations, the requirements have been derived from three complementary perspectives: user requirements, process requirements and technical requirements.

The user requirements emphasized the ability to find the right configuration and configuration related materials, ability to identify the correct version and visibility of the changes done to the configuration. In addition, the solution must support daily usability and controlled sharing of the configurations. From the user's perspective, it is essential that the system reduces uncertainty in the management of configurations and does not leave the identification of the right information solely based on the file name.

The process requirements emphasized controlled version control, support for standardized project structures, workflow support and support for standardized naming and identification. The purpose of these requirements is to move configuration management away from individual user practices towards more consistent and repeatable operating models. The solution should therefore support common practices between different projects so that versioning, identification and the usage of the project structures are not based only on project-specific solutions.

The technical requirements emphasize data structure management, template management, support for user and document rights management and controlled sharing

of the configurations. The solution should match the case company's current operating environment, and it should be the main environment for configuration management after the configurations are done. The solution should support data structure in a way that support identification of the configurations and ability to visualize the relationships between configurations and related material.

In summary, the requirements defined in this thesis describe the need to move from fragmented and partly person-dependent data management solution towards a more unified, visible and controlled configuration management.

5 Market Review and Alternative Solutions

5.1 Solutions suitable for the use case

This section presents the solutions that were selected for closer evaluation from the perspective of the intended use of this work. The selected solutions have been chosen because they represent alternative approaches to managing configuration-based data. The aim of this review is to form a justified basis for comparison as to what kind of solutions could suit the needs of this work. The selection of the solutions is based on their general suitability for supporting configuration management and requirements presented earlier and that the selected solutions approach the management in a bit different way.

SharePoint represents a document and a collaboration focused approach, which is also closely linked to the current operating environment of this work. SharePoint document libraries are intended for storing, sharing, updating and sharing files and can be extended with version controlling, views and the use of metadata. SharePoint also supports the controlled use of metadata, which allows content to be classified and structured in addition to or instead of a folder structure. SharePoint forms a natural competitor because of this (Microsoft, n.d.-g).

M-Files was chosen for the comparison because it represents metadata driven document management. Solution description emphasized metadata driven and context first thinking, where document management is not primarily based on navigation through folder paths, but on descriptive information attached to the documents. M-Files also emphasized document management, workflow support, version control, permissions and compliance management. Therefore it constitutes a clearly different alternative compared to the SharePoint environment, which focuses on document libraries (M-Files, n.d.-a).

DocuWare is considered as the third option, it was chosen because it represents more controlled document management solution, with an emphasis on centralized document management, version control, workflows and process automation.

DocuWare is a document management and workflow solution, where documents can be stored in a controlled manner, integrated into workflows and version history can be kept visible. Therefore, DocuWare provides a useful point of comparison when more controlled document lifecycle management and process support are in the interest (DocuWare, n.d.-a).

These solutions were chosen for the comparison for multiple reasons. They have a lot of similarities, but they also differ in a way that are useful for this study. SharePoint represents the current collaborative environment based on document libraries, M-Files represents metadata-based document management and DocuWare represents more controlled document and workflow management.

Table 2. Selected systems and rationale for inclusion for comparison.

Solution	Type of approach	Rationale for inclusion
SharePoint	Document-centric collaborative M365 environment	Represents the current environment and shared document library approach, currently in use within the case company
M-Files	Metadata based document management	Represents a metadata-based alternative, used in different teams within the case company
DocuWare	Controlled document management	Represents a more controlled document management approach, an external tool not in use in the case company

5.1.1 SharePoint

SharePoint is a Microsoft platform used by organizations to store, manage and share documents, files and other content. SharePoint document libraries serve as places where files can be created, uploaded, updated, organized and shared with other users. According to Microsoft's guidelines, the purpose of document libraries is to provide a place where users can easily find files, work on them together and access them across devices (Microsoft, n.d.-n).

Structurally SharePoint is based on sites and the libraries and lists within them. A document library can be created empty, based on existing libraries or using a ready-made template. The library can be named and be given a description that makes it easier to use and identify. This shows that the data management logic in SharePoint is built primarily around libraries (Microsoft, n.d.-b).

Data management in SharePoint can be based on a traditional folder tree structure, but the SharePoint also supports structuring based on metadata. SharePoint's managed metadata features allow content to be categorized and described centrally, so that the data management is not tied to a folder path. This makes SharePoint a flexible solution where content can be organized using folders, metadata or combination of those (Microsoft, n.d.-i).

SharePoint's fundamental features also include version control. When versioning is enabled in a library or a list, changes to files can be recorded, tracked and even reverted to the previous versions. SharePoint supports both major versions and a combination of major and draft versions, version control allows you to see the development history of a file over time (Microsoft, n.d.-h).

In addition to version control, SharePoint also supports more controlled document processing which can be achieved through a check-out and check-in functions. The purpose of these functions is to control who edits a file and when changes are returned

to a common use. These features are part of a situation where more control over document processing is desired than just a free co-authoring alone offers (Microsoft, n.d.-l).

Most important features SharePoint offers from a perspective of this study are related to the document libraries, version control and the use of metadata. These features are particularly relevant because they are directly related to the storage of the configurations, visibility of versions, structuring the data and sharing of the configurations. SharePoint is therefore a justified option.

SharePoint is included in this study comparison not only because it represents a knowledge management tool based on document libraries, but also because it is already part of the current operating environment. One realistic development direction is not only the implementation of a new system, but also the development of the current SharePoint environment so that it could better meet the requirements and development needs identified in this study.

5.1.2 M-Files

M-Files is a document management platform that is used to manage documents, data and data related processes in an organizational environment. M-Files platform is a metadata driven document management platform, where document management, automation and compliance are connected to the same environment. M-Files data management is based on metadata. As in older platforms, in M-Files the document management is not primarily based on navigation through folder paths, but on attaching descriptive metadata to documents. Based on that, they are structured, searched and viewed in different contexts. M-Files can be seen as context-first for this approach (M-Files, n.d.-a).

In that sense, discoverability of the data in M-Files is based on what the document is, not just where it is located. Metadata is utilized to organize, search and manage data, which makes the metadata theme a very compelling topic (M-Files, n.d.-d). Version controlling can be used in M-Files. The system can store previous versions of objects and the user can view the object's history and revert to a previous version if necessary (M-Files, n.d.-l). M-Files also supports workflow and lifecycle management of the document. The system can model document workflows in accordance with real processes. The workflow is divided into states that describe the different stages of a document or other object (M-Files, n.d.-n). M-Files also integrates security, permissions and audit controls into document management. Security, permissions, retention and audit controls are built-in features that are applied throughout the document lifecycle (M-Files, n.d.-a).

The key features of M-Files in a perspective of this work are particularly related to the metadata-based structure, version control, workflow support and a lot of other possibilities among those. These features are relevant when considering a solution that should support the ability to find the configurations, version control the configurations, visibility of changes and controlled management. M-Files is included in this comparison because it represents a clearly different data management tool than SharePoint currently in use. Based on the features known, it forms a relevant point of comparison for this work. It must be noted that this solution is in use in other departments of the case company and for that reason has a slight edge for next candidate.

5.1.3 DocuWare

DocuWare is a document management and workflow solution used for processing, storing, managing and tracking documents in an organizational environment. The system is being presented as solution that allows documents to be collected, managed and used centrally. DocuWare's data management logic is based on centralized document management. In DocuWare, documents can be brought into the same environment where they can be managed, processed and tracked (*DocuWare, n.d.-d).*

DocuWare emphasizes workflow management as part of document management. The solutions workflow management supports the planning, management and tracking of workflows and the ability to control the processing steps of the documents (DocuWare, n.d.-b). DocuWare seems not to be just a document storage environment but a solution where documents can be integrated into the work processes. Emphasizing workflow suggests that document processing can be controlled in stages and in a more controlled manner than in case of a simple file storage.

DocuWare's features also include version controlling of the documents, version control is presented as a way to manage different versions of documents, reduce errors and ensure that up-to-date information is available (DocuWare, n.d.-c).

The most important features regarding this thesis with DocuWare are related to centralized document management, workflow support and version control as part of document processing. These features are particularly relevant when solution should support controlled management of configurations, process consistency and more visible data management. Based on features described above, DocuWare is a solid candidate to comparison between different solutions. It has to be noted that it has little bit of a disadvantage to others as it is completely new solution to the case company, which brings a lot of questions whether it's possible to integrate the solution to the case company, how costly it is and in which timeframe.

5.2 Comparative analysis

SharePoint and M-Files are the solutions compared in this section. The comparison is based on the user requirements, process requirements and technical requirements defined in the previous chapter. These requirements are derived from the current state analysis, interviews with stakeholders and objectives of the thesis. This is why the

comparison focuses specifically on how well the selected solutions could support configuration management in this limited environment.

Although the previous chapter examined SharePoint, M-Files and DocuWare, the more detailed comparative analysis of this thesis focuses primarily on SharePoint and M-Files. The justification is that these two solutions are clearly the most relevant further options from the perspective of the case company's current operating environment and realistic nature of implementation.

This comparison constitutes a key analysis stage for the thesis, as its purpose is to assess the suitability of the solutions for the identified needs, the outcome may not be the best management solution in general, but a solution that would match the case company's needs for this solution. It must be noted that in this comparison, the suitability of the solution for the case company is heavily favoured because it creates a high chance that this thesis will be usable for the case company.

These perspectives can be used to assess the extent to which different solutions could support configuration management in different stages, supporting user work, management processes and system level technical requirements.

The comparison is structured through three requirement categories. First the solutions are compared from the perspective of technical requirements, after which they are compared based on user requirements and lastly process requirements. This structure will make the comparison consistent with the requirements seen earlier. These results will support the decision how the solution concept will form.

To structure the comparison, evaluation criteria is created for the identified requirements. The relative weighting of the criteria is based on how strongly they emerged in the empirical data and how relevant they are to the team. The weighting has not been formed using a separate decision-making model but is based on data-driven

assessment of the significance of the requirements. Solutions are assessed for each criterion using uniform scoring scale so that alternatives can be compared consistently.

The weight of the requirements was determined based on their relative importance from the perspective of current state analysis, interviews with the stakeholders and work objectives. The highest weighting was given to those requirements that were most clearly repeated in the data and that the requirements were most central to the configuration management problems.

The comparison is also limited on the data available in the time the thesis is done. Therefore, the results describe the estimated suitability of the systems based on available material and are not based on full-scale testing.

Table 3. Comparison weighting and scoring.

Weight	Explanation	Score	Explanation
3	High importance	5	Meets the requirement or exceeds
2	Medium importance	4	Meets the requirement well
1	Low importance	3	Meets the requirement satisfactorily
		2	Meets the requirement partially
		1	Meets the requirement poorly

5.2.1 Comparison through technical requirements

Comparison through technical requirements focuses on data structure management, template management, access rights, managed sharing and solution's suitability for the case company's current environment.

Table 4. Technical requirements.

Requirement category	Requirement area	Importance in comparison	Weight
Technical requirements	Data structure management	How flexibly does the solution support folders, metadata or a combination?	3
Technical requirements	Template management support	How well does the solution support the use and maintenance of templates?	2
Technical requirements	Permissions and controlled sharing support	How well does the solution support the internal and external sharing of the configurations?	1
Technical requirements	Suitability for the company environment	How well does the solution fit into the case company's existing practices?	3

In the terms of data structure management, SharePoint is based on document libraries which act as a place to store, find, share and manage files. Document libraries can also use folders and view data related to the files (Microsoft, n.d.-n). SharePoint also supports managed metadata, where metadata can be centrally managed and used to categorize content. Managed metadata can be based on term stores, term sets and managed metadata columns, which can be used to structure content alongside the folder structure. In SharePoint, metadata navigation allows lists and libraries to be browsed based on

metadata, rather than just folder paths. This means that structure management can be based on folders, metadata or a combination of both (Microsoft, n.d.-i).

In M-Files, structure management is based on object metadata. Metadata contains information about the object, such as name, category and time of creation. The amount of metadata affects how easily an object can be found (M-Files, n.d.-f). Metadata is managed with metadata card, which displays the key information of the object, such as title, time of creation, author and category. The metadata card gathers the information relevant to the structure in one view. The vault metadata structure is used extensively in various functions of the systems, such as views and search functions. This emphasized that the logic of the structure in M-Files is primarily based on the properties of the objects and not on their location in the folder structure (M-Files, n.d.-c).

SharePoint appears to be flexible option when the structure is desired to be based on a folders, but the solution also supports metadata if needed. M-Files on other hand emphasized a more clearly metadata-driven structure, where the document properties form the core of the structure.

From a template management perspective, SharePoint document libraries can be created from scratch, based on an existing library or using readymade template (Microsoft, n.d.-b). In SharePoint multiple content types can be associated with the same list or library using content type structure. When a content type is added to the library, it also brings with it the associated document template and other definitions (Microsoft, n.d.-a). Content types can be used to manage content consistently across an entire site or site collection. SharePoint seems to support template management, especially when document templates are wanted to be included as part of the structure of libraries and content types.

In M-Files, documents and projects can be created using templates. When creating a new document in M-Files, the user can select a template and category, predefined

content and metadata can be associated with templates (M-Files, n.d.-k). Template is defined per class, and the same template can be associated with multiple classes using the “Additional classes” definition (M-Files, n.d.-e). Therefore, M-Files structurally supports template management quite strongly, because the template is directly related to the document creation, class and metadata.

From a permissions and managed sharing perspective, SharePoint allows to manage permissions at the site, library, folder and individual file level. A library can either inherit permissions from its parent or use its own permissions that are different from its parent. Files and folders can be shared either by invitation or by link, and sharing can be limited to specific users. Permissions and sharing can also be removed later if given unnecessary (Microsoft, n.d.-d). SharePoint also supports sharing outside of the organization when external sharing is allowed in the environment settings (Microsoft, n.d.-f). This means that the same solution supports both internal and external controlled sharing.

In M-Files, object specific permissions can be viewed and edited via the object’s metadata card. Permissions can also be selected using ready-made named access control list definitions (M-Files, n.d.-g). Permissions can also be determined automatically based on metadata. Automatic permissions can be associated with a category, value item, object type or metadata value. The final effective permissions of an object are formed by the interaction of the object’s own permissions and automatic permissions (M-Files, n.d.-i).

SharePoint seems to support permissions and controlled sharing flexibly at different levels and both internal and external sharing. M-Files emphasized the structural control of permissions, where permissions are directly related to objects and their metadata. SharePoint stands out as flexible sharing environment.

In terms of suitability for case company’s environment, SharePoint has a clear head start because based on the current state analysis it already serves as main environment for

configuration management. Therefore, developing current SharePoint appears to be natural option when the goal is to improve current practices within existing environment. M-Files on the other hand represents a different, metadata-based approach to document management. It is not completely new to the case company because it is used in other parts of the organization. Therefore M-Files can be seen as realistic option, even though its implementation would require more changes to current practices than the further development of SharePoint. SharePoint seems to be better suited to the case company's current practices, while M-Files appears to be realistic option as well.

5.2.2 Comparison through user requirements

From a perspective of a user, the key benchmarks are how easily user can locate correct configuration and acknowledge its version and see changes made to the configurations compared to previous one.

Table 5. User requirements.

Requirement category	Requirement area	Importance in comparison	Weight
User requirements	Findability of correct configuration and related material	How easily can the correct configuration and related material be identified	3
User requirements	Identifying the correct version	How clearly can user identify the correct version?	3
User requirements	Visibility of changes	How easily can the user see what has changed in the configuration?	2

First requirement examines how easily the user can find the right configuration and related material and how the solutions support this.

In SharePoint, the findability of the right configuration and related material is supported by document libraries, where files and folders can be stored in the same place where key configuration related material is. In document libraries, you can also create your own views and add links to content located outside the library. This supports the fact that the configuration and related material can be compiled in the same managed environment for the user. SharePoint also supports the findability by managed metadata functionality, where content can be classified using managed terms and metadata columns. This means that in the same library, content can be structured using metadata in addition to the folder structure and metadata navigation also enables browsing the library based on the metadata.

In M-Files finding correct configuration is supported by the search function and views. Search is the primary way to find objects when the user only knows part of the configuration information such as name, author or some other property (M-Files, n.d.-j). Objects can be linked between each other representing relationships between each other which eases the work to find related objects (M-Files, n.d.-h).

From the perspective of findability of configurations and related documents, both solutions offer a catalogue of solutions to tackle this issue.

Second user requirement examines how easily can the user identify the correct version of the configuration and how does the solution support the version controlling.

In SharePoint, version control can be enabled in a list or library allowing the user to store, track and restore previous versions of a file. SharePoint supports draft and major versions (Microsoft, n.d.-h). Use can open a file's version history from library. Version history shows previous versions of the file and the user can view and restore previous

versions. When a previous version is restored, it becomes the latest version of the file and the previous history information is not deleted (Microsoft, n.d.-m).

In M-Files each object has its own version history. The solution also preserves the previous versions of objects. The user can open the object's history from the metadata card and restore previous versions (M-Files, n.d.-l).

From the perspective of how the solutions supports that the user can easily see what version user is looking for, both SharePoint and M-Files support identifying the correct version by version history. In SharePoint version identification seems to be more strongly tied to library-specific settings and in M-Files the version history is directly related to objects management.

From the perspective of change visibility, the key is how easily the user can see what has changed in the configurations between different versions and how does the solution support the fact that there needs to a trace of the changes done to the configuration.

In SharePoint, version history shows previous versions of file as well as information about when the file was changed and who made the changes (Microsoft, n.d.-m). SharePoint's version control makes changes visible in the sense that the user can view different versions of a file in the library. However, the visibility of the content of changes depends more on how version control is used and whether changes are documented separately as part of the process (Microsoft, n.d.-h).

In M-Files, object history shows previous versions of an object and the user can view the version history and restore previous versions if needed (M-Files, n.d.-l). In M-Files, object activity also shows comments, metadata changes and file changes related to the selected object. This makes changes more visible to the user, because in addition to the

existence of the version, the information what has been changed is visible in the same entity (M-Files, n.d.-m).

SharePoint seems to support the visibility of changes through the version history information such as time and showing the editor. M-Files supports the visibility of changes more broadly, because in addition to the version history, the user also sees comments, metadata changes and other object activities.

5.2.3 Comparison through process requirements

From a process requirements perspective, the key is how well the solution supports controlled version control, standardized project structures, workflow and standardized naming and identification. These requirements are used to assess how well the solution supports configuration management as part of common and repeatable practices across projects.

Table 6. Process requirements.

Requirement category	Requirement area	Importance in comparison	Weight
Process requirements	Support for controlled versioning	How well does the solution support version controlling?	3
Process requirements	Support for standardized project structures	How well does the solution support creating unified structures for different projects?	3
Process requirements	Workflow support	How well does the solution support controlled handling and progression of configurations?	1
Process requirements	Support for standardized naming and identification	How well does the solution support standardized naming and identification?	2

From a managed version control perspective, it is crucial how well does the solution support storing of old versions, version retention and the use of version control as part of shared practices. This includes whether version control is based on the systems own function or whether it relies more on user naming and local practices.

In SharePoint, version control can be enabled on a list on a list or a library (Microsoft, n.d.-h). Library version control settings can be used to specify the number of versions to be saved, the ability to present draft versions and whether the documents are required to be checked out before editing (Microsoft, n.d.-e).

In M-Files, previous versions are retained as part of the object's version history and the user can view previous versions and restore older versions (M-Files, n.d.-l). Changes made to the object's metadata are also saved as a new version. Therefore version control is directly related to the entire object's management and just not separate file versions (M-Files, n.d.-f).

Based on these observations, SharePoint seems to support managed version control through library-specific settings and version control functionality. M-Files integrates version control more directly into the entire object's lifecycle and metadata-based management.

In standardized project structures, the key principle is how well the solution supports the use of consistent structures across different projects. This means that the basic logic can be reused across all created projects without having to build each project from scratch.

In SharePoint, the structure of the project can be implemented using document libraries folders and subfolders. SharePoint supports copying the files and folders to another location in the same site, this supports that existing project structure can be used as the basis for a new project (Microsoft, n.d.-j).

In M-Files, standardization is done more on the vault's metadata structure. In M-Files, classes are defined as part of the metadata structure and document template can also be defined for a class. The same template can be attached to multiple classes which supports the creation of similar project structures (M-Files, n.d.-c).

By the basis of standardized project structures, SharePoint supports project structures more directly when the goal is to use ready-made folder and library structures. M-Files supports standardization of project structures more at the level of metadata, categories and document creation.

From a workflow support perspective, the key is how well the solution supports controlled progression of a configuration through different phases. This is particularly related to whether objects can be guided from one phase to another using the system's own functions and how well the solution supports controlled processing as part of its operating methods.

In SharePoint, approval functions can be implemented in the document library, whereby new or changed files can be set to wait for approval before they are visible to wider range of users (Microsoft, n.d.-k). In addition, Power Automate workflows can be used in SharePoint, which can be used to automate approvals, notifications and other steps related to document processing. This extends the workflow support beyond the basic functions of the actual document library (Microsoft, n.d.-c).

In M-Files, workflow is directly related to objects and their lifecycle. In the system, workflow consists of states that describe different stages of an object and objects can move from one state to another through predefined transitions (M-Files, n.d.-n). State-specific functions, user roles and rules can also be used in connection with workflow, which can be used to control the processing of an object in different stages (M-Files, n.d.-b).

From the perspective of workflow requirement, SharePoint supports workflow especially through approval functions and Power Automate solutions. M-Files connects workflow more directly to the management of objects and their step-by-step progression.

From a standardized naming perspective, it is crucial to determine how well the solution supports consistent identification of configurations and related data across different kind of projects. This includes whether the identification is based on the file name and folder structure or whether the solution also supports it through metadata, categories or other structural features.

SharePoint supports identification through document libraries, content types and managed metadata functions. Identification seems to be quite strongly related to the library structure, file names and how the policies have been implemented in the solution.

In M-Files, identification is based more strongly on the objects metadata, classes and metadata structure which are used extensively in the solution for example in views and search functions. This reduces the dependency on the file name alone and supports structural identification.

5.3 Summary of comparison

Market review and comparative analysis show that both SharePoint and M-Files have clear strengths in relation to requirement defined before. SharePoint is highlighted especially when the emphasis on compatibility with the current operating environment, the use of controlled sharing, document libraries and folder structure usage. M-Files appears to be strong from the perspective of metadata-based management, version control, change visibility and more structured control.

To support these findings, the solutions were scored on a requirement specific scale of 1-5 according to the previously defined scoring model. The weighted score for each requirement was formed by multiplying the given score by the requirement's weight, whereby requirements with higher weighting have a greater impact on the overall assessment. The total sum is the sum of the weighted scores.

Table 7. Weighted comparison of SharePoint and M-Files.

Requirement	Weight	SharePoint	SharePoint Weighted	M-Files	M-Files Weighted
Findability	3	4	12	5	15
Correct version	3	4	12	5	15
Visibility	2	3	6	5	10
Controlled versioning	3	4	12	5	15
Project structures	3	5	15	3	9
Workflow support	1	3	3	5	5
Naming and identification	2	3	6	5	10
Data structure	3	5	15	4	12
Template support	2	4	8	5	10
Controlled sharing	1	5	5	4	4
Company environment fit	3	5	15	3	9
Total			109		114

M-Files received slightly higher overall score than SharePoint. However, the difference is not large, which suggests that both solutions meet several of the requirements defined in this thesis quite well. The result does not show that one solution is unambiguously better in all areas, but rather that the strengths of the solutions are slightly focused on different aspects. SharePoint received higher scores especially from the perspective of standardized project structures, controlled sharing and suitability for case company's operating environment, while M-Files received higher scores especially from the perspective of visibility, correct version identification, change visibility, controlled version management and standardized identification.

Based on the comparison, both solutions offer a wide range of functions that can meet the requirements quite well. However, on practical level, the functionalities of the solutions does not only depend on the system characteristics, but also how the environment is defined, how the data structure is implemented and how consistently common practices are followed. The result of the comparison describes both the characteristics of the solutions and how naturally they support the construction of managed practices in the environment of the case company.

The choice of what solution is selected is not determined solely by the total score, but also by which requirements are to be emphasized most in further development. In the following chapter, these observations are used as starting point for forming a solution concept and assessing which implementation method would best meet the needs identified in this thesis.

6 Development of Solution concept

The starting point of the concept is the challenges identified in the current state analysis, the defined user, process and technical requirements and the comparison of alternative solutions. Based on the comparison, both SharePoint and M-Files meet several of the requirements defined in this work, but their strengths are focused on different areas. M-Files was chosen as the basis for the solution concept and the pilot. It has great benefits from the metadata-based management logic, object relationships, alternative way of version control, workflow functions and view and search functions for the findability. The justification is that it presents real alternative for the current solution used and presents varies ways of implementing features on different level that in the current solution.

The goal of the solution concept is to present how configurations can be managed in a more unified, traceable and less person-dependent way than currently. In the concept, the configuration is not treated as a single file or a document placed in a folder structure, but as a manageable object to which identification information, technical version information, related materials, change information, approval and publication states can be attached. In this way, the technical content of the configuration, its context and changes during its life cycle can be compiled into a manageable whole.

First, we look at key principles of the solution concept, then examine the concept's structure, system logic and process. The concept is then illustrated using a pilot built into the M-Files test vault. The pilot is not intended to be a readymade production solution, but to show how the key elements of the concept can be applied in practice.

6.1 Principles of the concept

The first principle of the solution concept is that configurations are treated as manageable objects and not just as individual files. This means that a configuration file

is part of larger entity, to which metadata describing the configuration, workflow steps describing its status, and connections to other documents can be attached. In this case, configuration management is based on what information about the configuration is stored and how it is related to other parts of the entity.

Second key principle is metadata-based identification and findability. Configurations must be findable based on, for example, project, product, configuration type, switchgear type, version and status. This way, the user does not have to identify the correct configuration based solely on the file name, date, or folder location. Metadata can be used to present the same configuration in different views for different purposes, such as project specific views, approved version views or deliverable document views.

The third principle is to make the relationships between configuration and its related material visible. Configurations often involve other materials, such as lists, specification documents, templates or other background documents. In the solution concept, related material can be linked to the configuration using a metadata relationship, allowing the user to see the related documents in connection with the configuration. This reduces dependency on the user's own memory or project specific experience.

The fourth principle is controlled revision and visibility of changes. In configuration management, it is important to distinguish between the system's internal change history and the technical configuration version that is relevant to the user. In this concept, the technical version is described as metadata, while workflow and version history support the management of the document's processing stages and editing history. Change information is linked to the configuration using a separate change log document, allowing the content, cause and effect of change to be documented more visibly than based on the file name alone.

The fifth principle is to use views instead of copying folder structures and files. Configurations and related material do not need to be moved to multiple parallel

locations for different purposes, but the same objects can be presented in different views based on metadata. For example, same document can have several locations based on the metadata or deliverables view can gather documents belonging to delivery together without having to copy them to a separate folder. This supports data integrity and reduces the risk of uncontrollable copies of the same document.

The sixth principle is related to templates and reusable project structures. In the solution concept, project initiation can be supported by project templates that already include placeholder objects related to configurations, change log templates and other documents related to the project. In addition, separate configuration template logic can support the controlled use of recurring configurations or more document templates related to configurations. This supports more consistent practices between different projects.

The aim of these principles is to reduce problems observed in the current state, such as project specific variation, uncertainty in identification the correct version, poor visibility of changes and personal dependency. The central idea of the concept is that configuration identification, discoverability, revision, related material and sharing are based on shared metadata and system management logic.

6.2 Structure and system architecture

The structure of the solution concept is based on the M-Files vault environment, where configuration related information is managed using objects, classes, metadata, workflow and views. In the concept the vault acts as the main environment for configuration management after the configuration has been created and it is intended to be brought into controlled use. The actual technical creation of the configuration still takes place in separate configuration tool, but M-Files acts as an environment where the finished or unfinished configuration is identified, linked, revised and shared in a controlled manner.

The central structural element of the concept is the configuration class. This allows configurations to be distinguished from other document types. An object belonging to the configuration class contains the actual configuration file and the metadata describing it. Metadata that is relevant for the configuration can include properties such as project, project section or gate, configuration type, switchgear type, configuration version, related material and change log properties. These allow the configuration to be linked to the correct project, product, technical purpose, revision and related material.

The M-Files metadata card forms the central user interface for configuration management in the concept. With the metadata card, the user can view and update information related to the configuration, such as its class, project, configuration, technical version so that the configuration identification is not based solely on the file name, but on the structured data maintained in the system.

Related material is implemented in the concept as a metadata relationship and not as separate document class. This means that various documents can be linked to the configuration regardless of which class they belong to. Related material can be a list, definition document, a template or other material important for understanding the configuration. The solution supports flexibility, because the material related to the configuration can be located in the same project or also in other projects without needing to copy the files into the same folder.

A separate change log class is used in the concept to manage changes of data. The change log document is linked to the configuration with a metadata relationship, allowing the description of changes, cause, effect and identification information related to the change to be stored in a connection with the configuration. This separates the documentation of changes from the actual configuration file but still keeps it part of the same managed entity.

The concept also includes workflow structure, the purpose of which is to distinguish between working, reviewable, approved, published and outdated configuration versions. The workflow does not primarily function as a generator of technical version number but describes the processing and publication status of the configuration. The technical configuration version, such as version number or revision ID, is managed as a metadata. In this way, the internal version history of M-Files, the workflow status and technical revision relevant to the user can be considered separate but mutually supporting data.

The consistency of projects is supported by project template logic. A project template can contain ready-made placeholder objects, such as a configuration template object and its associated change log template. When new project is created based on a template, these objects that are essential for management can be included at the beginning of the project. In addition, a separate configuration template logic can act as a managed source for reusable configuration templates.

Views form the final key part of the structure. In M-Files, the same object can be displayed in several different views based on definition of the view. In concept, views can be used to present project specific configurations, approved versions, templates or deliverables material for example. This reduces the need to maintain parallel folder structures or copy the same files to multiple locations. Views allow the user to view the same managed information for different purposes without disconnecting the original object from its metadata connections.

Overall, the solution structure is based on that the configurations, related material, change information, revisions and deliverable documents are connected to each other using metadata and object relationships. Such structure supports configuration management more broadly than simply storing files in a folder structure, because the system can describe both the technical context related to the file’s content and its administrative status and relationships to other documents.

6.3 Functionalities and process description

The process is based on that the configuration is managed in M-Files as an object, the identification, findability, processing of related material, change information and sharing of which are supported by metadata, workflow and views. Figure 2 shows the top-level process of the solution concept, which is divided into preparation, management and access and sharing.

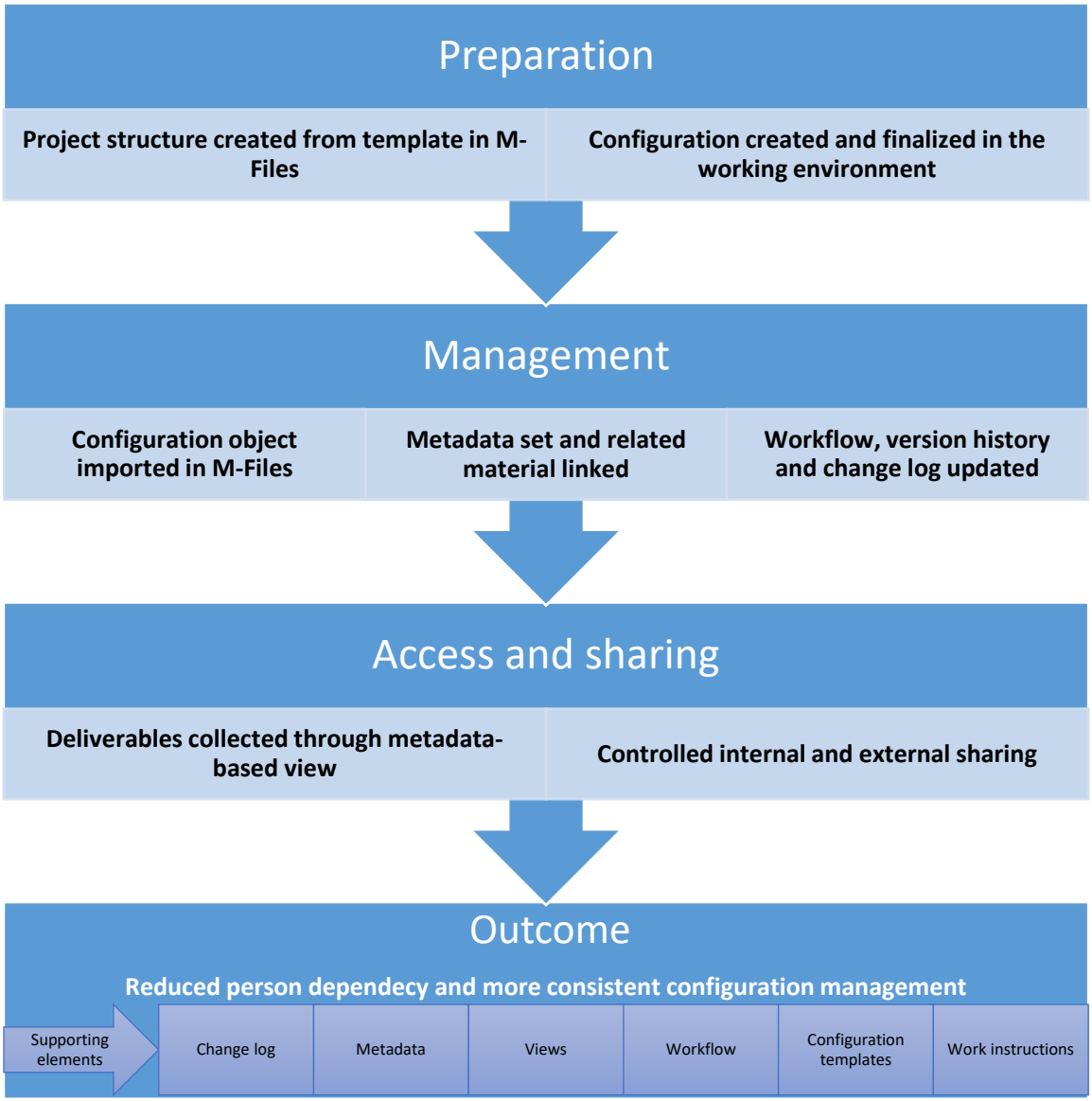

Figure 2. Process description for configuration management in M-Files.

The first phase of the process is preparation. In this phase, a uniform structure is created for the project in M-Files based on the project template. The project template can contain readymade placeholder objects related to the configurations, such as configuration template object and the change log template. This supports that at the beginning of a new project, objects relevant to configuration management are ready to use. However, the actual configuration file can still be created and finalized in separate working environment before importing it to M-Files. It can be copy pasted on top of the template.

The next phase is management, where the configuration is imported into M-Files as a managed configuration object. In this phase, the configuration is assigned with metadata properties that are essential for its identification and use, such as switchgear type, configuration version and links to related material and change log. With the help of metadata, the configuration can be linked to the correct project, project phase, change log and related materials. The management phase also deals with workflow, version history and change information. In this case, the configuration under development, the approved revision, the old revision and the retired version history support viewing the object's modification history, while the technical configuration version and the related change explanation can be described in the metadata and change log document.

The third phase is access and sharing. In this phase, configurations and related documents can be accessed through different views. Views group objects based on definitions such as metadata properties, project, created by or if an object belongs to deliverables list. The deliverables view can gather documents belonging to the delivery together without having to copy them to a separate folder. This supports data consistency and reduces the risk that the same document will start to live in several different places as different versions.

From sharing perspective, the concept supports both internal and external use. Access can be controlled using user rights and views. In internal sharing, there are several ways

to share, through views for example. For multiple document delivery, the concept can be extended with a delivery package and deliverables index so that metadata-based context is preserved outside M-Files

The supporting elements ensure that the process is also based on common practices. Overall, the goal of the process is to reduce personal dependency and improve consistency in configuration management across projects.

6.4 Pilot implementation

This subsection describes the piloting of the proposed M-Files based solution concept. The purpose of the pilot is to examine how the principles, structure and process description presented in previous subsections can be applied in practice in the M-Files environment. The pilot was implemented in a test vault, and its goal was to illustrate the key functionalities of the concept from the perspective of configuration management. The pilot is therefore not viewed as readymade production solution, but as a proof-of-concept implementation, which can be used to evaluate the functionality of the concept on a practical level.

The pilot focuses on those areas that emerged as key in the current state analysis of the work and definition of requirements. These were in the pilot, configuration management as class of its own, the use of metadata in identification and findability, linking related material and change log documentation to the configuration, workflow-based revision handling, the use of project templates and the demonstration of views.

6.4.1 Pilot environment and project template

The vault was created based on existing settings and modified for the purposes of the pilot within the available permissions and configuration possibilities. Test vault enabled

the solution concept to be tested in a limited environment without affecting the data or processes in production use. The goal of the pilot was to illustrate the key elements of the concept.

The project launch was demonstrated using project template logic. The template included placeholder objects that are essential for configuration management, such as the configuration template and the associated change log template. These can be used to start a new project with more consistent initial structure, where configuration related management elements are included right from the start of the project.

The initial project template used in the pilot is shown in Figure 3. Figure 3 illustrates how configurations and related document templates can be included in the project template before a new project is created. Figure 4 shows a test project created from a template and demonstrates how the predefined structure and workflow status can be applied to project documentation.

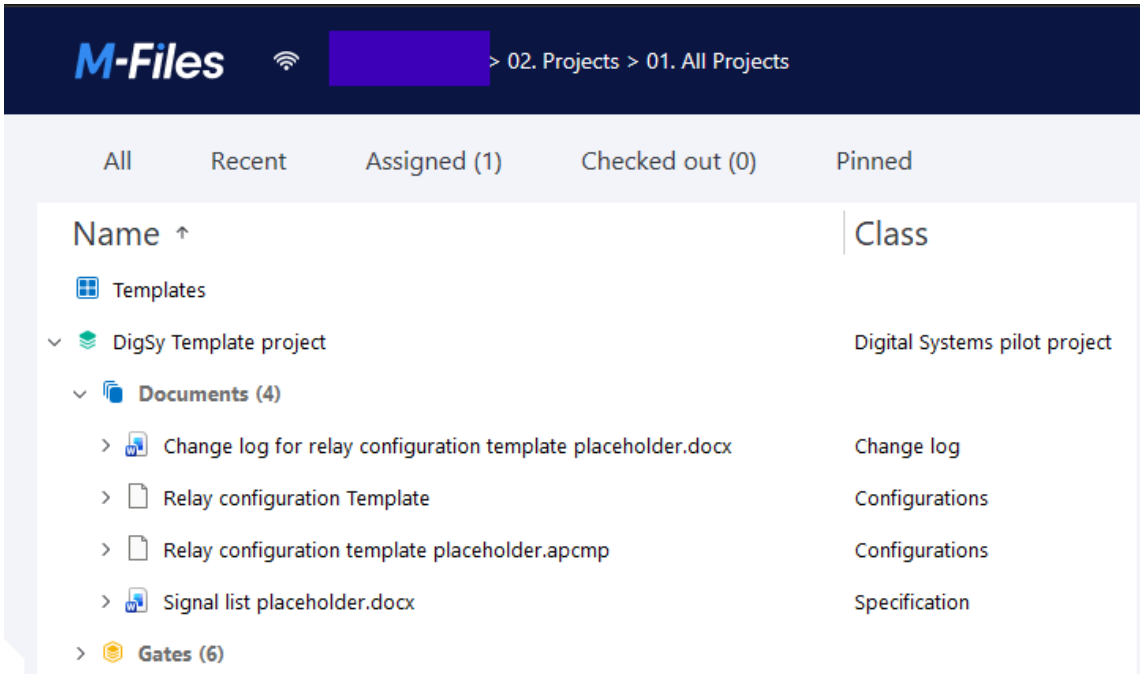

Figure 3. Initial project structure as a template.

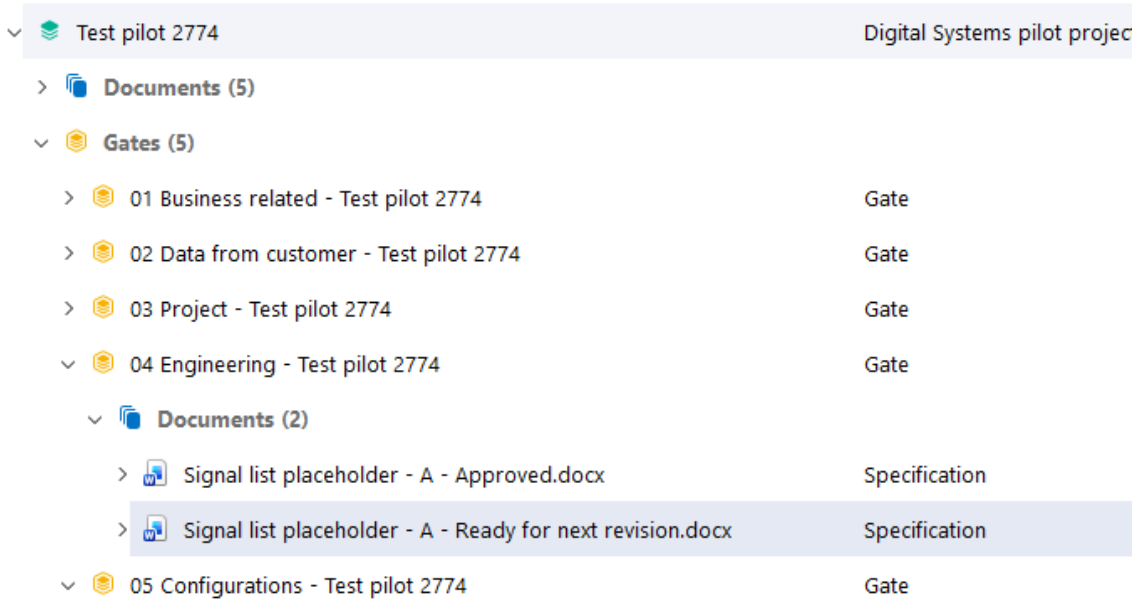

Figure 4. Project created from the template and workflow applied to Signal list.

6.4.2 Configuration object, metadata and relationships

Configuration class was created for the configuration, which distinguished it from other document types. The configuration object contains the actual configuration file and the metadata describing it. The key metadata for the configuration in the pilot were configuration type, switchgear type, configuration version, project or project gate, related material and change log. These properties allowed the configuration to be identified, linked to the correct project and linked to related material as shown in Figure 5.

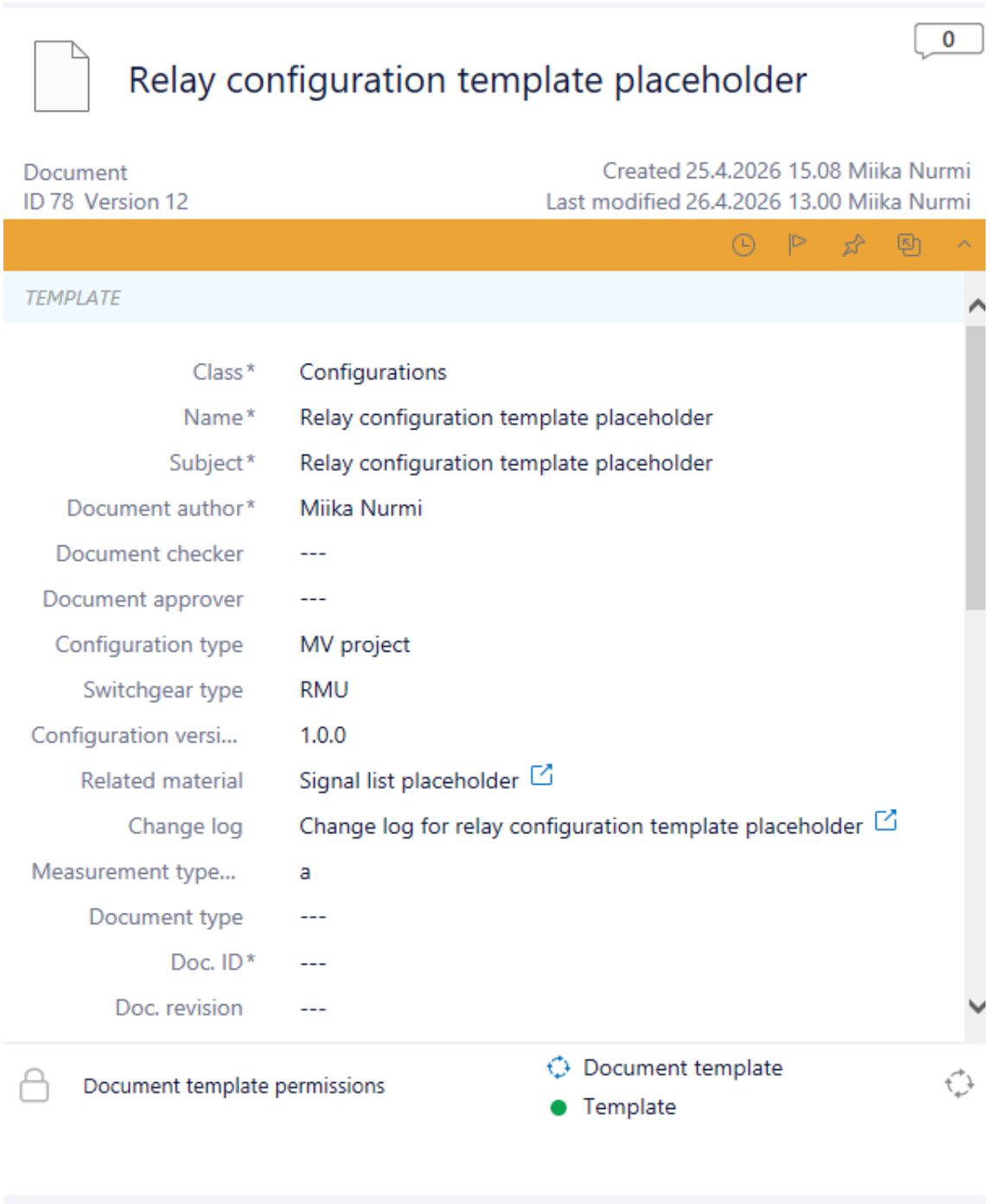

Figure 5. Metadata card properties for configuration class object.

Change log was implemented as a class of its own, which distinguishes the documentation of changes from other document types. Change log was linked to the configuration using metadata property. One key element was related material, which refers to any documentation which is somehow linked to the configuration and is valid

information. Related material was implemented as general metadata relationship property, which allowed documents belonging to different categories to be linked to the configuration. Such materials could be lists, specifications, documents or other project material.

6.4.3 Workflow and revision handling

Configuration approval and revision handling were demonstrated using a workflow.
The workflow is intended to distinguish between a configuration in progress, a version to be reviewed, an approved version, a published revision and an outdated or retired version. Therefore, the workflow is not used to generate a technical version number, but to manage the processing and publication status of the configuration. The technical configuration version, such as the version number is stored as separate metadata.

The workflow proceeds through the document approval stages. The configuration starts in the draft state, after which it can be moved to review and approval. After approval, the configuration becomes a released version, which is retained as an approved and published version. When the next revision is prepared, the new revision continues from the ready for next revision state back to the draft state and proceeds through review and approval again. The previously approved revision can then be marked as an old revision, which serves a similar purpose to the previous archive folder practice, but the revision remained connected to the object's history and metadata structure. The workflow shown in Figure 6 illustrates the difference between a working version and a released revision.

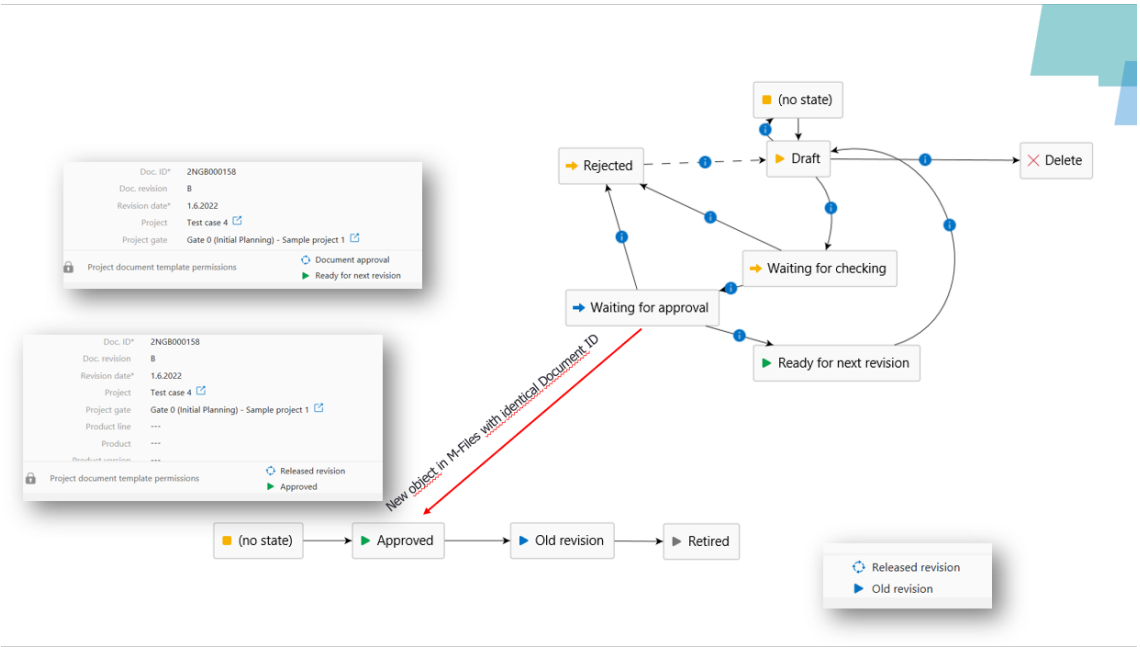

Figure 6. Workflow in M-Files.

M-Files internal version history supported this process by viewing the object's modification history. It is also possible to revert an object to previous state. However, the actual technical version control of the configuration was based on metadata, workflow state and change log documentation. This way, the pilot distinguished difference between the systems internal change history, the configuration version that was relevant to the user and the approval-based publishing process. Figure 6 shows the workflow and revision handling used in the pilot.

6.4.4 Views, deliverables and sharing

Views were used to gather information related to configurations and project documents for different purposes. In M-Files, views can be created based on object metadata and structural definitions, allowing the same object to be presented in multiple contexts without having to copy it to separate locations. In the pilot, this was illustrated with the deliverables view, which gathered the material intended for project delivery.

The purpose of the deliverables view was to demonstrate that objects related to delivery can be collected in one view without separate manual copying. In the pilot, the deliverable's view was implemented as project specific view, where the objects included in the delivery were available from the same place. This supports idea that the material to be delivered can be defined using the system structure and metadata, instead of the user manually creating a delivery folder, as shown in Figure 7.

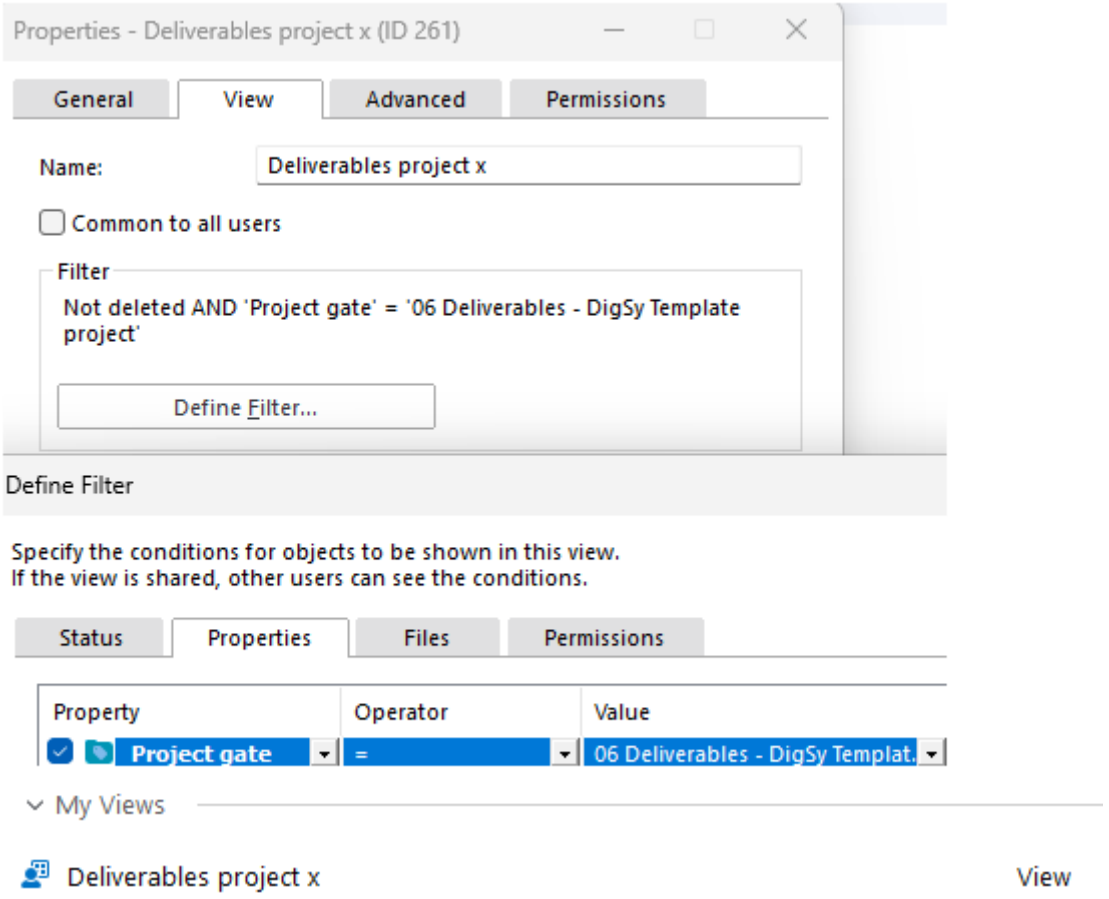

Figure 7. Properties of deliverables view.

Sharing was tested in the pilot from the perspective of both internal and external sharing. Internally, documents can be shared via M-Files quite easily to users with access to the vault via views, "Copy Link" or exporting documents in the view. For external sharing, the pilot tested the visitor link function for a single document. This way a link to a single

document can be shared via “Create Visitor Link” which then user can set an expiration time for the visitor link as shown in Figure 8.

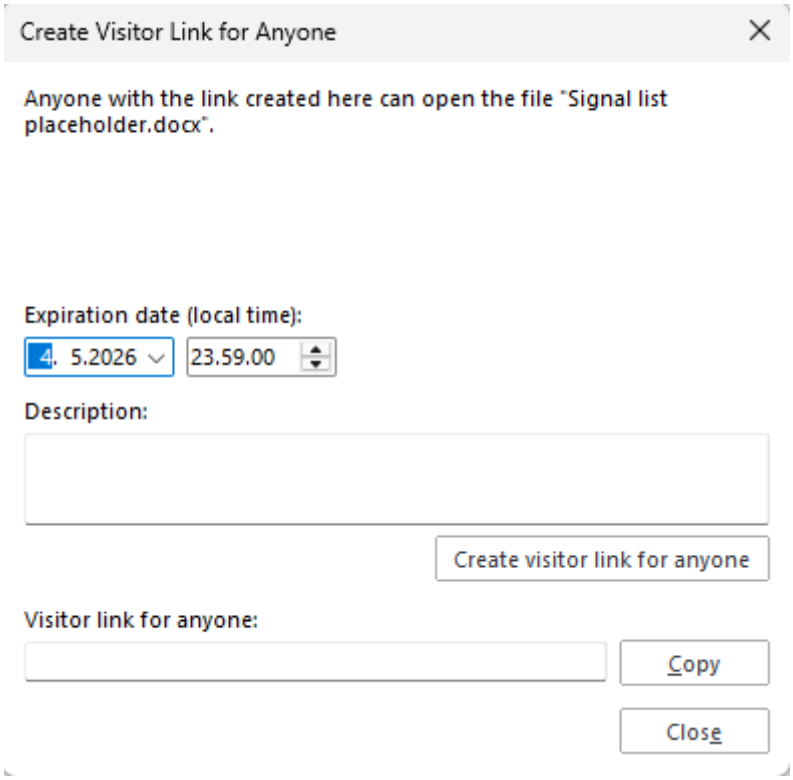

Figure 8. Visitor link created for non M-Files user.

6.5 Evaluation of concept

The section evaluates the M-Files based solution concept based on the observations made in the pilot. The purpose of the evaluation is to examine how well the key parts of the concept meet the development needs and requirements identified in the work. The evaluation is based in particular on what was demonstrated in the pilot in terms of configuration identification, findability, version control, change documentation, related material handling, template logic, views and sharing.

Based on the pilot, the key strengths of the concept can be summarized as follows:

- Configuration as an object: the configuration can be handled as a manageable object and not just a file.
- Metadata based identification: configuration identification can be supported by metadata, such as configuration type, switchgear type, configuration version and project.
- Object relationships: related material and change log can be linked directly to the configuration without having to copy documents to the same folder.
- Workflow and version control: Workflow helps distinguish between a working version, an approved version, an old revision and retired state.
- View and deliverables: Views allow objects to be grouped together for different uses without parallel folder structures.

These findings show that M-Files supports many of the needs identified in the current state analysis and requirements definition. Moving to the configuration object and metadata-based management reduces dependency on a single file name, date or folder location. At the same time, linking related material and change logs to the configuration supports information traceability and reduces person dependency, because the background information of the configuration can still be found later.

However, one of the most significant practical changes is related to the mindset. In the current way of working, users may be used to searching for information based on the folder structure, file name and the most recent file visible. In M-Files, the corresponding information is found using metadata, views and object relationships. The change offers better possibilities and higher scaling of the configuration management. It requires users to understand that views are not traditional folders, but context-based ways of presenting the same objects.

Another key observation is related to the quality of metadata. The functionality of the concept depends on users filling in metadata correctly and consistently. Therefore, production use would require a clear metadata model that defines mandatory fields,

selection lists, naming conventions and responsibilities. Without common processes, a metadata-based solution can produce a new kind of inconsistency.

Workflow and revision handling proved useful in the pilot, but they require precise definitions before wider implementation. In the pilot, some states of the workflow may be useless, depending on the needed states. Workflow distinguishes between the working version, the approved revision, the old revision status and retired status. This offers a more controlled alternative to the current archive folder thinking. At the same time, it is important to distinguish between M-Files internal version history, the workflow status and the technical configuration version. If these concepts are not clearly explained, it may be unclear to the user which version is technically correct, which is just an internal change version of the system and which is an officially published version.

Documenting changes using a change log improved the traceability of the concept, but this approach also requires practical guidance. The change log is only useful if the content, cause, effect and related configuration version of the change are systematically documented. The pilot showed that metadata can also be imported into the content of the change log document, which can reduce manual writing and improve consistency. This does still not eliminate the need to define what the user should record themselves.

In the deliverables and sharing functionalities, the pilot showed both possibilities and limitations. The deliverables view showed that deliverable documents can be collected in one view without manually copying them in the same folder. For external sharing, the visitor link was suitable for sharing of a single document with a user who does not have M-Files access rights. However, external sharing of multiple documents requires a separate delivery package, such as ZIP file and attached deliverables index. Index is important when sharing externally, so that the properties in the files will follow the files in the index. A clear deliverables process should be included in the concept in the further development. The process should define how the documents to be delivered are marked,

how the deliverables view is created, how the files are exported from the system and how the metadata-based context is preserved with the delivery.

Implementing the concept would require supporting work instructions and processes. The following things should be defined before production use: how to name the configuration, what metadata is given to it, how to link related material, how to fill in the change log, how to use workflow states, how to create a new revision and how to share objects. Without such instructions, the solution may be technically functional, but its use may remain inconsistent between different projects and users.

Overall, the pilot showed that the M-Files based solution concept is suitable as a basis for further development. The concept responds well to the problems identified in the current state, such as fragmented information management, uncertainty in identifying the correct version, finding related material, poor visibility of changes and personal dependency.

7 Conclusion

This chapter presents the conclusions of the thesis. First, the research questions are answered, after which recommendations are presented to the case company for further development of customer-specific configuration management.

7.1 Answers to research questions

1. What methods, systems and practices are currently used for managing customer-specific configurations?

Based on the current state analysis, customer-specific configuration management is based on combination of multiple tools and project-specific practices. The configurations are created using a separate configuration or design tools, and then stored and shared primarily via SharePoint, Teams, local folders or project-specific distribution channels. Practices vary depending on projects, users and customer needs. The key finding is that the current approach works in daily work, but it does not form a fully consistent, traceable or scaling management model.

2. How well do the existing systems support the needs of different stakeholders throughout the data lifecycle?

Current systems support most of the needs of the stakeholders, but the support is not consistent throughout the entire information lifecycle. SharePoint and other current tools enable file storage, sharing and collaboration but they alone do not solve the needs related to configuration identification, version control, change visibility and related material relationships to configurations. From the lifecycle perspective, the biggest challenges are that the use of configurations does not end at the time of its storing. The same configuration information may be needed in later in different phases of project for maintenance, change work or in documentation delivered to customer. If the history of the configuration, changes, related documents and usage context are not clearly preserved, the subsequent use of the information becomes difficult.

3. Is it feasible to manage configurations through a unified system?

Based on the thesis, managing configurations through unified system is a possible and justified goal, but it does not mean that all configuration related work would take place in one system. The technical creation of the configurations can still take place in separate configuration tools but the managing of the configurations after creation can be done in one system.

4. What internal and external solution options are available for improving the configuration data management?

The thesis examined both internal and external options for developing configuration management. Of the internal options, SharePoint represents a solution that is well suited to the current operating environment and familiar to users. Its strengths include accessibility, compatibility with existing tools and a low threshold for implementation of several improvements. With SharePoint, current practices should be developed with more uniform structures, common instructions, template management and versioning practices. M-Files offered a great alternative for the configuration data management. It presents a stronger foundation based on metadata and context-based management for the issues found in the thesis.

7.2 Recommendations for the company

First recommendation is to develop a clear common work instructions for configuration management. Based on the current state analysis, some of the key challenges are related not only to the systems used, but also to variation in operating practices. Therefore, the case company should define a unified process for naming configurations, storing them, identifying versions, documenting changes, linking related materials and sharing practices. If the case company continues with the current solution, these practices should still be standardized, and the existing functionalities should be used more consistently to improve scalability.

The second recommendation is to define a common minimum model for configuration identification. Even without a new system, configuration management would benefit from having key information, such as project, product, configuration type, version, date, change reason and related material defined in a consistent way. This could be supported through naming conventions and a simple change log practice.

The third recommendation is to improve version control and change documentation. The current archive folder and file name-based practices could be improved by clearly defining what is considered the latest approved version, how old versions are marked, where change information is stored and how. A simple and consistent change log practice could already improve change visibility and reduce person dependency in the short term.

The fourth recommendation is to standardize project templates and configuration templates. Each project should have a clear starting structure for configurations. This can be developed in the current environment and would reduce variation between projects.

If the case company decides to consider M-Files further, it should be approached as a limited follow up. The follow up pilot should focus on testing the concept with a real project and evaluating the required costs, ownership, support, user training and process changes. Attention should be paid to the shift from thinking model which is based on folder structures to a metadata-based working.

Overall, the recommended approach is gradual. The case company should first standardize the current configuration management practices, work instructions, naming, versioning, change documentation and templates. After that, it is more reasonable to evaluate whether a metadata-based system such as M-Files would provide enough additional value compared to the cost and organization effort required for implementation. This would also reduce the risk of transferring inconsistent practices directly into a new system.

References

Conradi, R., & Westfechtel, B. (1998). Version models for software configuration management. *ACM Computing Surveys*, *30*(2), 232–282. https://doi.org/10.1145/280277.280280

Coronado M, A. E., Sarhadi, M., & Millar, C. (2002). Defining a framework for information systems requirements for agile manufacturing. *International Journal of Production Economics*, *75*(1–2), 57–68. https://doi.org/10.1016/S0925-5273(01)00181-5

DocuWare. (n.d.-a). *Document Management Software | DocuWare Solutions*. Retrieved April 4, 2026, from https://start.docuware.com/document-management-software

DocuWare. (n.d.-b). *Document Workflow Management System & Software | DocuWare*. Retrieved April 5, 2026, from https://start.docuware.com/workflow-management

DocuWare. (n.d.-c). *The Ultimate Guide to Document Version Control*. Retrieved April 5, 2026, from https://start.docuware.com/blog/document-management/what-is-version-control-why-is-it-important

DocuWare. (n.d.-d). *What is Document Management (DMS)? | DocuWare*. Retrieved April 5, 2026, from https://start.docuware.com/document-management

Kim, M., & Compton, P. (2004). Evolutionary document management and retrieval for specialized domains on the web. *International Journal of Human-Computer Studies*, *60*(2), 201–241. https://doi.org/10.1016/j.ijhcs.2003.10.004

Liu, S., McMahon, C. A., & Culley, S. J. (2008). A review of structured document retrieval (SDR) technology to improve information access performance in engineering document management. *Computers in Industry*, *59*(1), 3–16. https://doi.org/10.1016/j.compind.2007.08.001

Lowe, A., McMahon, C., & Culley, S. (2004). Characterising the requirements of engineering information systems. *International Journal of Information Management*, *24*(5), 401–422. https://doi.org/10.1016/j.ijinfomgt.2004.06.008

M-Files. (n.d.-a). *AI & Metadata-Driven Document Management Platform | M-Files*. Retrieved April 4, 2026, from https://www.m-files.com/m-files-platform/

M-Files. (n.d.-b). *Configuring workflows*. Retrieved April 14, 2026, from https://userguide.m-files.com/user-guide/latest/eng/Workflows.html

M-Files. (n.d.-c). *Editing the vault metadata structure*. Retrieved April 13, 2026, from https://userguide.m-files.com/user-guide/latest/eng/Metadata.html

M-Files. (n.d.-d). *File Management System for Metadata-Driven Control | M-Files*. Retrieved April 5, 2026, from https://www.m-files.com/supplemental/file-management/

M-Files. (n.d.-e). *New class*. Retrieved April 13, 2026, from https://userguide.m-files.com/user-guide/latest/eng/New_class.html#new_class

M-Files. (n.d.-f). *Object metadata*. Retrieved April 13, 2026, from https://userguide.m-files.com/user-guide/latest/eng/object_metadata.html#object_properties

M-Files. (n.d.-g). *Object permissions*. Retrieved April 13, 2026, from https://userguide.m-files.com/user-guide/latest/eng/object_permissions.html

M-Files. (n.d.-h). *Object relationships*. Retrieved April 14, 2026, from https://userguide.m-files.com/user-guide/latest/eng/object_relationships.html

M-Files. (n.d.-i). *Permissions and Automatic Permissions*. Retrieved April 13, 2026, from https://userguide.m-files.com/user-guide/latest/eng/Permissions_and_Default_permissions.html#permissions_and_automatic_permissions

M-Files. (n.d.-j). *Searching*. Retrieved April 14, 2026, from https://userguide.m-files.com/user-guide/latest/eng/searching.html

M-Files. (n.d.-k). *Using document templates*. Retrieved April 13, 2026, from https://userguide.m-files.com/user-guide/latest/eng/using_template.html

M-Files. (n.d.-l). *Version history*. Retrieved April 5, 2026, from https://userguide.m-files.com/user-guide/latest/eng/object_history.html

M-Files. (n.d.-m). *Viewing object activity*. Retrieved April 14, 2026, from https://empower.m-files.com/content/web-object-activity

M-Files. (n.d.-n). *Workflows*. Retrieved April 5, 2026, from https://userguide.m-files.com/user-guide/latest/eng/using_workflows.html#workflow

Microsoft. (n.d.-a). *Add a content type to a list or library.* Microsoft Support. Retrieved April 13, 2026, from https://support.microsoft.com/en-us/office/add-a-content-type-to-a-list-or-library-917366ae-f7a2-47ad-87a5-9689a1884e60

Microsoft. (n.d.-b). *Create a document library in SharePoint.* Microsoft Support. Retrieved April 4, 2026, from https://support.microsoft.com/en-us/office/create-a-document-library-in-sharepoint-306728fe-0325-4b28-b60d-f902e1d75939

Microsoft. (n.d.-c). *Create a flow for a list or library.* Microsoft Support. Retrieved April 14, 2026, from https://support.microsoft.com/en-us/office/create-a-flow-for-a-list-or-library-a9c3e03b-0654-46af-a254-20252e580d01

Microsoft. (n.d.-d). *Customize permissions for a SharePoint list or library.* Microsoft Support. Retrieved April 13, 2026, from https://support.microsoft.com/en-us/office/customize-permissions-for-a-sharepoint-list-or-library-02d770f3-59eb-4910-a608-5f84cc297782

Microsoft. (n.d.-e). *Enable and configure versioning for a list or library.* Microsoft Support. Retrieved April 14, 2026, from https://support.microsoft.com/en-au/office/enable-and-configure-versioning-for-a-list-or-library-1555d642-23ee-446a-990a-bcab618c7a37

Microsoft. (n.d.-f). *External or guest sharing in OneDrive, SharePoint, and Lists.* Microsoft Support. Retrieved April 13, 2026, from https://support.microsoft.com/en-gb/office/external-or-guest-sharing-in-onedrive-sharepoint-and-lists-7aa070b8-d094-4921-9dd9-86392f2a79e7

Microsoft. (n.d.-g). *Get started with SharePoint.* Microsoft Support. Retrieved April 4, 2026, from https://support.microsoft.com/en-us/office/get-started-with-sharepoint-909ec2f0-05c8-4e92-8ad3-3f8b0b6cf261

Microsoft. (n.d.-h). *How versioning works in lists and libraries.* Microsoft Support. Retrieved April 4, 2026, from https://support.microsoft.com/en-gb/office/how-versioning-works-in-lists-and-libraries-0f6cd105-974f-44a4-aadb-43ac5bdfd247

Microsoft. (n.d.-i). *Introduction to managed metadata*. Microsoft Learn. Retrieved April 4, 2026, from https://learn.microsoft.com/en-us/sharepoint/managed-metadata

Microsoft. (n.d.-j). *Move or copy files in SharePoint.* Microsoft Support. Retrieved April 16, 2026, from https://support.microsoft.com/en-au/office/move-or-copy-files-in-sharepoint-00e2f483-4df3-46be-a861-1f5f0c1a87bc#id0ebd=microsoft_365

Microsoft. (n.d.-k). *Require approval of items in a list or library.* Microsoft Support. Retrieved April 14, 2026, from https://support.microsoft.com/en-gb/office/require-approval-of-items-in-a-list-or-library-cd0761c4-8c3f-4ea2-9435-13c28aa23d08

Microsoft. (n.d.-l). *Top questions about check out, check in, and versions.* Microsoft Support. Retrieved April 4, 2026, from https://support.microsoft.com/en-gb/office/top-questions-about-check-out-check-in-and-versions-7e941339-e972-4c7a-a79a-80a1fcf84076

Microsoft. (n.d.-m). *View the version history of an item or file in a list or library.* Microsoft Support. Retrieved April 14, 2026, from https://support.microsoft.com/en-us/office/view-the-version-history-of-an-item-or-file-in-a-list-or-library-53262060-5092-424d-a50b-c798b0ec32b1

Microsoft. (n.d.-n). *What is a document library?* Microsoft Support. Retrieved April 4, 2026, from https://support.microsoft.com/en-us/office/what-is-a-document-library-3b5976dd-65cf-4c9e-bf5a-713c10ca2872

Müller, P. (2013). Configuration Management – A Core Competence for Successful through-life Systems Engineering and Engineering Services. *Procedia CIRP*, *11*, 187–192. https://doi.org/10.1016/j.procir.2013.07.032

Pulicharla, M. R. (2024). Data Versioning and Its Impact on Machine Learning Models. *Journal of Science & Technology*, *5*(1), 22–37. https://doi.org/10.55662/JST.2024.5101

Ratchev, S., Urwin, E., Muller, D., Pawar, K. S., & Moulek, I. (2003). Knowledge based requirement engineering for one-of-a-kind complex systems. *Knowledge-Based Systems*, *16*(1), 1–5. https://doi.org/10.1016/S0950-7051(02)00027-8

Saiedian, H., & Dale, R. (2000). Requirements engineering: Making the connection between the software developer and customer. *Information and Software Technology*, *42*(6), 419–428. https://doi.org/10.1016/S0950-5849(99)00101-9

Van Der Aa, H., Leopold, H., Mannhardt, F., & Reijers, H. A. (2015). On the Fragmentation of Process Information: Challenges, Solutions, and Outlook. In K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, & Q. Ma (Eds.), *Enterprise, Business-Process and Information Systems Modeling* (Vol. 214, pp. 3–18). Springer International Publishing. https://doi.org/10.1007/978-3-319-19237-6_1

Whyte, J., Stasis, A., & Lindkvist, C. (2016). Managing change in the delivery of complex projects: Configuration management, asset information and 'big data.' *International Journal of Project Management*, *34*(2), 339–351. https://doi.org/10.1016/j.ijproman.2015.02.006

Zantout, H., & Marir, F. (1999). Document management systems from current capabilities towards intelligent information retrieval: An overview. *International Journal of Information Management*, *19*(6), 471–484. https://doi.org/10.1016/S0268-4012(99)00043-2

Zhang, W., & Yang, Z. (Eds.). (2025). *Methods and Applications of Data Management and Analytics*. MDPI. https://doi.org/10.3390/books978-3-7258-3205-7