# Enhancing traceability in heterogeneous design platforms

Roland Stolt* and Fredrik Elgh

Department of Industrial Product Development, Production and Design,
School of Engineering,
Jonkoping University,
Gjuterigatan 5, 551 11 Jonkoping, Sweden
Email: roland.stolt@ju.se
Email: fredrik.elgh@ju.se
*Corresponding author

**Abstract:** Companies are frequently using in-house developed scripts, macros and applications to support and automate the development of products and product variants. These items are often of different types creating a design platform with a multitude of dependencies. Such environments are often difficult to overview and maintain. In the current paper, an aerospace company with a design platform that has proven challenging to maintain has been studied. A method and limited software implementation have been tested and evaluated by interviewing four professionals at the company. The implementation provides a high granularity, which has been found to be necessary to give a quick platform overview and to facilitate maintenance and expansion of the design platform.

**Keywords:** design platform; dependencies; traceability; maintenance; product data management; PDM.

**Reference** to this paper should be made as follows: Stolt, R. and Elgh, F. (2019) 'Enhancing traceability in heterogeneous design platforms', *Int. J. Product Lifecycle Management*, Vol. 12, No. 1, pp.62–80.

**Biographical notes:** Roland Stolt holds a Master of Science in Mechanical Engineering at Lund University and a PhD in Product and Process Development at Chalmers University of Technology. His research interests involve design for manufacturing (DFM) and design automation (DA). Recent research has been directed towards design for additive manufacturing (DFAM) and in particular dies and aerospace parts manufactured by selective laser melting (SLM). This has resulted in around 40 scientific publications in journals and conferences. He has since 2013 been the head of an international master programme named 'product development and materials engineering' at Jönköping University.

Fredrik Elgh is a Full Professor of Product Development at the Department of Industrial Product Development, Production and Design at Jönköping University. He holds an MSc in Mechanical Engineering and a PhD in Product and Production Development from Chalmers University of Technology. He has managed and worked in 12 externally funded research projects executed in close collaboration with more than 35 companies and published more than 95 papers in scientific journals and conference proceedings. Earlier works are oriented towards cost engineering, design for producibility, design automation and knowledge-based engineering, while later work also includes design for additive manufacturing, design methodology, set-based concurrent engineering,

product platforms, design platforms, knowledge engineering and lean management with applications in automotive, aerospace, mechanical, consumer and house-building industry.

## 1 Introduction

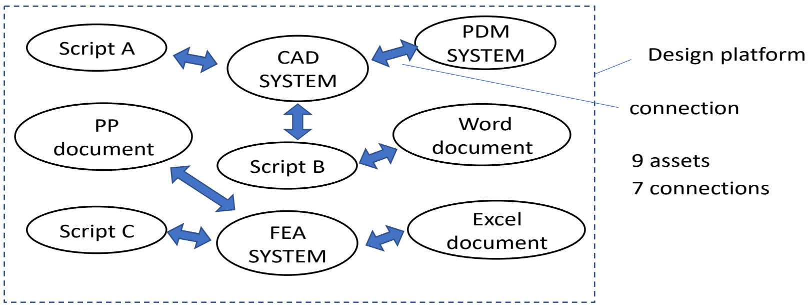
In engineering design, a variety of tools are used to promote the reuse of engineering knowledge. The motivation is making the creation of new products or product variants quicker and to enhance product and process quality. Examples of tools may include using template CAD models containing relevant product knowledge (Tiwari et al., 2015) which are adapted to the design requirements at hand. It can also involve integrating tools for design synthesis and design analysis with information and knowledge sources into an environment for automating the engineering process (Elgh and Cederfeldt, 2008; Johansson, 2011). Such systems are generally referred to as design automation (DA) systems. If the system is built on a product structure approach using adaptable CAD-models, it is termed a knowledge-based engineering (KBE) system. The DA system is more oriented towards the automation of a design process whereas KBE is more about increasing the productivity and quality of interactive CAD work. Examples of research on KBE systems are plentiful. Some examples are found in Chapman and Pinfold (2001), Quintana-Amate et al. (2017), Salchner et al. (2016), Sandberg et al. (2017) and Trehan et al. (2015).

The KBE and DA systems are often heterogenous meaning that they consist of many different subsystems of several types. In literature, such heterogeneous systems are sometimes referred to as ‘platforms’ meaning a collection of different objects such as scripts, information sources, parametric CAD and FEM models possibly with scripts and macros for automation. These are connected to perform engineering tasks, automatically or with a degree of user interaction. However, the term platform is a very commonly used and can mean anything form software platforms to oilrigs. To describe the platform in this paper the word *design platform* is most accurate. An example of a design platform is found in André and Elgh (2018). Here, the objects of the design platform are referred to as ‘engineering assets’. For simplicity, the terms *platform* meaning design platform and *asset* or *platform asset* as a denomination for the objects that are used in the platform will be used for the remainder of this paper. I must be noted that assets are not documents such as engineering drawing describing an instance of a design. Assets are what is used to create new designs or design variants. The meaning of platform and asset will be further discussed in the paper.

The assets of the platform can be connected in various ways. Figure 1 shows a simple example of a platform with nine interconnected assets by seven connections. These connections are created because there is an information exchange between the assets. It can for example occur when one asset calls another for requesting data or getting a calculation performed and retrieving the results.

The platform and its assets are used to assist the development of technology and products. Assets are everything needed to conduct the development. Thus, items such as ready product documentation such engineering drawings are not assets in the platform. They are merely instances and a result of the processes performed in the platform.

**Figure 1** Examples interconnected assets in a platform (see online version for colours)



An example of an asset is the PDM system. It is very good at version control of documents, workflows and keeping track of the connectivity between documents. Thereby, it can facilitate the development of products, but it cannot drive the development process forward by itself. Other assets, as for instance scripts and software for making engineering calculations, are needed.

The problem addressed in this paper is visualising and keeping the dependencies between assets in any heterogeneous platform up to date as the platform is changed over time. The paper explores using a 'dependency manager' (DM) application that allows the definition and tracing of dependencies as well as visualising them using graph theory. The DM has been tested in an industrial environment at an aerospace company. The company has over time developed a highly heterogeneous platform, making it an ideal test environment for DM. The DM is evaluated via an interview with four involved staff at the company. The interview questions are aimed at finding out how DM should be implemented in the organisation in full scale, how the information should be captured and stored and how and finally, what information about the dependencies (meta-data) that should be captured and managed. The continuation of this paper is organised as follows. After the literature Section 2, a description of the research methods is found in Section 3. In Section 4, an elaboration of the platform concept is made. Section 5 describes the implementation at the aerospace company. This is followed by discussions in Section 6 and finally conclusions and future work in Section 7.

## 2 Literature

As stated in Johannesson et al. (2017), there is a misconception that product platforms only consist of a set of components that are combined into products to increase economic viability. The authors elaborate on product platforms and present a methodology for platform preparation and platform execution. Functional modelling is used in the platform preparation to find an efficient modularisation into configurable components that supports the platform development.

The review article (Zhang, 2015) contains an interesting elaboration on different types of platforms. It shows that the number of papers published on platforms is massive. Thus, the paper categorises platforms into different types such as process platforms, layout platforms, flexible platforms, function–technology platforms, and multi-branded platforms. It is concluded that many types of platforms are in use and are applied at all life-cycle stages.

A platform can incorporate the whole design process or part of it. Platforms are often highly specialised towards their tasks, making it difficult to come up with any general guidelines for designing them in companies. However, for retrieving and systematising the product and process knowledge that will form the base of the platform there are well established methods such as MOKA and PVM (Hvam, 2008; Stokes, 2001) directed towards systematising knowledge to build KBE systems and product configurators. There are few recommendations on how to integrate heterogeneous subsystems so that they can be managed in a coherent manner. However, some examples can be found such as in Kristianto et al. (2015) who take naval architecture as an example. However, ship building projects are quite unique. Therefore, building a product platform for supporting the process seems hard to motivate, at least with the traditional component view on product platforms. Still, authors propose a platform from which ships can be configured on a system level allowing the identification of areas that must be addressed in the project. It is a converging process starting from configuring system parameters down to the configurable components. The dependencies are kept track of using a domain mapping matrix (DMM) to get the system level dependencies. On the subsystem level, the dependency structure matrix (DSM) is used. This leads to that dependencies are made explicit on different levels of granularity. A similar approach is taken in André and Elgh (2018). Here, coupled matrices are used to visualise and manage dependencies.

The updating of the platform assets creates a need for updating the dependencies between its assets. With proper tracing it can be made explicit which platform asset that was included when applying it to a design or when tracing how the platform assets depend on each other. Keeping a record of which versions of the platform assets that were used at a particular run when a product was designed, makes it possible to trace sources of the error which perhaps only are discovered later in the product lifecycle. For this purpose, PDM for tracing the relations between the platform assets is discussed in Cho et al. (2016). The authors highlight the challenges in using PDM. One of the main challenges is that the platforms have not been constructed for PDM in the first place. They have an ad hoc structure that has been built up over time. This is especially true for keeping track of dependencies between parts of documents, i.e., applying a higher degree of granularity than merely between complete documents as in PDM. The possibility to distinguish between different degrees of granularity has been identified as important when tracing dependencies (Hjertberg et al., 2018). In the publication, granularity refers to the size of the unit of information that is interconnected. However, there does not seem to be a generally accepted definition of granularity. In Algeddawy and Elmaraghy (2013), granularity level refers to the structure tree of modules and components.

There is functionality in PLM tools for integrating relational and document information. This is useful, but not always achievable for small and medium sized enterprises (David and Rowe, 2016). In the same publication, the requirements for partial PLM are listed and applied in automotive and clothing companies in the NPD process.

A suggestion presented in Poorkiany et al. (2016a) is to use the design rational (DR) to keep track of sub document level relations in platforms. The DR contains the information about HOW to design a product i.e. the rationale behind the design decisions taken. Traditional product documentation like engineering drawings only conveys the result of the design process. The DR involves one or several of the platform assets used in a way that is explained by the DR. The DR explain which versions of the platform assets that should be used to solve a particular task and how. It can also support a higher granularity level if it explains which part of the platform asset that should be used. By documenting the DR and its legacy (Elgh, 2014; Poorkiany et al. (2016b), it is possible to keep track of the platform and its assets, provided that the DR is elaborate enough.

In Monticolo et al. (2015), a knowledge configuration model (KCModel) is proposed allowing traceability of design parameters. This enables a very fine granularity and makes it possible to understand and re-use the parameters.

Granularity in the context of PDM is discussed showing that there are several problems with the workflows in PDM. If one document is locked for revision by a user, then all documents with relations to that documents also needs to be locked trough anticipating that there also to the related documents. This is conservative and may slow down the work (Chan and Yu, 2007).

Proper structuring of the platform assets is important to obtain traceability, transparency and maintainability. It is suggested that the platform assets be formalised into 'knowledge objects' (KO) (Elgh and Johansson, 2014). A KO is a self-contained functional asset capable of performing a task independently. Solving the task may involve obtaining data and information from platform assets. The KO has one or several input parameters. When the input parameters are known, the knowledge object can perform its designated task. The result is a set of output parameters which, in turn, can be used as input in other knowledge objects (KO:s). These KO:s can be managed by a specially purpose-built software called 'Howtomation' (Johansson, 2015). A graphical interface is used to connect the different KO:s with each other and data/information sources giving complete control of the platform connectivity. It uses an inferencing engine to manage the execution and data flow between different KO:s. The KO:s can be revised, added or removed without having to consider the flow of execution between them.

Using Howtomation, a platform that has an explicitly mapped connectivity is defined. The way the system processes the problem is very transparent. A record of which version of each KO that was used throughout each product development project can be kept. Depending on how the KO:s are defined, a varying level of granularity can be obtained. For a high granularity, the scope of each KO should be kept as limited as possible containing the least functional unit. Howtomation has been used to manage knowledge objects in several different types of engineering applications. An example from the aerospace industry is given in Pabolu et al. (2016).

To conclude from the literature study, the need of connecting different platform assets into a complete system occurs in several different situations. The system may be a KBE or DA, a product configurator, or some other type of integration of several subsystems that perhaps are managed by PLM. In the literature, the recommendation is to prepare the platform assets to be integrated in the IT infrastructure. Still, there are subsystems that have not be designed for this integration and still needs to be managed, a situation that is little discussed in literature. In such case the DM tool presented will make it possible to get an overview and to visualise the dependencies making version control and system maintenance possible.

## 3 Research approach

This paper has been written as part of a 3-year research project. In the project, the design research methodology (DRM) (Blessing, 2009) has been used. It is characterised by descriptive and prescriptive phases. Though interviews conducted at the companies reported in Stolt et al. (2016), enablers and success criteria to direct the research project were found. It indicated that reuse of knowledge was the most important driver for the companies to engage in the research project. At the aerospace company, traceability was found to be a major enabler for reuse of knowledge, why a design manager (DM) software was development and reported in Hjertberg et al. (2018).

In the current paper traceability related to commonly used document such as Microsoft word, PowerPoint and Excel are examined. The DM is tested at the aerospace company's platform to give an estimate how DM can be implemented.

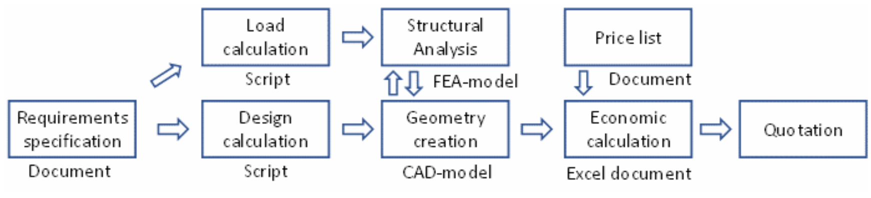
## 4 Platforms

Platforms evolve as the underlying knowledge and processes of product development which they automate are advanced. Therefore, it becomes a challenge to maintain them over time. An example is when an asset is superseded by an improved version. Likely, the asset has other assets connected to it. For efficient maintenance and continuous platform extension, it is very useful to know how the assets connect to each other so that it can be checked that it responds in the same way as the superseded asset.

### *4.1 Platform architecture*

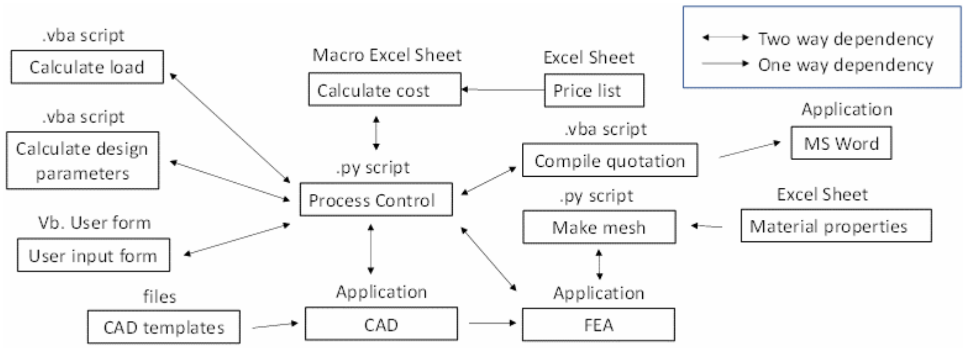
In Figures 2 and 3, an imagined platform is described. Figure 2 shows the process view of it. The process view shows the order in which the different tasks are performed. Figure 3 shows the platform view, i.e., which assets that are involved and how the information exchange between them is made. The imagined platform is included to exemplify a platform and show how dependencies in it are formed. As seen in Figure 2, the process begins by reading the requirement specification from a document. It can for example be created by letting a potential customer fill out a web form or it may have been defined via personal contacts in sales. The platform then uses a script that apply handbook formulae to make a preliminary calculation of the design parameters (design calculation) Simultaneously, the expected loads on the design is calculated based on the requirement specification (load calculation). The design parameters are then used to create a CAD-model of the preliminary design. The loads are applied, and it is analysed using FEA to assess if it can carry the specified loads. If not, the CAD-model is automatically revised and re-calculated. Thereafter, it is passed on to structural analysis to determine if the proposed design can withstand the calculated loads. Finally, it is sent to an excel sheet which determines the cost of the product. Prices of materials, machine time, labour and so on are retrieved from a price list. A quotation for the design as well as a CAD-model describing its geometry can be provided to a potential customer.

**Figure 2** Flow between tasks in an engineering task (process view) (see online version for colours)



To realise the process described in Figure 2 there are a number of assets performing each of the sub-tasks. This is shown in Figure 3. The platform must have process control which initiate the process, controls the parameter flow and initiate the presentation of the results. For this an asset 'process control' is included. In Figure 3, the dependencies between the different assets are shown. It also shows what type of asset it is. Assets are scripts of various types (Python and Visual Basic) and also applications such as Excel sheets, CAD and FEA.

**Figure 3** Interconnected assets in an example platform (system view) (see online version for colours)



The example shows that the assets have one set of connections for solving a specific engineering task. It must be noted that the assets of the platform also can be used individually or in a different order to perform other engineering tasks. In such case the connections will be different. Thus, the set of connections is valid for a performing a certain engineering task.

## *4.2 Connectivity granularity*

In Figure 3 example, the assets are connected by eight double and five one way directed dependencies. It is only the connectivity between complete assets that are shown. Connections can be described in a finer granularity, by subdividing the assets into sections. By a section is meant a part of the asset such as some lines of code in a script. One way of designating a section in an asset is to use 'tags'. A tag is a text marker, pointing out the section. Figure 4 shows a tag in a Python script. The row '#DM_Tag_ID: 18' identifies the starting point of the tag and the row '#DM_TagExit' marks the end of the tag.

**Figure 4** Example of tagged section in a Python script (see online version for colours)

```
 #DM_Tag_ID: 18
   #Entity name: btnBrowse_click
#Type of system entity: Python function
   #Makes use of: 10;; is used to instantiate tag class from
loaded xml file
   #Makes use of: 20;; is used to provide file browsing
functionality for the user
   #DM_TagExit
```

The use of tags enables dependencies to be defined between parts of assets giving an increased granularity. By using special software, the tags in the assets can be found, systemised and kept track of.

This was shown in Hjertberg et al. (2018). Tags were introduced in the assets. These tags were intended to be added automatically or manually as the platform assets are developed. The tags allow a graphical overview of the connectivity at different granularity levels to be generated. The tags were kept track of in an XML file which also included meta-data for the tags which made it possible to understand the role of the section and to follow the change history of the tagged section. The platform described above connects a number of assets. However, the same assets could also be used in other platforms or they could be connected in other ways to perform different tasks. This will result in a complex set of connections. It becomes necessary to establish a systematic way of keeping track of the connectivity and to manage the versions of all assets.

## 4.3 Platform maintenance

As the platform is developed over time, a need for continuous maintenance of these dependencies arises. Assets are updated or superseded and then the effect on the dependent assets must be predicted. It is not always the whole asset that is replaced. Changes may be limited to only a few lines of code in the asset designated by a tag. If so, it must be traced in which other assets these lines are referenced to keep all assets up to date. The connections between parts of assets represent a finer granularity level than document to document, i.e., granularity refers to what level of detail that the connection is defined. Common product data management (PDM) systems do not suffice since they are foremost intended for keeping track of dependencies on a document to document level.

A first and useful step to understand the connectivity in the platform is to generate a graphical map of the connections. Here, graph theory can be used as a means of handling the complex dependencies between the subsystems. An example is found in Johansson (2017).

## 4.4 Tagging in documents

In the current paper, the DM is extended to include tags in textual documents such as Microsoft Word, PowerPoint and Excel. This allows the dependencies between files written in Word or PowerPoint and sources of data to be traced to sections of code in scripts.

To do the tagging, working time must be put in defining the tags as the documents are being created. Therefore, a justification for the organisation to spend time on documenting the dependencies is needed. The developer will also need to provide information about the dependency.

## 5 The DM system

As mentioned, the DM system works by manually or automatically inserting tags in documents and scripts. In scripts, the tag is simply inserted as a comment in the code and is identified by a reserved key word string such as '#Tag_ID001' in the preamble of the comment. In this paper, the DM has been further developed to allow documents like MS Word, PowerPoint and Excel to be tagged. After defining the tags, the dependencies between them can be traced in the DM application allowing them to be stored in an XML-file.

In addition to letting the user point out sections by tagging, DM also has the possibility of automatically tracing the dependencies by the function calls made in the scripts. If a name of a function is stated in two assets, when it is assumed that a dependency exists between them.

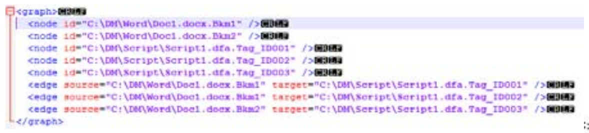
The dependencies are automatically captured and stored in a GraphML file. It is a variant of the XML file adapted for direct use in graph software such as Gephi (http://www.gephi.org). Thus, a graph of the dependencies can be generated for visualisation purposes. The GraphML file can be read and edited in DM so that new tags and dependencies can be added, deleted or revised. By regularly issuing versions of the GraphML file in the PDM system, a record of the historic development of tags and dependencies is kept.

### *5.1 Tagging in documents*

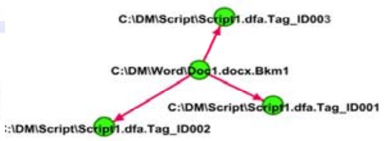
Inserting the same types of tags in documents as in scripts would reduce the readability and ruin the appearance of these documents. To overcome this problem, the bookmark function of MS Word has been used. This allows the user to put bookmarks on sections of the text. The comment contains only the tag ID. Once the Word document is bookmarked, DM can read the bookmarks which corresponds to tag ID:s and let the user define the dependencies and then commit them to the GraphML file. PowerPoint does not have a bookmark function. Instead, comments (from the review menu) containing the tag IDs can be inserted and be used in the same way as bookmarks. Also, MS Excel has the possibility to assign comments to individual cells that can be used for tagging purposes.

Figure 5 shows a simple GraphML file on the left side. It contains MS Word bookmark tags and script tags represented as nodes in lines 3–6. Lines 7–10 define the dependencies between them. The bookmark (Bkm1) of the Word file 'Doc1' has dependencies to Tag_ID001, Tag_ID002 and Tag_ID003 of the script file. On the right side of Figure 5, the corresponding graph is shown.

**Figure 5** The GraphML file and the resulting graph (see online version for colours)



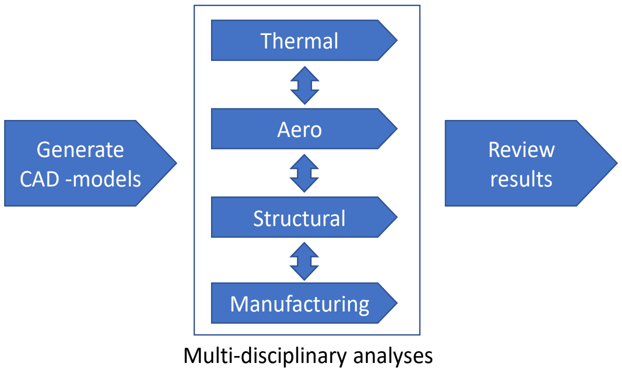
```
<graph>
  <node id="C:\DM\Word\Doc1.docx.Bkm1" />
  <node id="C:\DM\Word\Doc1.docx.Bkm2" />
  <node id="C:\DM\Script\Script1.dfa.Tag_ID001" />
  <node id="C:\DM\Script\Script1.dfa.Tag_ID002" />
  <node id="C:\DM\Script\Script1.dfa.Tag_ID003" />
  <edge source="C:\DM\Word\Doc1.docx.Bkm1" target="C:\DM\Script\Script1.dfa.Tag_ID001" />
  <edge source="C:\DM\Word\Doc1.docx.Bkm1" target="C:\DM\Script\Script1.dfa.Tag_ID002" />
  <edge source="C:\DM\Word\Doc1.docx.Bkm2" target="C:\DM\Script\Script1.dfa.Tag_ID003" />
</graph>
```


C:\DM\Script\Script1.dfa.Tag_ID003
C:\DM\Word\Doc1.docx.Bkm1
C:\DM\Script\Script1.dfa.Tag_ID001
:\DM\Script\Script1.dfa.Tag_ID002

## 5.2 *Testing DM at the aerospace company*

The company has a platform for making extensive analyses of conceptual designs to explore the whole design space. This helps in narrowing it down before going into detailed design. After finalising a conceptual design suggestion, a parametric CAD-model of it is made. This CAD model can be varied both in dimensions and in topology by changing its parameters. By systematically trying different combinations of parameters and analysing the resulting geometry from many different aspects, the company makes design space explorations allowing them to take decisions on what setting of the design parameters that results in the best trade-off between the requirements. Thus, a limited number of design options can be progressed to further development. Figure 6 shows an overview of this process.

**Figure 6** Overview of analyses made in the platform (see online version for colours)

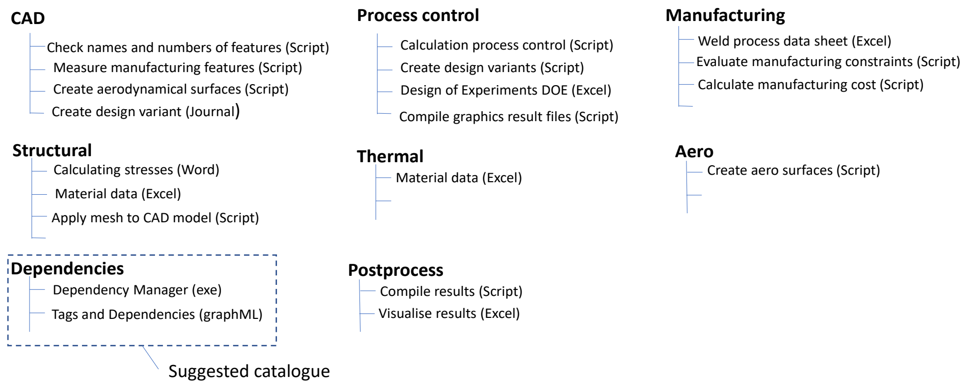


The results are gathered automatically and are presented to the engineers. The analyses are extensive involving evaluations of hundreds of different variants. To complete the analysis within an acceptable time of a few days, both the generation of the design variants and the subsequent analysis and the postprocessing of them, need to be automated. To realise this, the company has created in-house software, scripts and information sources that are used in conjunction with their commercial CAD, FEA and CFD packages. This forms a heterogeneous platform that is automated using process control software. The assets are interconnected with many dependencies between them. It is an example of system that has grown in an ad hoc way.

The company manages all product documentation in a PDM system. However, the PDM system is currently not flexible enough for the early design stages. Instead, both documents and assets that are involved in the analyses are managed by WinCVS, which is a free-ware for concurrent versioning for Windows. A vault containing the documents is placed on a server. WinCVS has version control keeping track of the versions of the assets. The whole (or part) of the database of the vault can be copied to a local client forming a catalogue structure on the hard drive of the client. In principle, the catalogue structure that is copied to the client looks as depicted in Figure 7. The catalogues shown in the figure is a limited and simplified representation of the actual catalogue structure. However, the same principle of coping the server files to a local client, defining all the parameters and running the analysis is used throughout the company.

Each catalogue has a number of files containing the scripts, data and so on. As seen in Figure 7, the names on the catalogues correspond to each discipline. As an example, everything related to aerodynamic calculations is found in catalogue 'Aero'. In the catalogue 'CAD' a script is found that checks so that the naming and numbering convention is followed in each of the created CAD-models. There are also scripts and information sources in catalogues for structural, thermal and manufacturing calculations. In the catalogue, 'process control' scripts for invoking and progressing the automated analyses are kept. When planning the DM implementation, it was suggested to the company to add a catalogue called 'dependencies' (lower left in Figure 7) to be used for storing the DM application and the graphML file. It can then be managed in WinCVS providing a versioning of these files.

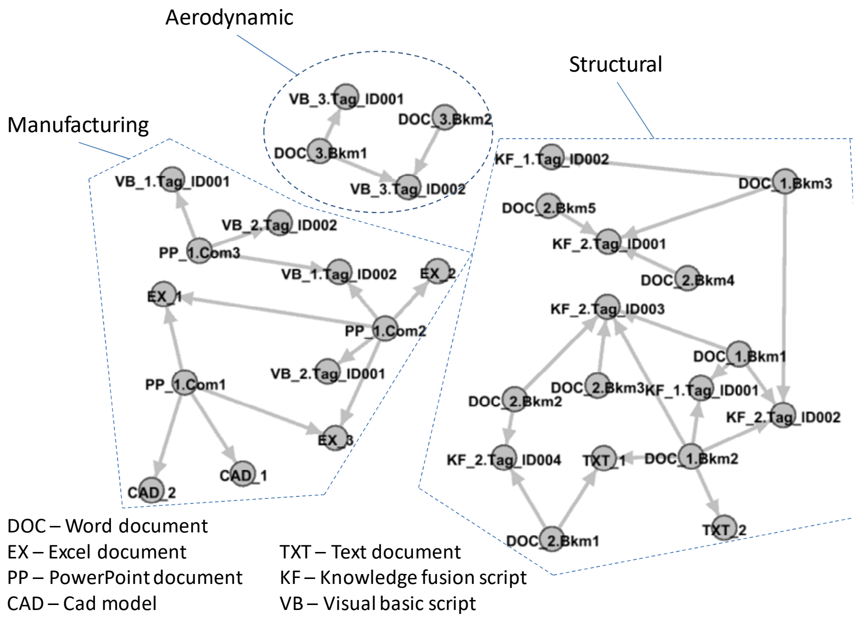
**Figure 7** Client catalogue structure (see online version for colours)


CAD
Check names and numbers of features (Script)
Measure manufacturing features (Script)
Create aerodynamical surfaces (Script)
Create design variant (Journal)
Process control
Calculation process control (Script)
Create design variants (Script)
Design of Experiments DOE (Excel)
Compile graphics result files (Script)
Manufacturing
Weld process data sheet (Excel)
Evaluate manufacturing constraints (Script)
Calculate manufacturing cost (Script)
Structural
Calculating stresses (Word)
Material data (Excel)
Apply mesh to CAD model (Script)
Thermal
Material data (Excel)
Aero
Create aero surfaces (Script)
Dependencies
Dependency Manager (exe)
Tags and Dependencies (graphML)
Postprocess
Compile results (Script)
Visualise results (Excel)
Suggested catalogue

Currently there are no tags, bookmarks or comments in the company's documents or subsystems. To test, 32 tags were added in the documents on a client computer at the company. Equally many dependencies between them were defined using DM. This involves only a small subset. Fully tagged, the catalogue structure will comprise about 1,800 files in 250 catalogues.

A GraphML file was created and a graph was generated from it using Gephi. The graph is seen in Figure 8. Which type of file it is can be seen in the graph: DOC – Word, EX – Excel, PP – PowerPoint, CAD – Cad model, TXT – Text document, KF – Knowledge Fusion script, VB – Visual Basic script. The extensions refer to bookmarks (Bkm), comments (com) and tags (Tag_ID). As seen, the dependencies involve three clusters of files belonging to aerodynamic, structural and manufacturing analyses. The arrows represent the direction of the dependencies (one or two ways). As an example, in the aerodynamic cluster, the script VB_3.Tag_ID002 uses information from the document DOC_3.Bkm1 and DOC_3.Bkm2. The script VB_3.Tag_ID001 used information from DOC_3.Bkm1.

**Figure 8** Dependency graphs generated in the company's DA system (see online version for colours)



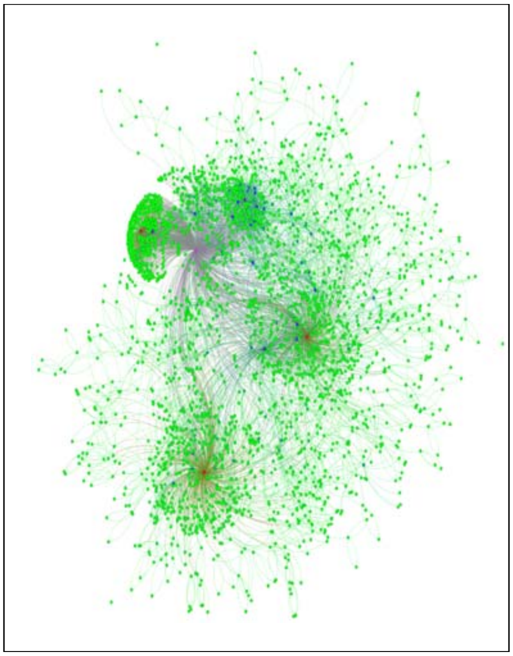
Having confirmed that the documents can be tagged and traced in the proposed way, it will be possible to introduce tags in all documents and keep the connectivity information in the graphML file.

## 5.3 Connectivity between KF scripts

Since DM has functionality for automatically tracing dependencies, a trial was made on all KF scripts found on the client server. KF is a declarative language used to write scripts to automate procedures in the CAD system Siemens NX, extensively used by the aerospace company. No tagging was introduced in the KF-scripts. They are instead traced by the calls to other scripts. Thus, when a name of a script appears in the code of another script it is assumed that there is a dependency between the two scripts. This information is stored in the GraphML file, showing the connectivity between all scripts.

This involved all the KF-scripts at the company and gave an indication of the scalability of DM. Many of the KF-scripts have been written in house for a long period of time, exceeding ten years. It also includes KF-scripts that originate from the CAD system installation, totalling about 2,800 KF-script files. The Gephi-graph of the dependencies was generated using DM and is shown in Figure 9. The dependencies between each of the scripts are depicted. As seen in Figure 9, three scripts have a very high number of dependencies (the red nodes). These represent basic functions in the CAD systems such as converting degrees to radians or converting the format of the CAD-file itself and are therefore called from other scripts often. The graph revealed that a number of scripts do not have any connections and thus appearing as isolated dots. This indicates that they are not in use and can be removed.

**Figure 9** Automatically traced dependencies between all KF scripts in the company (see online version for colours)



## 5.4 *Evaluation of DM by interview*

DM has been demonstrated to some of the staff at the aerospace company. After the demonstration, the company staff were interviewed about how they perceive the proposed system. It was in total four different professionals with long working experience in the company who were interviewed. One of the professionals is responsible for the system development, two of them are users of the system, and the fourth is a manager of the complete analysis team encompassing all engineering disciplines.

The questions they were asked belongs to three different categories seen in Table 1. The categories are implementation, file management and meta-data. The first category is meant to find out how the implementation of DM should be made in the organisation and how to motivate the system developers to keep track of the dependencies and understanding the value of this. The second category is how the GraphML file (or files) should be managed over time and the final category is finding what meta-data, i.e., what data should be kept concerning each dependency. The questions also had the purpose of finding what metadata that should be kept for each tag and dependency and how the tagging of dependencies should be stored in the WinCVS and finally how the management of the GraphML file should be carried out. Under each of the headings after the table, a summary of the answers is given.

**Table 1** Interview questions

| *Implementation* | *File management* | *Meta-data* |
|---|---|---|
| How can users be motivated to start adding tags in the documents? | If the DM and GraphML files are handled in Win-CVS, will the traceability be sufficient? | What additional metadata do you think will be needed? |
| What dependencies do you think should be kept track of? | When should new versions of the GraphML-file be issued? | Can meta-data be used to facilitate the maintenance of the system? |
| Can anything be done so that the users will remember to tag? | Should there be several XML-files like, for instance, one with project specific dependencies and one with general dependencies? | |
| | Can you propose any additional analyses of the XML-file that could be of interest? | |
| | Are there any additional groups of users with needs on special views on the XML-file? | |
| | Are additional granularity levels needed in addition to the file to file and part of file to part of file dependencies? | |

### *5.4.1 Implementation*

The answers show that the staff in the organisation needs to be motivated to add tags and dependencies gradually as they revise documents from the WinCVS. For this, the value of it must be further demonstrated. As a suggestion, the implementation of the dependency WinCVS catalogue could start in a single project first. If it turns out to be a value adding activity and sufficiently useful when maintaining or extending the platform it can be extended to a full project and perhaps finally be prescribed as a standard corporate procedure.

To make sure that the users remember to tag and add dependencies, simplicity is recognised as important. It is suggested that improved graphics in the GUI is needed. It should give an overview of the existing dependencies and allow the adding and revising of dependencies by clicking.

Among the dependencies that should be kept track of, manually added ones between data and information sources and scripts are identified as most value-adding. The likelihood that the systems at a later point need to be reconstructed (as for example for tracing the source of error in the final product) is deemed low.

### *5.4.2 Meta-data*

Categorisation of the dependency into disciplines, i.e., aero, manufacturing, structural and so on is suggested to be included as meta-data. In addition, file size, the number of rows and the type of file is of interest when analysing the dependencies. Perhaps also the role and the competence of the person, as well as which team site that the person belongs to should be kept as meta-data.

It is not expected that any additional type of meta-data can be added to facilitate the maintenance. Only getting an overview of the dependencies is expected to be enough.

### 5.4.3 File management

Handling the DM and GraphML-files as proposed in WinCVS is expected to provide sufficient traceability. However, currently unforeseen needs of traceability may emerge when DM is used. Interviewees state that more experience is needed before this question can be finally answered.

A new issue of the GraphML-file should be made as soon as new or updated versions of documents containing tags are committed to WinCVS. This will result in many versions of the GraphML-file. However, this is not expected to be problematic.

It is believed that a single GraphML-file is sufficient. From this file different graphic or other views, adapted to different types of users can be generated.

The GraphML file can additionally be analysed with regards to type of file, size and number of rows. It is expected that, apart from users and platform developers, there is a need for special views for platform owners (those who are reasonable for the function of the platform) and those who specify the system and for those who validate new methods. It must be possible to trace which discipline (viz. structural, aero, manufacturing) that the dependency belongs to. There is no need for additional granularity levels according to the interviews.

### 5.4.4 Remarks on the interview answers

It should be noted that future versions of the platform will have the dependencies between documents managed by the PDM system. That is the GraphML file (or files) will be kept in the PDM system together with all other assets. This is because the company will handle all documents including the ones from the conceptual design phase in PDM in the future. It is a principal decision and it is motivated by the fact that the PDM system is becoming more flexible and adapted to the early stages of design. This includes the product specifying documents as well as the assets.

It was expressed in the interviews that the origins of the dependencies should not be included in the meta-data. Instead, a text should the entered directly in the assets by the user explaining why the dependency exists.

Knowing what discipline a file belongs to will make it possible to check how disciplines are cross-referenced. In some cases, it is desirable to use an asset stand-alone as for example when only doing the structural analysis. In such case, it is valuable to know how the different disciplines are connected so that it can be readily predicted what measures need to be taken to run a module as stand-alone. One example of such interdisciplinary dependencies is that the structural analyses are dependent on the thermal analysis because the thermal gradients will introduce stresses in the structure. Thus, making the structural analyses requires a complete thermal analysis. This is an example of a one way directed relation. It does not go both ways since the stresses do no not have any effect on the thermal distribution.

## 6 Discussion

The problem described in the introduction of this paper is visualising and keeping the dependencies between assets in platforms up to date as the platform is changed over time. It was argued that keeping track of the dependencies on a document to documents level, something that is often done using PDM systems, does not always suffice since it does not take the need of finer granularity and the dynamic nature of platform connectivity into account. PDM has primarily been developed for keeping track of product documents, whereas the assets used for generating these product documents was not the primary in the scope of PDM. By letting the user point out sections of interest by tagging and defining the connections between the tagged sections, a finer granularity is achieved. This will allow users to quickly understand how the assets are connected when attempting to change them. When new staff is introduced to the platform, a graphical overview of the connectivity is expected to be useful. Further, assets or sections of assets that do not have any connections can quickly be identified. This indicates that they are not in use and can therefore be removed so that they do not clutter up the platform. It must be observed that the connectivity is valid for solving a particular task in a certain way using the platform. The assets may be used stand alone or in combination with other assets to solve other tasks, perhaps interdisciplinary. Thus, the set of connections is unique for every different way of using the platform, albeit the set of connections is expected to show similarities. Possibly, one GraphML file for each way of running the platform will be needed. Considering the effort of defining tags and tracking the connectivity, a cost verses benefit consideration is needed. This is to estimate how much effort it is reasonable to put in given the benefit of facilitated changing and increased lucidity of the platform. This is a difficult consideration since factors like reducing the risk of errors when changing will have to be considered.

The interviewed staff at the company is now considering introducing a procedure to gradually identify dependencies within their platform. This will be done starting in a limited area, covering a single discipline. If it can be made sufficiently convenient to define tags and relations, they hope to gain user acceptance so that it by time can be expanded to complete projects.

It can be assumed that not all dependencies are equally important. Perhaps it will be possibly to find what types of dependencies that are particularly important to keep track of. It may be that some relations can be omitted without compromising the value of dependency tracing too much. This will save time and effort. Dependencies that are manually defined as opposed to those that exist because there are function and procedure calls in the code are more important, the interviews indicated.

From the interviews it is thought to be useful to extract dependencies belonging to different disciplines. Then, visualisations that are adapted to the working roles in the company can be created. One of the important advantages that is gained by increased traceability and transparency is that working with continuous improvements is facilitated. If a script has been replaced with an improved one, knowing where this script is used will make it possible to predict the consequences of using the improved script everywhere in the system.

It is assumed that keeping up to date with the newest developments within PDM is important for the company. Perhaps more functionality will be added to commercial PDM for assisting in keeping track of the dependencies on a high granularity level and keeping a record on the nature of these dependencies.

## 7 Conclusions and future work

A system for defining, capturing and tracing dependencies between the assets of platforms has been presented. The problems addressed are quickly getting an overview of the platform connectivity to facilitate maintenance and extension of the platform and to provide a possibility for novel users to quickly understand how the platform assets are connected to solve a particular task. It has also been proposed how to include platform documents without compromising their appearance. The paper has addressed the need of capturing relations on a finer granularity level than what is normally found in PDM systems. Several implications of this have been observed. The relations depend on what task the platform is used for. Thus, there will be one set of relations for each task, making it difficult to predict the amount of relations and the effort needed to keep track of them.

One insight from interview evaluation is that the possibility to get an overview of the platform connectivity in order to predict the effect of changes to the platform and to provide is a major motivation for DM. Managing the relations between the various assets of the platform over time is a step towards PLM philosophy since the relations represents some corporate intellectual capital.

So far, tracing dependencies in heterogeneous platforms has only been tried in a single organisation and in a limited implementation. There may be other ways in which the platform assets are used. The question of if there are any dependencies are particularly value adding to trace remain unanswered. An implementation with the actual design teams of the company is needed to get into the details of pros and cons of platform asset dependency tracing.

## References

Algeddawy, T. and Elmaraghy, H. (2013) ‘Optimum granularity level of modular product design architecture’, *CIRP Annals – Manufacturing Technology*, Vol. 62, pp.151–154.

André, S. and Elgh, F. (2018) ‘Modeling of transdisciplinary engineering assets using the design platform approach for improved customization ability’, *Advanced Engineering Informatics*, Vol. 38, pp.277–290.

Blessing, L.T.M. (2009) *DRM, A Design Research Methodology*, Springer, London.

Chan, E. and Yu, K.M. (2007) ‘A concurrency control model for PDM systems’, *Computers in Industry*, Vol. 58, Nos. 8–9, pp.823–831.

Chapman, C.B. and Pinfold, M. (2001) ‘The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure’, *Advances in Engineering Software*, Vol. 32, No. 12, pp.903–912.

Cho, J., Vosgien, T., Prante, T. and Gerhard, D. (2016) ‘KBE-PLM integration schema for engineering knowledge re-use and design automation’, in Harik, R., Rivest, L., Bernard, A., Eynard, B. and Bouras, A. (Eds.): *Product Lifecycle Management for Digital Transformation of Industries: 13th IFIP WG 5.1 International Conference, PLM 2016*, 11–13 July, Columbia, SC, USA, Revised Selected Papers, Springer International Publishing, Cham.

David, M. and Rowe, F. (2016) ‘What does PLMS (product lifecycle management systems) manage: data or documents? Complementarity and contingency for SMEs’, *Computers in Industry*, Vol. 75, pp.140–150.

Elgh, F. (2014) ‘Automated engineer-to-order systems – a task-oriented approach to enable traceability of design rationale’, *International Journal of Agile Systems and Management*, Vol. 7, Nos. 2–3, pp.324–347.

Elgh, F. and Cederfeldt, M. (2008) 'Cost-based producibility assessment: analysis and synthesis approaches through design automation', *Journal of Engineering Design*, Vol. 19, No. 2, pp.113–130.

Elgh, F. and Johansson, J. (2014) 'Knowledge object – a concept for task modelling supporting design automation', in Cha, J. et al. (Eds.): *Development to Service Clouds in the Global Economy*, IOS Press, Vol. 1, pp.192–203.

Hjertberg, T., Stolt, R., and Elgh, F. (2018) 'A tool for obtaining transparency and traceability in heterogeneous design automation environments', *Computer-Aided Design and Applications*, Vol. 15, No. 4, pp.488–500.

Hvam, L. (2008) *Product Customization*, 1st ed., Springer, Berlin.

Johannesson, H., Landahl, J., Levandowski, C. and Raudberget, D. (2017) 'Development of product platforms: theory and methodology', *Concurrent Engineering Research and Applications*, Vol. 25, No. 3, pp.195–211.

Johansson, J. (2011) 'How to build flexible design automation systems for manufacturability analysis of the draw bending of aluminum profiles', *Journal of Manufacturing Science and Engineering, Transactions of the ASME*, Vol. 133, No. 6, p.133.

Johansson, J. (2015) 'Howtomation© suite: a novel tool for flexible design automation', *Advances in Transdisciplinary Engineering*, Vol. 2, pp.327–336.

Johansson, J. (2017) 'Analysing engineering knowledge in cad-models and spread sheets using graph theory and filtering', in *Transdisciplinary Engineering Conference 2017: Proceedings of the 24th ISPE Inc. International Conference on Transdisciplinary Engineering*, IOS Press BV, Netherlands, Vol. 5, pp.629–638.

Kristianto, Y., Helo, P. and Jiao, R.J. (2015) 'A system level product configurator for engineer-to-order supply chains', *Computers in Industry*, Vol. 72, pp.82–91.

Monticolo, D., Badin, J., Gomes, S., Bonjour, E. and Chamoret, D. (2015) 'A meta-model for knowledge configuration management to support collaborative engineering', *Computers in Industry*, Vol. 66, pp.11–20.

Pabolu, V.K.R., Stolt, R. and Johansson, J. (2016) 'Manufacturability analysis for welding – a case study using howtomation© suite', in *Transdisciplinary Engineering Conference 2016: Proceedings of the 23rd ISPE Inc. International Conference on Transdisciplinary Engineering*, IOS Press BV, Netherlands, Vol. 4, pp.695–704.

Poorkiany, M., Johansson, J. and Elgh, F. (2016a) 'An explorative study on management and maintenance of systems for design and manufacture of customized products', *IEEE International Conference on Industrial Engineering and Engineering Management*, pp.1453–1457.

Poorkiany, M., Johansson, J. and Elgh, F. (2016b) 'Capturing, structuring and accessing design rationale in integrated product design and manufacturing processes', *Advanced Engineering Informatics*, Vol. 30, No. 3, pp.522–536.

Quintana-Amate, S., Bermell-Garcia, P., Tiwari, A. and Turner, C.J. (2017) 'A new knowledge sourcing framework for knowledge-based engineering: an aerospace industry case study', *Computers and Industrial Engineering*, Vol. 104, pp.35–50.

Salchner, M., Stadler, S., Hirz, M., Mayr, J. and Ameye, J. (2016) 'Multi-CAD approach for knowledge-based design methods', *Computer-Aided Design and Applications*, Vol. 13, No. 4, pp.471–483.

Sandberg, M., Tyapin, I., Kokkolaras, M., Lundbladh, A., and Isaksson, O. (2017) 'A knowledge-based master model approach exemplified with jet engine structural design', *Computers in Industry*, Vol. 85, pp.31–38.

Stokes, M. (2001) *Managing Engineering Knowledge: MOKA: Methodology for Knowledge Based Engineering Applications*, Professional Engineering Publishing, London.

Stolt, R., Johansson, J., André, S., Heikkinen, T. and Elgh, F. (2016) 'How to challenge fluctuating requirements – results from three companies', in *Transdisciplinary Engineering Conference 2016: Proceedings of the 23rd ISPE Inc. International Conference on Transdisciplinary Engineering*, IOS Press BV, Netherlands, Vol. 4, pp.1061–1070.

Tiwari, V., Jain, P.K. and Tandon, P. (2015) 'Design decision automation support through knowledge template CAD model', *Computer-Aided Design and Applications*, Vol. 12, No. 1, pp.96–103.

Trehan, V., Chapman, C. and Raju, P. (2015) 'Informal and formal modelling of engineering processes for design automation using knowledge based engineering', *Journal of Zhejiang University: Science A*, Vol. 16, No. 9, pp.706–723.

Zhang, L.L. (2015) 'A literature review on multitype platforming and framework for future research', *International Journal of Production Economics*, Vol. 168, pp.1–12.