Uski, P., Nieminen, J. & Ellman, A. (2023) Will Model-based Definition Accelerate the Inspection Phase in the Manufacturing Process? Teoksessa Proceedings of the 24th International Conference on Engineering Design (ICED23). Cambridge University Press, s. 3899-3908.

WILL MODEL-BASED DEFINITION ACCELERATE THE INSPECTION PHASE IN THE MANUFACTURING PROCESS?

Uski, Pekka (1);
Nieminen, Joni (2);
Ellman, Asko (3)

1: Etteplan;
2: Tampere University of Applied Sciences;
3: Tampere University

ABSTRACT

Model-Based Definition provides several benefits for communicating between engineering and other downstream stakeholders. Particularly, semantic PMI information included in 3D models benefits both CAM programming and inspection phases. However, the efficiency of generating CAM- and CAI codes automatically according to the semantic PMI information varies due to the compatibility of different systems. This paper focuses on the experiment that we made on inspecting three different parts with the Coordinate Measuring Machine (CMM). We compared four different programming methods and found that the efficiency of inspecting depends on the serial size of the parts. The completely automatic CAI-programming method does not necessarily produce the most effective CAI code compared to the competent human programmer. This is notable, especially in large series due to achieved cumulative time savings in the inspection of each part. With small series, fully automatic PMI method and human-assisted automatic PMIfp method provide significant benefits in the inspection process due to time savings in CAI-programming work.

Keywords: Computer Aided Design (CAD), Product Lifecycle Management (PLM), Tolerance representation and management

Contact:
Uski, Pekka
Etteplan
Finland
pekka.uski@tuni.fi

Cite this article: Uski, P., Nieminen, J., Ellman, A. (2023) 'Will Model-Based Definition Accelerate the Inspection Phase in the Manufacturing Process?', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.391

1 INTRODUCTION

Model-Based Definition (MBD) enables several automatic functionalities in manufacturing process due to semantic machine-readable model (Völz et al., 2010), which contains not only 3D geometric, but also Product and Manufacturing (PMI) information (Quintana et al., 2010) (Agovic et al., 2022). A Design engineer distributes information with the MBD model via the Digital Thread to other downstream stakeholders. In the ideal process, the MBD model automatically delivers information of materials, surface treatment, and tolerances to CAM (Computer Aided Manufacturing), and to CAI (Computer Aided Inspection) programming. The CAI is mentioned here in a measuring context where Coordinate Measuring Machine (CMM) is used. Ideally, material information that is defined in metadata (Riley & National Information Standards Organization (U.S.), 2017) of the 3D model, is delivered automatically within the MBD model to CAM programming and for example to ERP (Enterprise Resource Planning) and MES (Manufacturing Execution System). However, based on our observations on mechanical engineering and manufacturing industry in Finland, this process is rarely ideal (Uski et al., 2020). In a typical process, the material information is first entered to ERP and then typed to CAM programming. Also, the surface roughness value is typed manually to the CAM code, because this information is rarely automatically transferred from the 3D model to CAM code.

Mechanical engineering industry in Finland consists mainly small and medium-sized enterprises (SME) (*SME Definition*, n.d.) and significant share of SMEs is micro-sized companies. This means companies have limited resources to invest in software and applications. Therefore, some companies have modest resources to read 3D models. In our previous study on a manufacturing ecosystem (Uski et al., 2020) we found out that due to variations in the MBD maturity of individual companies, utilization the capability of MBD needs to be motivated by business factors.

Figure 1 describes how CAI can be integrated into Model-Based Enterprise (MBE) with other subprocesses. In this ideal process, the CAM code is downloaded from the CAM software to a machine centre's (MC) Computer Numeric Control (CNC). Then the machined part is delivered to a CMM for inspection. This phase is described as the Part-labelled yellow arrow in Figure 1. The CMM is comparing dimensions of the machined part to the MBD model and in case of unacceptable values, feedback is transferred to the engineering phase. Then the Design engineer can plan following actions and decides whether it is needed to change the 3D model or make corrective machining operations. A system where all enterprise's subsystems get information from and where the original data locates is called as Digital Thread (Singh & Willcox, 2018). This single-source of truth enables also the re-use of design information. (Ellman et al., 2018).

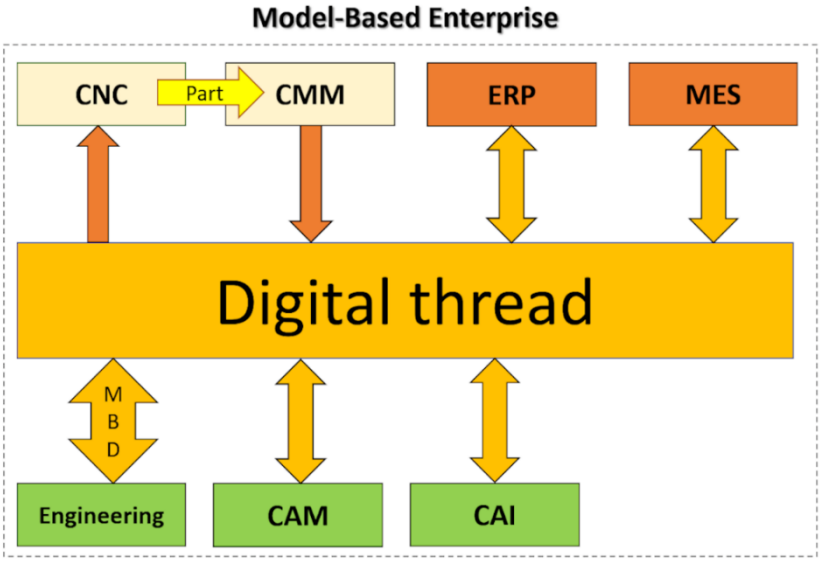

Figure 1. Digital product process.

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

Völz et al. introduces an approach where a semantic annotated model reduces the communication gap between global product development teams (Völz et al., 2010). The purpose of the MBD is to increase understanding of communication between engineering and downstream stakeholders in product development process. This paper focuses on the CAI phase of the MBD process. The paper compares efficiency of CAI phase in the case of different methods for producing the CAI code. From the experience it is known that manually generated CAI code can be more efficient than automatically generated CAI code. However, manual CAI coding takes time. Therefore, usefulness of manual coding is dependent on number of inspected parts.
We study inspection phase in the case of three dissimilar parts. We compare CAI programming times and corresponding part inspection times in the case of four different methods for CAI code generation. The research questions of this paper are:

- Can manually generated CAI code be more efficient than automatically generated code?
- How CAI coding method affects to a meaningful number of inspected parts and inspection time?
- What are reasons to the gap between coding methods?

2 CAI MEASUREMENT

This paper describes the measurement of milled parts that we also investigated in our earlier paper of the one-off production experiment (Uski et al., 2022). Furthermore, we studied one 3D-printed part. For the measuring we used Mitutoyo Crysta Apex 574S Coordinate Measuring Machine (Fig. 2), for CAI programming Mitutoyo MiCat Planner v1.7, and Mitutoyo MCosmos 4.3 for planning the measuring paths. For 3D printing we chose Stratasys F170, which is an FDM (Fused Deposition Modelling) 3D printer.

Figure 2. CMM with various sizes of measuring probes.

The CMM utilizes several various kinds of measuring probes (Fig. 2). We used single-head probes touching surfaces from different directions due to jointed probe.

2.1 Test specimens

We measured three dissimilar parts, Part 1, Part 2, and Part 3 (Fig. 3) and compared how long time it took time to generate CAI code. Parts 1 and 2 are machined and Part 3 is 3D-printed part. Parts 1 and 2 were used in the collaborative industrial project (Uski et al., 2020). Part 3 we created for this experiment to represent complex part for inspection. This paper focuses on investigating what is the benefit generating the measuring code, CAI code, automatically utilizing PMI information of the semantic 3D model.

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

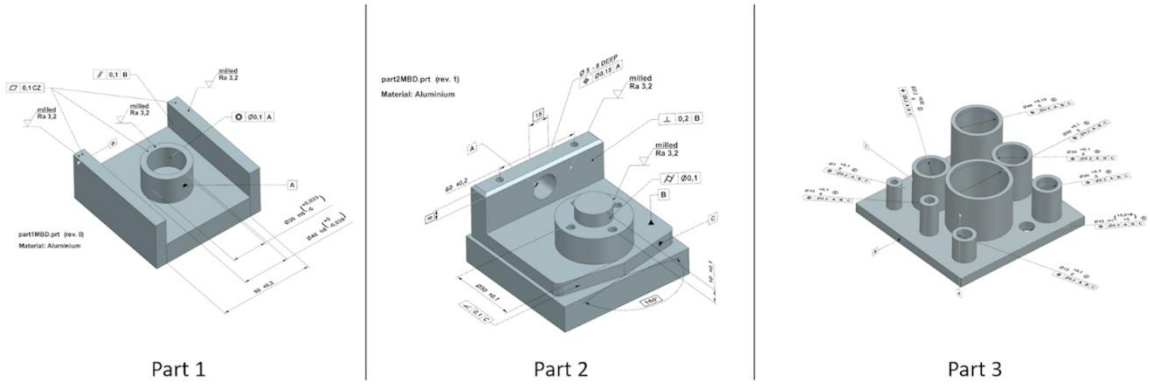

Figure 3. Parts 1…3 with semantic PMI information.

We did the experiment with the following steps:

- Preparation
  - Picking the fixture
  - Running to the origin
- Programming
  - Toolpath planning
  - Code generating with the postprocessor.
- Measuring

The Preparation phase is the same for all experiments, and we used the same fixture for both machined parts, but different for the 3D-printed part. The Programming method varies according to the experiment. We executed the experiment in phases that we describe in sections 3.1…3.4. In this experiment our operator has over twenty-years' experience in manufacturing and inspection professionality, and he made the CAI codes and observed the measuring phases. This experiment represents one of the typical programming and measuring operations in industry.

3 CAI-PROGRAMMING METHODS

In all experiments in sections 3.1…3.4 parts' zero point in the measurement area of the machine was defined the same way and saved in the memory of the software and used from there in each programming method. The program in each experiment would use this as baseline location and re-measure it to eliminate the possible positioning errors between parts as would be the case in production programs.

3.1 Manual programming

We programmed the CMM manually using the machines control software by manually telling it what and where to measure. Measuring information was programmed based by technical drawing and the programmer decided measurement point amounts, locations, and patterns. Created programs was then executed with the same software. We call this method "Manual". Significant issue with this method is that CMM is needed during the CAI-programming phase and this method represents online programming. The CMM executes the commands after each programming command is entered which helps to quickly detect possible programming mistakes but reserves the machine for programming instead for production. While programming with this method also produces the measurements for a "first part" the execution times are given from based on the finished program that would be used in production setting, same with the method described in section 3.2.

3.2 Programming utilizing the 3D model

In this phase we used an add-on for the CMM's control software which allowed use to use a CAD model to program the machine. Measurement information was still interpreted from technical drawing, but locations and feature orientations could be determined by clicking a 3D-CAD model instead of using manual input as in method in section 3.1. This method makes it also easier to avoid obstacles in

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

and between measurements since the program can use the model to calculate measurement paths. As this is an add-on to the control software this also reserves the machine for programming and executes the measurements after each command. We call this method as “CAD” and it is online programming.

3.3 Programming utilizing the semantic PMI-annotated 3D models

In this method we used an external software which can run without being connected to the CMM. This allows the CMM to be available for production measurements while other parts are being programmed. Programming is done by loading the machine model, fixture model and semantic PMI-annotated part model (MBD model) to the software. Then those models' relative positions are specified and then their location in the machine is specified. Program automatically goes through the PMI information and compares it to the machine models information, which also contains the machines movement area and different probes and orientations configured in the machine. Based on those the software determines which probe and orientation to use in which feature. Measurement patterns, methods and point amount are automatically determined by the software’s rules library. This method eliminates the variance in results between programmers because measurement strategies are determined by the common library. If any features cannot be measured, the software will inform the user to make necessary corrections in the measurement program. Then user must define which datum features are used to orient the programs coordinate system. Finally, user generates measurement paths, and the software goes through the paths and determines the optimal measurement order for the features. As this point we can simulate the generated program and transfer it to CMM’s control software. We call this method as “PMI”.

3.4 Programming with the PMI annotations and forced probe

This method differs from the method in section 3.3 at the simulation phase where we looked at the probe and orientation changes. Occasionally, the software decided to change either probe or orientation without visually significant benefit to the measurement. Probe and orientation changes extend the programs execution time. In these cases, the programmer selected the probe and orientation manually to try to make it faster instead of letting the machine choose automatically. We call this method “PMIfp”. The acronym fp means the forced probe.

4 RESULTS

4.1 Measurements and comparation of CAI methods with test specimens

The measurements consist of CAI-programming time and measuring time for a single part in case of each CAI-programming method. These characteristics varies significantly according to CAI-programming method used. Usefulness of CAI-programming method depends on serial size. One method may offer a fast CAI-programming time but slow measuring time, for example. Therefore, total time needed for inspection of n parts can be estimated from equation 1 we have formulated:

$$T_{Tot}^{Meth} = T_{Prog}^{Meth} + n \times T_{Mea}^{Meth} \quad (1)$$

where,

T_{Tot}^{Meth} = Total inspection time with a selected method,

T_{Prog}^{Meth} = Programming time with a selected method,

n = Number of parts,

T_{Mea}^{Meth} = Measuring time of a single part with a selected method.

Table 1 presents the measurements for Part 1 in case of four different CAI-programming method. In this case the PMIfp method offers both fastest programming and measurement times. In this case, it is the best method independent on serial size. Furthermore, it is notable, that PMIfp method offers faster measurement time than fully automated PMI method. This is one example how human knowledge in CAI programming wins automated programming method. Figure 4 presents total inspection times for Part 1 as a function of serial size. This time excludes fixing time of the part.

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

Table 1. Inspection results for Part 1.

Method	Manual	CAD	PMI	PMIfp
Programming [s]	1800	900	301	301
Measuring [s]	186	168	208	145

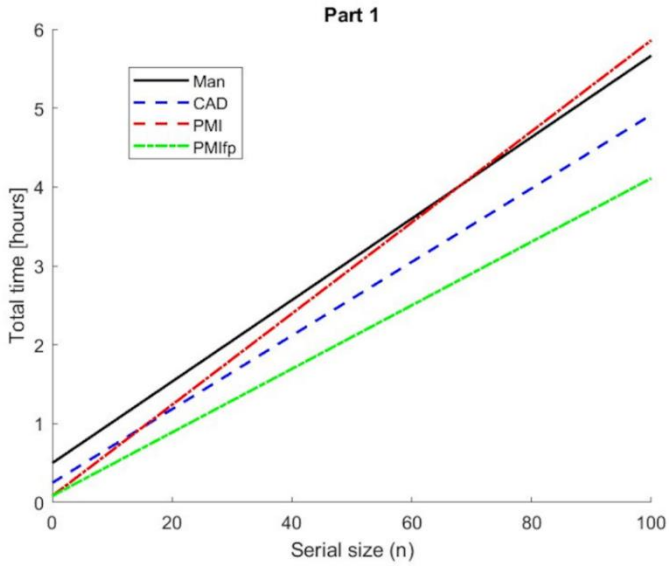

Figure 4. Total inspection times for Part 1 in case of different CAI-programming methods.

Table 2 presents the measurements for Part 2 in case of four different CAI-programming method. Respectively, the figure 5 presents total inspection times for Part 2 as a function of serial size.

Table 2. Inspection results for Part 2.

Method	Manual	CAD	PMI	PMIfp
Programming [s]	3600	1680	302	302
Measuring [s]	244	256	347	222

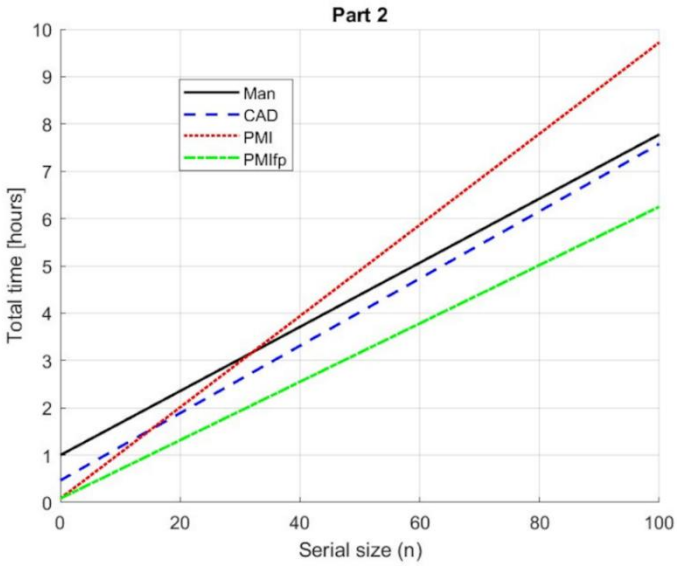

Figure 5. Total inspection times for Part 2 in case of different CAI-programming methods.

Also, in this case the PMIfp method offers both fastest programming and measurement times and therefore it is the best method independent on serial size. However, if this method could not be used,

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

both Manual- and CAD method would lead to better result within higher serial size compared to PMI method. Table 3 presents the measurements for Part 3 in case of three different CAI-programming method. Respectively, Figure 6 presents total inspection times for part 3 as a function of serial size. Due to the complexity of Part 3, the manual method turned out to be too time consuming and therefore, it was given up. Furthermore, in this case both PMI- and PMIfp methods are giving the same results. This is due to fact that in this case automated CAI-programming method has selected the optimal measurement probe.

Table 3. Inspection results for Part 3.

Method	Manual	CAD	PMI	PMIfp
Programming [s]	-	3900	60	60
Measuring [s]	-	478	588	588

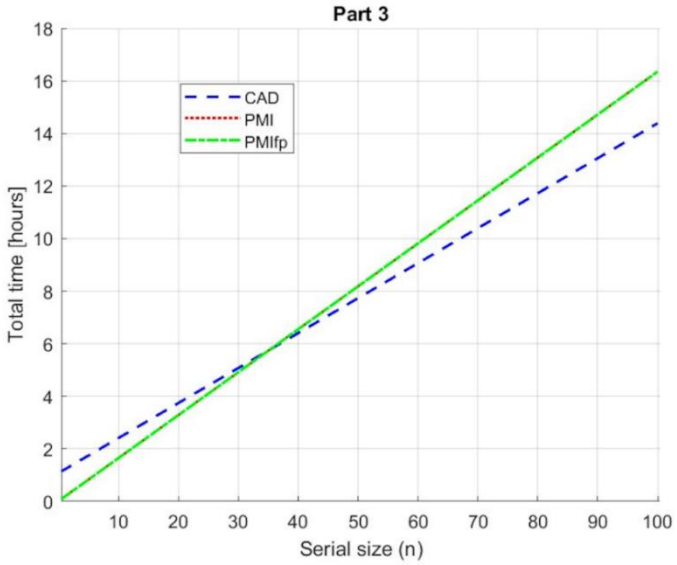

Figure 6. Total inspection times for Part 3 in case of different CAI-programming methods.

4.2 Comparation of CAI methods

CAI methods used in this study differs from time required to program a CAI code as well as from measurement time of one inspected part. Generally, if CAI code is more optimized, it's programming time will be longer, but it enables shorter measurement time. Therefore, some CAI methods are more suitable for a small production series and some for large production series. Longer programming time is acceptable, if it leads to a shorter total time in certain size of production series. In the following our formulated inequality we compare one CAI method (Meth) to an alternative CAI method (Ref), within difference of achieved programming time and measurement time in the case of a certain production series.

$$T_{Prog}^{Meth} - T_{Prog}^{Ref} - n \times \left(T_{Mea}^{Meth} - T_{Mea}^{Ref}\right) < 0 \quad (2)$$

where,

T_{Prog}^{Meth} = Programming time with a selected method,

T_{Prog}^{Ref} = Programming time with a reference method,

n = Number of parts,

T_{Mea}^{Meth} = Measuring time with a selected method,

T_{Mea}^{Ref} = Measuring time with a selected reference method.

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

Depending on the sign of difference in measuring times the inequality gives two alternative solutions:

$$n > \frac{T_{Prog}^{Meth} - T_{Prog}^{Ref}}{T_{Mea}^{Meth} - T_{Mea}^{Ref}} \; if \; \left(T_{Mea}^{Meth} - T_{Mea}^{Ref}\right) > 0 \tag{3}$$

$$n < \frac{T_{Prog}^{Meth} - T_{Prog}^{Ref}}{T_{Mea}^{Meth} - T_{Mea}^{Ref}} \; if \; \left(T_{Mea}^{Meth} - T_{Mea}^{Ref}\right) < 0 \tag{4}$$

If difference of programming times is zero, then difference in programming times solves the selection of the best CAI method.

Table 4. Comparation of serial size between CAI methods for Part 1.

Part 1		Reference method			
CAI-Method		Man	CAD	PMI	PMIfp
	Man		0	> 68	0
	CAD	∞		> 15	0
	PMI	< 68	< 15		0
	PMIfp	∞	∞	∞	

Comparing one CAI method to reference methods for Part 1 (Table 4) it is noticed that the fully automatic PMI method is effective with the serial sizes less than 68 parts with the Manual method. PMI method compared to the CAD method is effective with the serial sizes less than 15 parts.

Table 5. Comparation of serial size between CAI methods for Part 2.

Part 2		Reference method			
CAI-Method		Man	CAD	PMI	PMIfp
	Man		> 160	> 32	0
	CAD	< 160		> 15	0
	PMI	< 32	< 15		0
	PMIfp	∞	∞	∞	

Comparing PMI method to reference methods for Part 2, it is seen in Table 5 that PMI method is effective with serial size less than 32 parts compared to the Manual method and less than 15 parts compared to the CAD method. The CAD method is effective with the serial size of less than 160 parts compared to the Manual method and more than 15 parts compared to the PMI method.

Table 6. Comparation of serial size between CAI methods for Part 3.

Part 3		Reference method			
CAI-Method		Man	CAD	PMI	PMIfp
	Man				
	CAD			> 35	> 35
	PMI		< 35		∞
	PMIfp		< 35	∞	

Inspecting the Part 3 with Manual method was irrelevant due to the complex nature of the part. Therefore, other inspection methods were investigated, and it was noticed that both PMI methods are effective with the serial size of less than 35 parts.

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

5 DISCUSSIONS

We acknowledge that our experiment was done by one programmer and measuring operator only. Furthermore, the CAI coding was done with one software and CMM only. Therefore, these results may not be generalized. However, they point out practical and an important issue in a realistic industrial application. The programmer and measuring operator used in this study has professional level experience on CAI programming and using the CMM we used in this experiment.
The answers for the research questions are listed in following:

- Can manually generated CAI code be more efficient than automatically generated code?
  - An experienced operator can optimize the measuring probe selection and adjust toolpaths for them. Human-assisted CAI coding is more effective than automatically with small series of measuring parts.
  - However, when the part's complexity increases, manual CAI coding is a slow process. This is illustrated in Table 6 where manual coding became irrelevant.
- How CAI coding method affects to a meaningful number of inspected parts and inspection time?
  - Manual method suits for larger serial sizes when coding time is irrelevant compared to whole processing time.
- What are reasons to the gap between coding methods?
  - An experienced operator can optimize the inspection process by selecting suitable probes and design toolpaths for them. Fully automatic PMI method has limited abilities to do same.
  - CAI programming software do not recognize all PMI annotations.

6 CONCLUSIONS

Our finding in this experiment is that PMI method can be effective in small and large series depending on serial size. CAI programming with manual assistance can produce more effective CAI code than fully automatic coding because experienced operator can select optimal measuring probes as well as control probe changes. In addition, an experienced operator can decide how many measuring points is useful to select according to round hole diameter variations. For instance, similar number of measuring points is not effective for all sizes of holes, in other words, small holes need to be measured fewer measuring points than large holes. To understand what is small and what is large hole, needs to be defined in the measuring program.

REFERENCES

Agovic, A., Trautner, T., & Bleicher, F. (2022). Digital Transformation - Implementation of Drawingless Manufacturing: A Case Study. *Procedia CIRP*, *107*, 1479–1484. https://doi.org/10.1016/j.procir.2022.05.178

Ellman, A., Paronen, J., Juuti, T. S., & Tiainen, T. (2018). Re-use of engineering design rationale in Finnish SME project based industry. *Proceedings of International Design Conference, DESIGN*, *4*, 1825–1832. https://doi.org/10.21278/idc.2018.0363

Quintana, V., Rivest, L., Pellerin, R., Venne, F., & Kheddouci, F. (2010). Will Model-based Definition replace engineering drawings throughout the product lifecycle? A global perspective from aerospace industry. *Computers in Industry*, *61*(5), 497–508. https://doi.org/10.1016/j.compind.2010.01.005

Riley, J., & National Information Standards Organization (U.S.). (2017). *Understanding metadata : what is metadata, and what is it for?* https://www.niso.org/publications/understanding-metadata-2017

Singh, V., & Willcox, K. E. (2018). Engineering design with digital thread. *AIAA Journal*, *56*(11). https://doi.org/10.2514/1.J057255

SME definition. (n.d.). Retrieved November 6, 2022, from https://single-market-economy.ec.europa.eu/smes/sme-definition_en

Uski, P., Ellman, A., Laine, I., Hillman, L., & Nieminen, J. (2022). Model-Based Definition Accelerates Product Life Cycle in Manufacturing and Inspection Phase-Experiment of Machined One-Off Production. *International Design Conference-Design* 2022, 643–652. https://doi.org/10.1017/pds.2022.66

Uski, P., Pulkkinen, A., Hillman, L., & Ellman, A. (2020). Issues on Introducing Model-Based Definition-Case of Manufacturing Ecosystem. *Product Lifecycle Management Enabling Smart X - 17th IFIP WG 5.1 International Conference, PLM 2020, Rapperswil, Switzerland, July 5-8*, 604–617. https://doi.org/10.1007/978-3-030-62807-9_48

Völz, D., Schule, A., & Anderl, R. (2010). An approach to use semantic annotations in global product development to bridge the gap in interdisciplinary and intercultural communication [Proceeding]. *WMSCI 2010 - The 14th World Multi-Conference on Systemics, Cybernetics and Informatics, Proceedings*, *3*, 49–54.

https://doi.org/10.1017/pds.2023.391 Published online by Cambridge University Press

CAMBRIDGE
UNIVERSITY PRESS