Enhancing the Flow of Information in the PLM by Using Numerical DSMs – An Industrial Case Study

Thomas Luft, Johanna Bochmann, Sandro Wartzack

Enhancing the Flow of Information in the PLM by Using Numerical DSMs – an Industrial Case Study

Thomas Luft[1], Johanna Bochmann[2], Sandro Wartzack[1],

[1] Friedrich-Alexander-University of Erlangen-Nuremberg, Chair of Engineering Design, Martensstrasse 9, 91058 Erlangen, Germany,
[2] Siemens AG, Energy Sector, Energy Service Division, Frankenstrasse 70-80, 90461 Nuremberg, Germany

luft@mfk.fau.de

Abstract. This paper proposes a methodology to enhance the information flow in the product life cycle through the example of the product development process. The methodology refines the existing approaches of revealing indirect dependencies with a multiple domain matrix by introducing the numerical derived design structure matrix as indicator of quantifying indirect dependencies between process steps, information and people. These dependencies are quantified by the pieces of information causing the indirect linkage. The subsequent analysis is narrowed down to the content level of information to identify the root causes. This methodology has been successfully tested in an industrial case study. As a result, this paper helps to manage the complexity in the PLM by making the linkages caused by the information flow visible to improve collaborative product development.

Keywords: Collaborative product development, information management, process management, design structure matrix, multiple domain matrix, case study.

1 Motivation and Objectives

The successful development of modern products is becoming increasingly difficult for companies because they are faced with many challenges such as the consideration of a variety of customer requirements, the compliance with Design for X rules regarding different issues that occur in different product life cycle phases as well as the increasing need of collaboration of diverse engineering disciplines in the product development process (PDP). This leads to a steadily growing complexity of the relationships and interactions between the following three important domains: ‘process steps’, ‘people/departments’ and ‘information’.

Due to these complexities, it comes to a lack of information exchange between the departments and consequently to delayed product development processes. One reason is that it is often not known how people (within the meaning of roles) are connected to each other based on the information they generate or receive [5]. The other reason is

that it is seldom understood how process steps are connected to each other based on the information being generated or used during a process step [5]. Thereby, the term information covers the 'knowledge and information objects' (KaI-object) according to Luft [6]. Therefore, not only the direct dependencies but also the indirect dependencies between these three domains create problems within the development of modern products.

Subsequently, the information flow between the affected employees and every single step of the development process has to be analyzed and enhanced in order to realize an improved flow of information [8]. The main objective of this paper is to provide a methodology which allows revealing and quantifying indirect dependencies between different departments (domain people) and different process steps (domain process) based on the exchanged information or KaI-objects (domain information), since these are not obvious as the direct dependencies. In order to do this, a Multiple Domain Matrix (MDM) that consists of several Design Structure Matrices (DSM) and Domain Mapping Matrices (DMM) is created. On this basis, the calculation of the so called numerical derived Design Structure Matrix (nd-DSM), which is introduced and presented in this paper, is possible. The nd-DSM allows besides the identification also the quantification of indirect dependencies. Thereafter, an analysis strategy is proposed in order to improve the information flow in the PDP. Using this methodology in an industrial case study, the authors show that not only the transparency of the dependencies between the three domains increases but also the flow of information in the collaborative product development process can be improved significantly.

2 State of the Art and Related Work

The current state of science as well as the relevant work regarding the objective pursued is described in this chapter. The PDP can be considered as a complex system of different subsystems which can be also called as domains [5]. The modeling of the entire system allows understanding the complexity of it by considering the different domains of interest (e.g. process steps, people, information) and the relationship between the respective elements [2]. The linkage of different elements of a certain subsystem can be modeled by the DSM developed by Steward [9]. The DSM is an intra-domain square matrix showing the direct dependencies of elements to each other, which are belonging to the same domain (figure 1). However, the DSM cannot represent the dependency between elements of different domains (e.g. certain information cannot be assigned to a process step).

In contrast, the DMM is an inter-domain matrix that reflects the direct dependencies between the elements of two different domains and is also considered as a cause-effect-matrix [9]. Thereby, the rows represent the elements of one domain while the columns show the elements of the other domain. Both types of matrices represent only direct dependencies, which become visible, e.g. by analyzing the product structure or by mapping the development process steps (figure 1). In the following the IR/FAD (input in the rows/feedback above the diagonal) convention according to Eppinger is applied to read the following matrices [4].

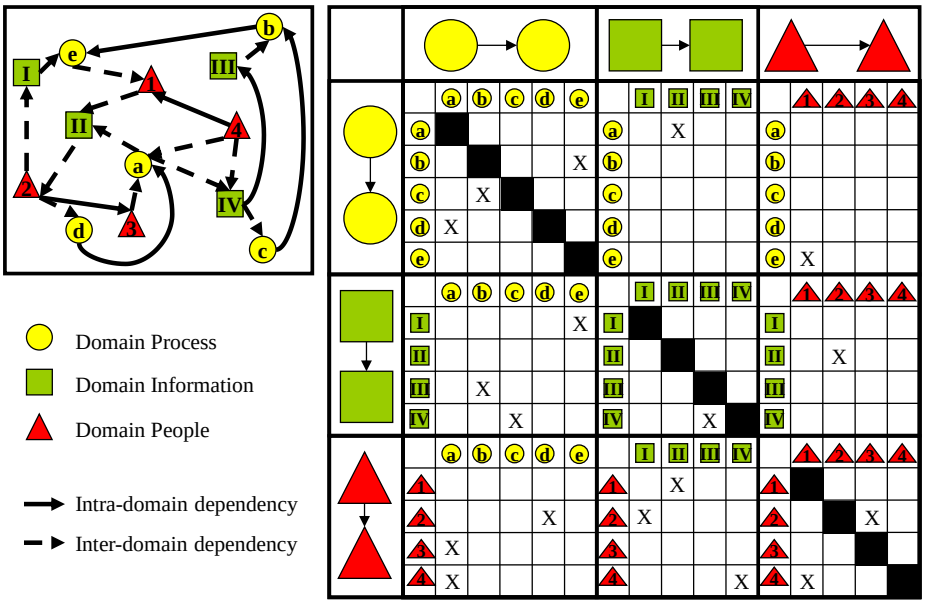


Fig. 1. Dependencies between the elements of the different domains in a MDM [cf. 5]

Maurer developed the MDM being a square matrix, which shows the direct dependencies within a single domain (DSM) and between different domains (DMM) [5] (figure 1). Consequently, the MDM consists of DSMs and DMMs. Nevertheless, for improving the PDP it is also important to understand and to analyze indirect dependencies. An indirect dependency is caused through the linkage of two elements of certain domain via one or more elements of another domain. As shown in figure 1 at the top on the right, person (2) is indirectly dependent on person (1) because person (1) generates a KaI-object (II) (e.g. a piece of information) and the other person (2) uses it, even though they are not directly linked (e.g. by belonging to the same department). The indirect dependencies are apparent from derived DSMs, which are the result of a matrix multiplication of different matrices of the MDM.

For enhancing the information flow in the PDP, the following question needs to be answered: How are the different departments as well as process steps indirectly dependent on each other based on the exchanged information [5]? However, the above mentioned matrices only display the existence of a dependency by setting a cross or the binary value ‘1’. This means no further information is available how strong the dependency is and therefore they are called binary matrices.

In literature several approaches (e.g. [3], [7] and [9]) exist to weight the dependency of elements of a certain domain by using numerical DSMs. While a binary DSM only indicates a relationship between elements by ‘0’ and ‘1’, a numerical DSM has the numerical value of the number of elements causing the dependency. Consequently, the advantage of numerical DSMs is that more information is given about the relationship between two elements. Numerical DSMs already considered in research

[3], [7] and [9] to be only applicable in a flow-directed environment by weighing the marks, which are indicating, for example, a rework loop in activity based DSMs.

However, all previous methods are based on a subjective assessment of the dependency and, moreover, the self-dependency is considered in none of these approaches. For this reason a methodology is proposed in the following, which allows to quantify the strength of indirect dependencies and to analyze these on a content level as well as to identify self-dependencies.

3 Methodology

The methodology for enhancing the flow of information between the participating departments in the PDP consists of three phases. The first phase is the identification of the problem area, followed by the modeling of the development process and finally the analysis of it. All steps can be done with usual spreadsheet software (e.g. Excel) and Matlab. So, no expensive and difficult-to-use software application is necessary.

First of all, the problem area of the PDP needs to be identified. Hereby, it is recommended to conduct explanatory interviews with the affected domains along the process chain of the PDP to get background knowledge about the holistic process. Besides the different pieces of information, the process steps themselves as well as the involved departments which exchange information throughout the process have to be investigated. So, a detailed analysis of the three relevant domains 'process', 'information' and 'people' needs to be carried out in order to enhance the information flow. As a first step, the process is mapped by using event-driven process chain (EPC) which is a type of flowchart and is in particular used for business process modeling. With the EPC the flow of activities of the process is modeled as functions which starts and ends with an event. Thus, the entire PDP is shown in its logical sequence.

Once the whole process is mapped with all its activities, the focus is on the information flow because the information exchange between the departments with their respective staff is seen as an enabler for a successful PDP. Therefore, according to Behncke, four different levels have to be taken into account [1]: The 'transmitter-receiver level' describes the communication between the departments, the second level 'content level' focuses on which information is exchanged whereas the 'information carrier level' considers how the information is transmitted between sender and receiver; the fourth level 'target level' is not in focus of this methodology because an evaluation and a categorization of information are not absolutely necessary.

Each process step within the workflow is analyzed from these three perspectives of an information flow. For this purpose, a so called information flow matrix (IFM), which is depicted in figure 2, is built up. The process step itself is marked by the activity (function of the EPC-diagram) and the organizational unit carrying out the process step (executer). For each process step the information input and output is determined. The focus is not only put on the transmitter-receiver relationship (first level or brown colored) but also on the content of the information (second level or orange colored) and carrier of it (third level or blue colored). Moreover, the fourth level 'storage of information' helps to identify whether the information stays at the same place. Finally, the perspective 'information type' is also considered (cf. [6]).

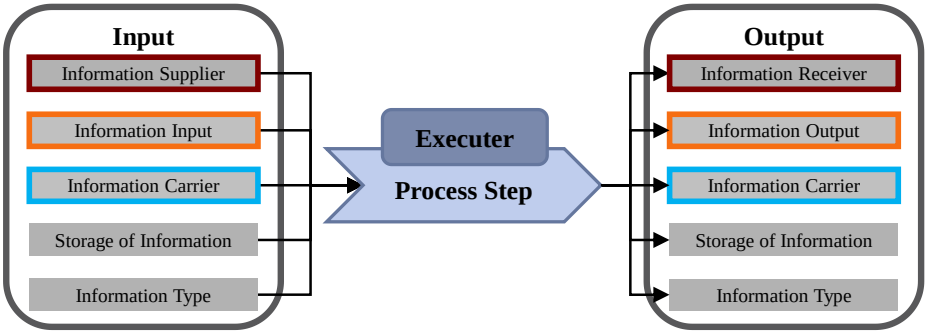


Fig. 2. Structure of the Information Flow Matrix

Since the process, information and people perspective is covered by the IFM, it is used as the basis for building up the MDM. Hereby, all elements of every domain are listed in the columns and in the rows of the MDM. It helps to use the pivot-function in Excel in order to prevent double entries in the MDM because some elements, especially of the domains information and people, exist more than once in the IFM. Therefore, three Pivot-tables are created of the specific rows (information input and output, executer, process step) of the IFM. The rows 'information input' and 'information output' are combined into one row. So, all elements of the domains are directly transferred to the rows and columns of the MDM.

Afterwards, all DMM fields need to be filled out with information from the IFM. It has to be ensured that the IR/FAD convention is used in order to fill in the correct data. The first DMM ($DMM_{Process \rightarrow Information\ Output}$) indicates which process step generates which information output. The $DMM_{Process \rightarrow People}$ shows the executer of each process step. This DMM is the transposed one to $DMM_{People \rightarrow Process}$ and therefore only one of the two DMMs need to be filled out. The entries of one of the two DMMs can be transferred to the opposed DMM by using the function 'paste' of the transposed values in Excel. The third $DMM_{Information\ Input \rightarrow People}$ shows which information the executer of the process step(s) needs in order to conduct the process step. Hereby, it is useful to insert the information process step by process step because the executer of a certain process step needs a certain information input. The $DMM_{People \rightarrow Information\ Output}$ considers the information output, which is generated by a certain executer. As in the opposed $DMM_{Information\ Input \rightarrow People}$ it is helpful to do this process step wise. The $DMM_{Information\ Input \rightarrow Process}$ focuses on the information input needed for a certain process step [5].

It is important to set the number '1' instead of a cross in the fields of the DMMs in which a dependency exists in order to use the matrix for further analysis. Sometimes information is used several times by the same executer and in these cases it is assumed that the information is still available for the executer. Consequently, the entered figure is not increased. The empty fields, which are representing the non-existing relationships, can be filled out easily with '0' by using the 'replaced-by'-function in Excel. The result is a binary MDM with the respective DMMs.

Once the MDM is filled out, the indirect dependencies can be determined. In accordance with the initially mentioned objective are two issues in the focus of this contribution (cf. [5]):

- How are people (or departments) connected based on the information they generate or receive?

- How are process steps connected to each other based on the information being generated or used during a process step?

For each dependency a separate matrix multiplication needs to be done. The DMMs are multiplied with each other in order to receive the nd-DSMs, which quantify the strength of the indirect dependency. It is recommended to conduct the matrix multiplication with the software Matlab to avoid the limitation of the array size of the matrices in Excel. For the first issue, the dependencies between people, equation 1 is applied while the second equitation 2 is needed for answering the second question about the dependency of the process steps. The formula for the matrix multiplication is written down in equation 3. Matrix A (a_{ij}) is a (m, n)-matrix and is multiplied with the (n, l)-Matrix B (b_{jk}). The product of this multiplication is matrix C (c_{ik}) with the array size (m, l).

$$\text{nd - DSM}_{\text{People}} = \text{DMM}_{\text{People}\rightarrow\text{Information Output}} \times \text{DMM}_{\text{Information Input}\rightarrow\text{People}} \quad (1)$$

$$\text{nd - DSM}_{\text{Process}} = \text{DMM}_{\text{Process}\rightarrow\text{Information Output}} \times \text{DMM}_{\text{Information Input}\rightarrow\text{Process}} \quad (2)$$

$$A \cdot B = (a_{ij}) \cdot (b_{jk}) = (c_{ik}) \quad \text{with} \quad c_{ik} = \sum_{j=1}^{n} a_{ij} \cdot b_{jk} \quad (3)$$

In order to get a better overview, these two nd-DSMs are entered into the MDM on the place of the original DSMs. This is only possible because the direct dependencies between elements of the same domain are not respected in this case. In cases of considering also direct dependencies, it is important to store the nd-DSMs in a separate matrix instead of overwriting the original DSMs (figure 3).

The multiplication of the blue (or marked) row with the blue (or marked) column results in the (pink) colored people-people-matrix. The department d_2 delivers three information (i_2, i_3, i_7) as an output while the department d_3 needs only two of them (i_2, i_7); besides three other information (i_1, i_4, i_6) as input. Hence, a dependency exists only if a certain piece of information (e.g. i_2) is generated by a department (e.g. d_2) and this one is required by another department (d_3). If the information (e.g. i_3) is not used (multiplication with zero), there is no dependency. The sum of the dependencies is corresponding to the number of information of which the departments or process steps are dependent. This allows giving a quantitative analysis about the strength of dependency, which is objective through the actual number of information (figure 3).

By considering the linkage of two different elements, both ways of dependencies with their weightings has to be considered in detail. As depicted in figure 3 at the bottom on the right (and schematically on the left), the element d_3 provides element d_2 four different pieces of information while d_2 only provides two pieces of information to d_3 (figure 3). This assumes the existence of different weighted dependencies. This weighted dependency based on the exchanged information can be expressed formally as follows: The element d_i of the domain d is dependent on the element d_j of the same

domain based on x different pieces of information. So, the information-generating element has to be before the information-receiving element. Thus, a sequential analysis of the diverse functions within the process is possible and can be used for process improvements as well as serves as a basis for concurrent engineering.

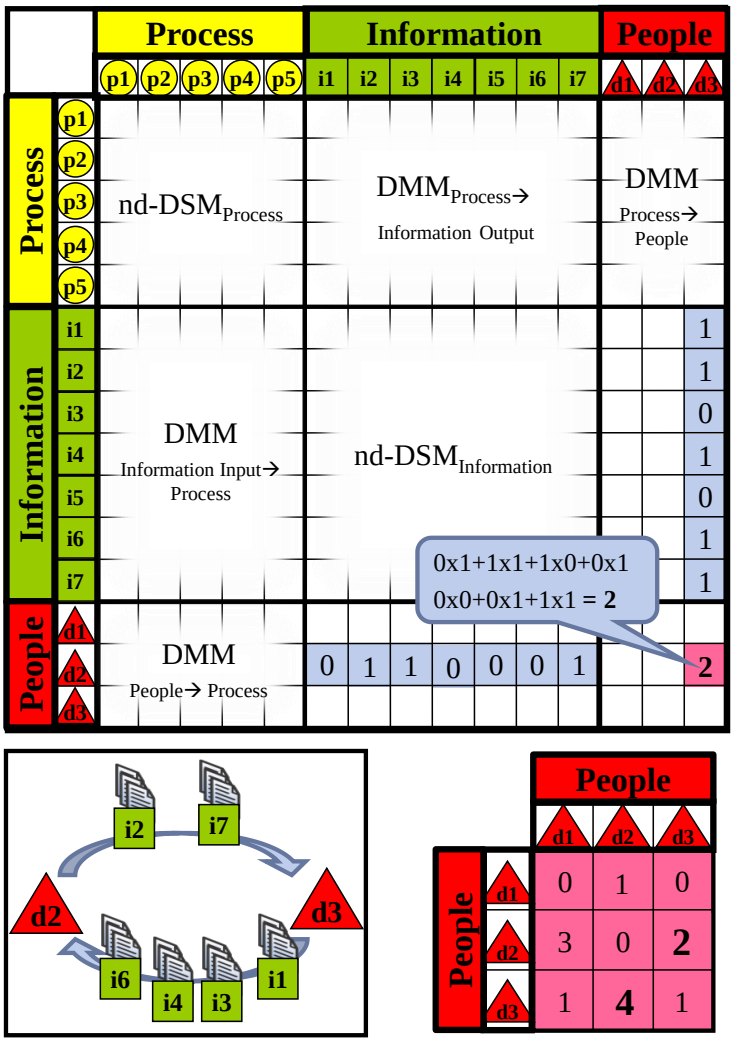


Fig. 3. The calculation of the nd-DSMPeople (simplified extract)

The analysis is done on a content level by looking on each individual KaI-object itself. Hereby, the knowledge about the process background is important in order to categorize the type of the different indirect dependencies. By considering both directions of a dependency, three different types of a mutual dependency are possible. These are summarized and elucidated briefly in figure 4.

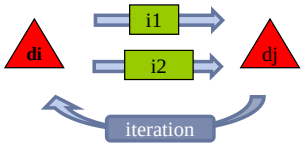
Graphical representation	Type	Example
	No mutual dependency on the same information.	Information i1 generated by di is needed by dj in order to generate information i2 which is needed by di.
	1) Negative iteration with rework due to wrong or missing information. 2) Positive iteration with a positive effect on the product development regarding the product's degree of maturity.	1) Information i1 coming from di to dj is not sufficient. A request is sent from dj to di regarding information i1. Di provides an improved or new information i2. 2) Information i1 coming from di is needed by dj. The information i1 is revised from dj due to e.g. new insights and sent back to di. Di provides an improved information i2 based on the revision by dj.
	Feedback loop with no effect on the current development	Information i1 generated by di is needed by dj. Dj sends a feedback (e.g. that i1 is okay) back to di regarding information i1, which does not effect the current development.

Fig. 4. Types of mutual dependencies

Besides analyzing both directions of dependencies, this methodology allows also the quantification of the self-dependency of elements. This is possible with the nd-DSMs as shown in figure 3, but only one direction of dependency can be considered because the information generating elements is at the same time the information receiving element. The self-dependency is shown by a value equal or greater one in the diagonal of the nd-DSMs. The self-dependency should not be out of focus because it shows that lots of information is generated and used by the element itself. The MDM with its nd-DSMs as shown in figure 3 can be used for analyzing the pieces of information that are responsible for the self-dependency. The self-dependencies are an indicator whether internal structure of certain elements needs to be set-up (especially if the figure in the diagonal is high) or whether they have to be evaluated regarding their efficiency.

4 Case Study

This methodology has been evaluated through its application in an industrial case study (development of a repair method for a certain expendable part of a gas turbine). The development process was mapped by the EPC-methodic in several workshops. It consists of 56 different process steps, includes 155 KaI-objects and 15 units (departments), which are executing the process steps, are identified by the IFM.

Based on the IFM, the MDM (226 elements) with its respective DMMs are built up. For this manual transferring, two days were needed in this case. Hence, a macro for speeding up the set-up is recommended. The pieces of information causing the indirect dependencies were calculated by the mentioned equations. Using Matlab, the multiplication of the matrices takes only a few seconds. This results in the nd-DSMs which are showing the strength of the indirect and self-dependencies between the different process steps and the involved departments.

The nd-DSM$_{People}$ of the industrial case study is illustrated in figure 5. The department d_{12}, for example, generates 38 pieces of information which are needed later in the process by the department itself. This reveals the demand for an identification and optimization of the internal information flow within this department.

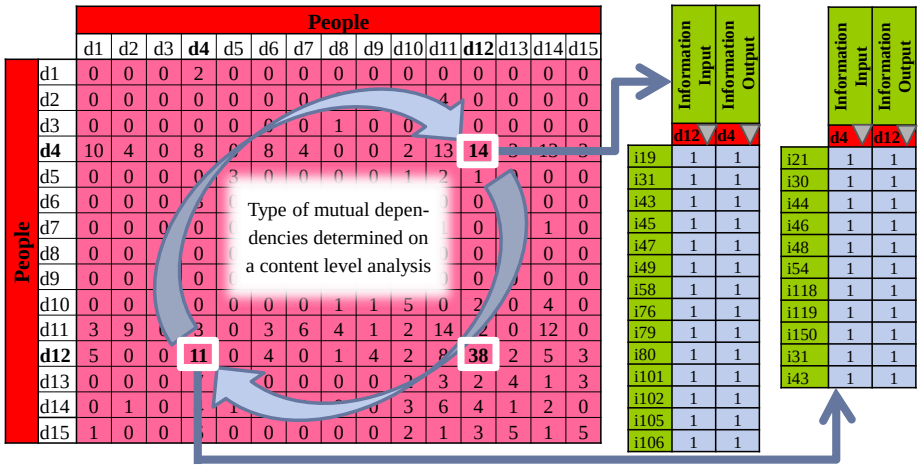


Fig. 5. The nd-DSM$_{People}$ together with the information dependency between d12 and d4

Besides this, the nd-DSM emphasizes the strength of the indirect dependencies by quantifying the pieces of information which cause the dependency. This quantitative information can be seen as advantage compared to the binary derived DSM proposed by Maurer [5]. High figures in the nd-DSM indicate elements being critical in product development. These have to be picked out for the content-level-analysis of the indirect dependencies by focusing on the KaI-objects itself [6].

In the shown industrial case, the information flow between the elements e.g. d_{12} and d_4 is of importance to understand their linkage. The department d_4 delivers fourteen pieces of information to d_{12} while eleven pieces of information are provided by d_{12} to d_4. To define the type of a mutual dependency between these elements, the content level of the pieces of information needs to be analyzed in detail. For this analysis, the two corresponding DMMs from the MDM, for example, DMM$_{Information \rightarrow People}$ and DMM$_{People \rightarrow Information}$, are used. The original, binary DMMs make it possible to use a filter for indicating whether a certain piece of information belongs to the element of interest. After transposing of the DMM$_{People \rightarrow Information}$, both DMMs are copied into a new sheet. So, the matrices have the elements of the domain people in columns and the information in rows. By selecting 'auto-filter', the two elements of interest are linked through one KaI-object. If the filter is set to '1' for d_4, only the KaI-objects are shown which are generated as an output by d_4. By setting the filter for d_{12} to '1', only the information gets visible that are generated as output by d_4 and used by d_{12} as input (as shown in figure 5 on the right). Afterwards, the type of mutual dependency can be determined by looking at the content level. The detailed knowledge about the KaI-objects causing the dependency enables enhancing the collaboration between the involved people since the information flow is the basis for cooperation.

6 Conclusion and Further Research

The proposed methodology allows identifying the pieces of information causing the indirect dependencies between two elements by quantifying it as a numerical value in the nd-DSMs what differs from existing approaches. One advantage is hereby that the analysis is narrowed down to the content level of the information. After identifying the case of mutual dependencies, this leads to an easier analyzing strategy and serves as a basis for improving the information flow between departments along the PDP. This allows developers to enhance the information exchange and to improve the coordination and collaboration in the PDP. The practical suitability of this approach was evaluated and successfully proven in an industrial case study.

The introduction of weighting factors might be of interest in order to differentiate between the importances of certain information. Furthermore, it would be also interesting to provide a guideline for classifying information in several levels to define different collaboration environments during the PDP because different people are interested in different information. This leads to further research topics.

Acknowledgment. Part of the work presented was supported by the German Research Foundation (DFG) within the research project "Product-oriented process management – iteration management based on a property-based product maturity".

References

1. Behncke, F., Gabriel, F., Langer, S., Heppler, C., Lindemann, U., Karl, F., Pohl, J., Schindler, S., Reinhart, G., Zaeh, M. (2011) 'Analysis of information flows at interfaces between strategic product planning, product development and production planning to support process management. A literature based approach' in Proceedings of the IEEE International Conference in Systems, Man and Cybernetics, Anchorage.
2. Browning, T.R. (2001) 'Applying the design structure matrix to system decomposition and integration problems' IEEE Transactions on Engineering Management, vol. 48, p. 292-306.
3. Browning, T.R., Eppinger, S.D. (2002) 'Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development', IEEE Transaction on Engineering Management, vol. 49, p. 428-442.
4. Eppinger, S.D., Browning, T.R. (2012) 'Design Structure Matrix Methods and Applications', The MIT Press, Cambridge.
5. Lindemann, U., Maurer, M., Braun, T. (2009) 'Structural Complexity Management - An Approach for the Field of Product Design', Springer, Berlin.
6. Luft, T., Wartzack, S. (2012) 'Requirement analysis for contextual management and supply of process- and design knowledge – a case study' in Proceedings of the 12th International Design Conference, Dubrovnik.
7. Pimmler, T.U., Eppinger, S.D. (1994) 'Integration Analysis of Product Decompositions' in Proceedings of the ASME Design Theory and Methodology Conference, Minneapolis.
8. Sandkuhl, K. (2011) 'Improving Engineering Change Management with Information Demand Patterns', in Proceedings of 8th International Conference on PLM, Eindhoven.
9. Steward, D.V. (1981) 'Design structure system: A method for managing the design of complex sytems', IEEE Transaction on Engineering Management, vol. 28, p.71-74.