10TH INTERNATIONAL DESIGN STRUCTURE MATRIX CONFERENCE, DSM'08
11 – 12 NOVEMBER 2008, STOCKHOLM, SWEDEN

USING THE DESIGN-STRUCTURE-MATRIX FOR THE AVOIDANCE OF UNNECESSARY ITERATIONS

J. Roelofsen[1], H. Krehmer[2], U. Lindemann[1] and H. Meerkamm[2]

[1] Institute of Product Development, TU München
[2] Chair of Engineering Design, University Erlangen-Nürnberg

Keywords: avoidance of iterations, classification of iterations, design situation, development process

1 INTRODUCTION

Due to the growing application of electrical, software-based and electronic components in technical systems the complexity in these systems and their development steadily increases. The participation of many different domains such as mechanical engineering, automatic control engineering, software design, electrical engineering, and information technology adds up to this challenge. Due to the often unclear definition of interfaces between the different development partners and the increasing complexity, a strong risk of unnecessary iterations during the development of those systems arises. This points out the demand for an approach to avoid unnecessary and unwanted iterations in product development processes. Those are iterations, which have no contribution to an increasing degree of product maturity and thus are to be recognised as time-consuming and cost-intensive detours. The aim of this contribution is to use the Design-Structure-Matrix as a tool for the preventive avoidance of those kinds of iterations.

2 APPROACH TOWARDS THE AVOIDANCE OF UNNECESSARY ITERATIONS

To be able to avoid unnecessary iterations and to distinguish the useful from the unwanted ones, it is required to define criterions, which denote iterations as unnecessary or as advantageous. Therefore it is necessary to classify the iterations regarding their possible causes and their influences on the development process.

2.1 Classification of iterations

In prior work an "approach on the control of iterations in the multidisciplinary development of technical systems" was presented [1]. This approach is based on a classification of iterations depending on research for the causes and influences of different kinds of iterations. It identifies two classes of iterations, the so called class of "large iteration" and the class of "small iteration". Another classification of iterations is presented in [2]. This classification identifies six different kinds of iterations: "Exploration", "Convergence", "Refinement", "Rework", "Negotiation" and "Repetition". In the following chapter the six kinds of iterations according to Wynn [2] will be integrated into the framework of the two classes of "large" and "small" iterations.

Small iteration

According to [1] the class of "small iteration" can be understood as a quantitative approximation towards the optimal solution. This means, the developer has to approach iteratively through several partial steps, which becomes necessary because often the solution cannot be found in one step due to the complexity of modern technical systems. This small iteration is necessary for engineering design and has to be supported. Four of the six kinds of iterations according to [2] can be integrated into this class of "small iteration":

"*Exploration*" according to [2] means an "iterative exploration of problem and solution spaces" which is determined by a "repeated process of space divergence" in synthesis followed by convergence in evaluation. This can be understood as the constant alternation from synthesis to analysis and vice versa. "*Convergence*" according to [2] is the iterative approach towards a "satisficing design", which gets necessary because of the fact that a solution can not be found in one step [2].

"*Refinement*" is the third kind of iteration according to [2], which can be seen as a part of the class of "small iteration" according to [1] and means the "further refinement to enhance secondary

characteristics", in cases where the product meets its primary requirements. "Convergence" has to precede "Refinement", because the solution first has to approach (converge) the optimal solution, not till then it is possible to refine this solution.
The fourth and last kind of iteration according to [2] which can be seen as part of the class of "small iteration" is the "*negotiation*". This means, that there are for example developers of different disciplines that are contributing together to achieve an acceptable solution in spite of competing goals.
These four kinds of iterations according to [2] have in common, that they are helpful and absolutely necessary in engineering design. So, all kinds of "small iterations" are to be supported during the development of technical systems.

Large iteration
Triggers for this class of iteration for example can be a change in the information basis, unclear requirements in the beginning of the process, a completion of the data basis by new cognitions regarding the total system. This means, that the development process must be run through again, so this class of iteration causes a return to the beginning of the product development process, why it is called the class of "large iteration" [1].
The iteration called "*Rework*" according to [2] is an equivalent to the class of "large iteration" and means that some "tasks may require rework in response to problems that emerge as analysis is conducted". Rework of one or less process steps can be seen as *convergence.*
The kind of iteration called "*Repetition*" can be seen as an outlier, which cannot be classified as "small" or "large" iteration: In different phases of the process some (same) design activities are conducted to achieve different goals. This is different from all the other kinds of iterations: In the other kinds different activities are conducted repeatedly to achieve the same goals.

2.2 Design-Structure-Matrix for preventive avoidance of unnecessary iterations

After the classification of different kinds of iterations the approach towards optimised process planning supported by a DSM will be introduced. It is the goal of this approach to prevent large and unnecessary iterations and to identify potential support for the small ones. By conducting small iterations as early as possible and preventing late time consuming large iterations an optimal result is promoted and knowledge is generated as soon as possible. Moreover support for the selection of development methods shall be provided later on in the research project. The approach will be described using part of an exemplary development process. In this process the basic steps that have to be carried out in order to generate the product concept are defined, but the sequence in which to carry them out is not determined. The defined sub-steps are: "Planning energy-supply", "Planning flow of energy", "Planning flow of signals", "Defining signal processing", "Defining Energy conversion", "Defining geometrical layout", "Preselecting material", "Preselecting manufacturing method", "Rough dimensioning", "Subdividing available space", "Developing software concept".
These sub-steps have to be arranged that way, that large iterations are prevented and small iterations are supported. This is done by a DSM. The approach is based on the use of time-based DSM as described in [4]. In this DSM the influence of the sub-steps on each other is represented according to the development project on hand. Dependencies result from the dependency of one process step on the results of another step or from a high demand for communication between different steps. The project on hand is classified by certain parameters (industrial sector, risk assessment, type of product, complexity of product). This project classification will be used to address different kinds of project situations according to which the DSM will be filled differently. In this example the DSM was filled by a team of experts as an algorithm for automatically filling the DSM is not developed yet. The different kinds of situations are still to be developed. The suggestion for process planning derived from this DSM is to start with the most active element as it provides most information for the following steps. Thus downstream information flow is enabled and upstream information flow prevented [3], which would result in a large iteration. Another possibility to use this DSM is to analyse it for clusters and start these clusters as work packages in order to support short iterations by short communication cycles.

		1	2	3	4	5	6	7	8	9	10	11	Active sum	Passive sum	Activity	Criticality
1	Plan energy-supply		x	x	x	x	x	x			x		7	4	1,75	28
2	Plan flow of energy	x		x	x		x	x					5	6	0,833333333	30
3	Plan flow of signals		x				x					x	3	4	0,75	12
4	Define signal processing		x	x							x	x	4	4	1	16
5	Define Energy conversion		x				x				x		3	3	1	9
6	Define geometrical layout	x	x					x			x		4	6	0,666666667	24
7	Preselection of material						x		x	x			3	5	0,6	15
8	Rough dimensioning	x				x		x		x	x		5	3	1,666666667	15
9	Preselection of manufacturing method							x	x				2	3	0,666666667	6
10	Subdivide available space	x			x	x	x		x	x			6	5	1,2	30
11	Develop software concept		x	x	x								3	2	1,5	6
	Passive sum	4	6	4	4	3	6	5	3	3	5	2				

Figure 1. DSM displaying the influence of process steps

2.3 Benefits of the approach

This approach is meant to prevent the class of "large iterations" and support "small iterations" and thus to reduce development time and cost. By planning the sequence of process steps according to their activity a better downstream information flow is achieved. As the process steps that provide information for other process steps are carried out first, late changes can be prevented. By conducting small iterations as soon as possible at least the same knowledge can be generated as by conducting a large iteration without the disadvantages of the large iteration. By starting clustered process steps at the same time small iterations can be supported better, as co-work of the different domains is simplified.

3 CONCLUSION AND OUTLOOK

This contribution introduces a DSM-based approach towards the prevention of iterations in system design. Next step in enabling this approach will be to generate an algorithm to fill the described DSM automatically concerning the project situation. Afterwards the concept will be validated at a partner company affiliated to the research alliance this work stems from.

Acknowledgements

The projects this approach was developed in take place as part of the research alliance ForFlow consisting of six Bavarian research institutes working on the fields of engineering design and computer science collaborating with 21 companies that is promoted by the Bayerische Forschungsstiftung.

REFERENCES

[1] Krehmer, H.; Stöber, C.; Meerkamm, H.: Approach on the control of iterations in the multidisciplinary development of technical systems. In: Proceedings on the 10th International Design Conference – DESIGN 2008, D. Marjanović (Ed.), FMENA, Zagreb, 2008.

[2] Wynn, D.; Eckert, C. M.; Clarkson, P. J.: Modelling iteration in engineering design. In: Proceedings on the International Conference on Engineering Design, ICED´07, Paris, 2007.

[3] Grebici, K.; Goh, Y. M.; McMahon, C.: Uncertainty and risk reduction in engineering design embodiment process. In: Proceedings on the 10th International Design Conference – DESIGN 2008, D. Marjanović (Ed.), FMENA, Zagreb, 2008.

[4] Browning, T.: Applying the Design Structure matrix to System Decomposition and Integration Problems: A Review and New Directions. In: IEEE Transactions on Engineering management, Vol. 47 No. 3, 2001

Contact: J. Roelofsen
Technische Universität München
Institute of Product Development
Boltzmannstraße 15
85748 Garching
Germany
+49 89 289 151 54
+49 89 289 151 44
roelofsen@pe.mw.tum.de

10TH INTERNATIONAL DSM CONFERENCE

Using the Design-Structure-Matrix for the Avoidance of Unnecessary Iterations

J. Roelofsen[1]
H. Krehmer[2]
U. Lindemann[1]
H. Meerkamm[2]

[1]Institute of Product Development, Technische Universität München
[2]Chair of Engineering Design, University Erlangen Nürnberg

Technische Universität München

FOLLOW THE FLOW OF INFORMATION!
MANAGE COMPLEX SYSTEMS

Agenda

- Introduction
- Iterations
- Approach
- Benefits
- Conclusion and Outlook

Technische Universität München

Introduction

- increased application of electrical, software-based and electronic components in technical systems
- participation of many different disciplines
- growing complexity of products and development processes
- unclear definition of interfaces and requirements causes iterations in development processes, that do not contribute to product maturity

→ demand to help the designer to prevent time-consuming and cost-intensive iterations

Iterations - Causes and Influences

- causes for iterations
  - complexity of products and development processes
  - high degree of division of labor (SE, CE)
  - lack of communication
  - undetermined boundary conditions
  - faulty decisions on basis of unclear or uncertain assumptions
- influences of iterations
  - extended development period
  - difficult traceability of the development process
  - less and challenging reuse of existing solutions
  - increasing costs

Classification of Iterations

- Small iteration
  - solution cannot be found in one step
    - → iterative approach through several partial steps
      (quantitative approximation towards the best solution)
    - small iteration is necessary in engineering design and has to be supported

- Large iteration
  - change of requirements / boundary conditions
  - unclear requirements in the beginning of the development process
  - false assumptions due to unclear or uncertain data basis
  - lack of communication
    - → jumping back to prior process steps
    - → repeated passing through the hole development process

Kinds of Iterations

Small iterations
- exploration: constant alternation from synthesis to analysis
- convergence: iterative approach towards the best solution
- refinement: further refinement of secondary characteristics (product meets the requirements)
- negotiation: contributing experts form different discipines

Large iteration
- Rework: Process steps need rework to solve problems that emerge in later process steps

Outlier
- Repetition: same design activities are conducted in different phases to achieve different goals

Source: Wynn, D.; Eckert, C. M.; Clarkson, P. J.: Modelling iteration in engineering design. (ICED´07)

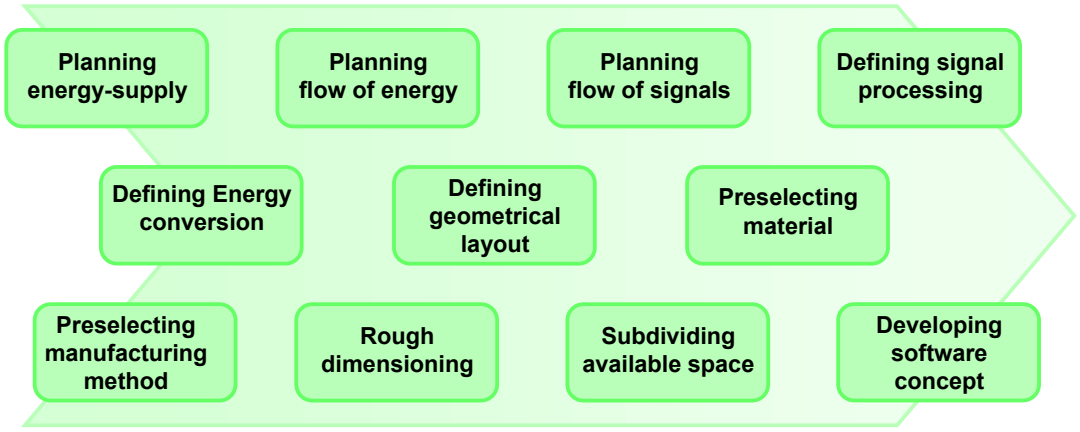
Approach

- consideration of dependencies between process steps depending on the design situation
- start with the most active process step to prevent changes in later process steps
- identify independent process steps
- identify clusters of steps with high communication-demand to start simultaneously
- communicating the results of one step with its „neighbor“ is simplified
  → large stepbacks during the development process shall be avoided
- approach shall help to „make things right“ the first time by arranging process steps sensibly

Definition of Design Situation

- parameters:
  - type of product (mechanic or mechatronic)
  - degree of novelty (new product, development of variants, changes in existing product)
  - product complexity (low, medium, high)
  - units produced (single unit, small batch, large batch, mass production)
  - customer (easy to work with – hard to work with)
  - development risk (small, medium, high)

Process Steps to Create a Design Concept



Filling the DSM

- DSM showing the dependencies between the process steps according to the design situation is filled and analysed
- an example is given for the situation:
  - type of product: mechatronic
  - degree of novelty: new product
  - product complexity: medium
  - units produced: small batch
  - customer: easy to work with
  - development risk: medium

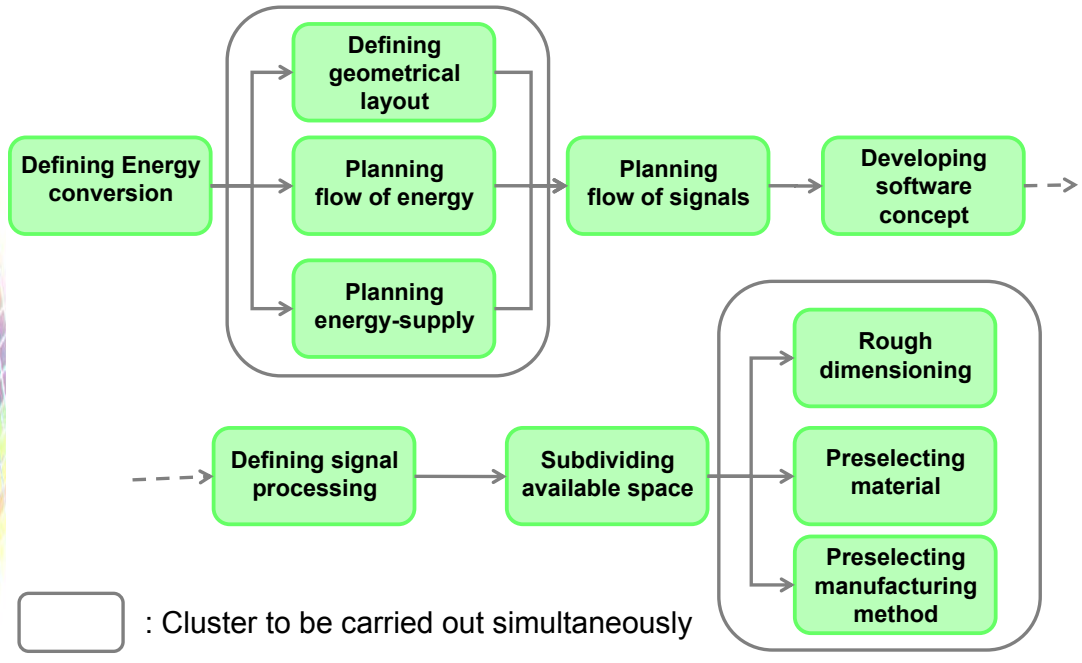
DSM

	1	2	3	4	5	6	7	8	9	10	11
1 Planning energy-supply	■	X	X	X	X	X	X			X	
2 Planning flow of energy	X	■	X	X		X	X				
3 Planning flow of signals		X	■			X					X
4 Defining signal processing		X	X	■						X	X
5 Defining Energy conversion		X			■	X				X	
6 Defining geometrical layout	X	X				■	X			X	
7 Preselecting material					X	■	X	X			
8 Preselecting manufacturing method	X				X	X	■	X	X		
9 Rough dimensioning						X	X	■			
10 Subdividing available space	X			X	X	X		X	■		
11 Developing software concept		X	X	X						■	

after triangularization

	5	6	2	1	3	11	4	10	9	7	8
5 Defining Energy conversion	■	X	X					X			
6 Defining geometrical layout		■	X	X				X		X	
2 Planning flow of energy		X	■	X	X		X			X	
1 Planning energy-supply	X	X	X	■	X		X	X		X	
3 Planning flow of signals		X	X		■	X					
11 Developing software concept			X		X	■	X				
4 Defining signal processing			X		X	X	■	X			
10 Subdividing available space	X	X		X			X	■	X		X
9 Rough dimensioning									■	X	X
7 Preselecting material		X							X	■	X
8 Preselecting manufacturing method	X			X				X	X	X	■

Derived Sequence of Process Steps



Benefits

- Benefits
  - designer is supported in the avoidance of preventable and unnecessary iterations
  - knowledge about the technical system is generated as soon as possible in the development process
  - small iterations (as are to be carried out in clustered steps) are recognized and carried out as soon as possible to prevent major fallbacks in the development process
  - reduction of development time
  - reduction of costs

Conclusion and Outlook

- approach to plan development processes according to the design situation
- next steps:
  - validate derived process sequences in a development project
  - generate an algorithm that fills the DSM automatically according to the assessment of the design situation