13TH INTERNATIONAL DEPENDENCY AND STRUCTURE MODELLING CONFERENCE, DSM'11
CAMBRIDGE, MASSACHUSETTS, USA, SEPTEMBER 14 – 15, 2011

A MORE FLEXIBLE WAY OF MODELING STRUCTURE WITH MULTIPLE DOMAINS

Sebastian Kortler, Bergen Helms, Kristina Shea and Udo Lindemann
Institute of Product Development, Technische Universität München, Germany

ABSTRACT
During the planning and development process of innovative products diverse changes of the product architecture occur. Changes are often discovered in the late phases of the product development process and lead to costly modifications of all affected parts of the system and consequently into change impacts of used models. The widely used hierarchically modeling approach is one particular limitation of common modeling techniques. Promising combinations of domains which are not considered in early modeling phases cannot easily integrated in the model. Therefore, this paper presents a more flexible modeling technique.

Keywords: MDM, flexible modeling, structural complexity management

1 INTRODUCTION

The complexity of products, development processes and organizational structures is constantly increasing. The reasons therefore are manifold, like: increasing complexity of the customers' requirements; increasing pressure of time and cost in the manufacturing and development processes; integration of multiple domains within one product, like mechanics, software, electronics or service (Lindemann, 2006; Pulm, 2004; Leimeister and Glauner, 2008; Abramovici and Schulte, 2007). During the planning and development process of innovative products, diverse changes of the product architecture occur or are inevitable. For instance, requirements' changes are often discovered in the late phases of the development process (Lindemann and Reichwald, 1998) and results into a costly modification of all affected parts of the system. The widely used hierarchically modeling approach is one particular limitation of common modeling techniques not allowing for the required degree of flexibility. Especially the initial assigning of used elements to domains and the combination of domains in early modeling phases may lead to restrictions in later analysis of the system's structure (Kohn and Lindemann, 2010). The research presented in this paper aims to tackle this challenge with a more flexible way of modeling structure with multiple domains.

This paper is structured as follows: After a review of common hierarchical modeling techniques in Section 2, the authors present the essentials of a more flexible modeling approach in Section 3. Next, a software implementation of the approach is presented in Section 4. Finally, the paper concludes with an outlook and a conclusion in Section 5.

2 COMMON MODELLING APPROACHES

Research on matrix based complexity management has come a long way. Originating from a process focus with the first published formulation of a DSM (Steward, 1981) a whole scientific community has developed around this research area. The DSM is a means to model and analyze dependencies of one single type within one single domain, e.g. geometrical adjacency relations between components. Browning (2001) classifies four types of DSMs to model different types of problems: component-, team, activity-, and parameter-based DSMs. However, many other classifications exist (e.g. in Maurer, 2007) nowadays.

Danilovic and Browning (2007) have extended DSM to DMM, i.e. Domain Mapping Matrices. The goal was to enable matrix methodology to include not just one domain at a time but to allow for the mapping between two domains, as previously postulated e.g. by Yassine (2003). Lindeman et al. (2008) have taken this approach further to model whole systems consisting of multiple domains, each having multiple elements, connected by various relationship types by combining DSM and DMM methodologies under the framework of Structural Complexity Management (StCM). The authors refer to this approach as Multiple Domain Matrix (MDM). StCM provides a five-step procedure that supports users in system definition, information acquisition, deduction of indirect dependencies, structure analysis and the interpretation of structural criteria depending on the respective system's context. During the first step of system definition, the domains of the model are selected according to the modeling hypotheses and the scope of the later analysis or according to the existing information sources (Lindeman et al., 2008). In the following steps, promising combinations of domains are derived. Finally, the structure of these combinations is analyzed. In this step lies the limitation of the hierarchical five-step procedure. If changes occur during the product development process, such as requirements' changes in the late phases of the development process, (arising of further information sources, changes of requirements or modified goals) the user needs to apply changes in all phases of the five-step procedure. For example, promising combinations of domains which are discovered in late modeling phases cannot easily be considered. Nevertheless, as especially the last DSM conferences have shown, the MDM approach integrating multiple views "domains" becomes more and more accepted to manage several perspectives onto a system.

3 THE FUNDAMENTALS OF A MORE FLEXIBLE MODELLING TECHNIQUE

The main, undermining driver of the proposed modeling approach is the possibility of modeling elements of different levels of abstraction at the same time. Thus, the user can define domains, relations between domains, relation types and relations between elements in any phase of the modeling process. Additionally, instances of the modeled components (domains, elements and relation types) shall be (re-)used in different matrices, i.e. in different contexts. But, changes of the elements in one occurrence shall be propagated via update functions to all other corresponding instances. Moreover, further combinations of domains shall get integrated at anytime.

In this way, changes which take place throughout the product development process can easily be integrated in the model. In this way, further promising combinations of domains can be considered in structural analysis.

4 A MORE FLEXIBLE MODELING APPROACH

The presented modeling tool consists of two main parts. With the palette on the right area of the modeling tool the user can select a category of components and drop it on the canvas (see Figure 1).

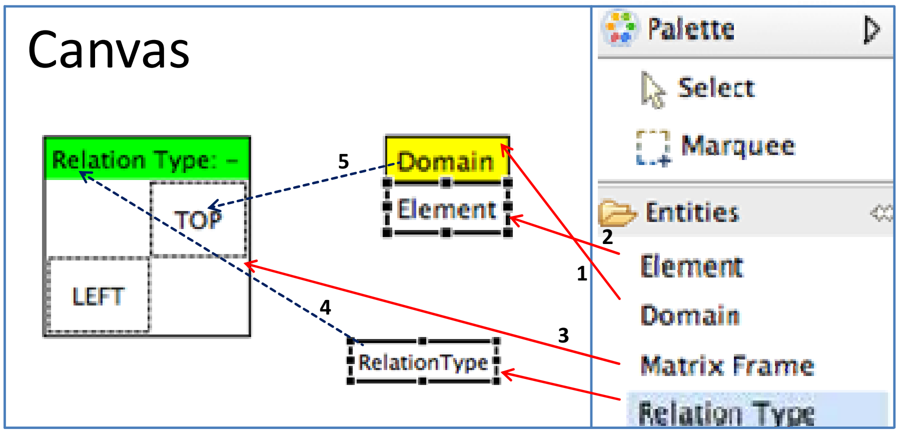
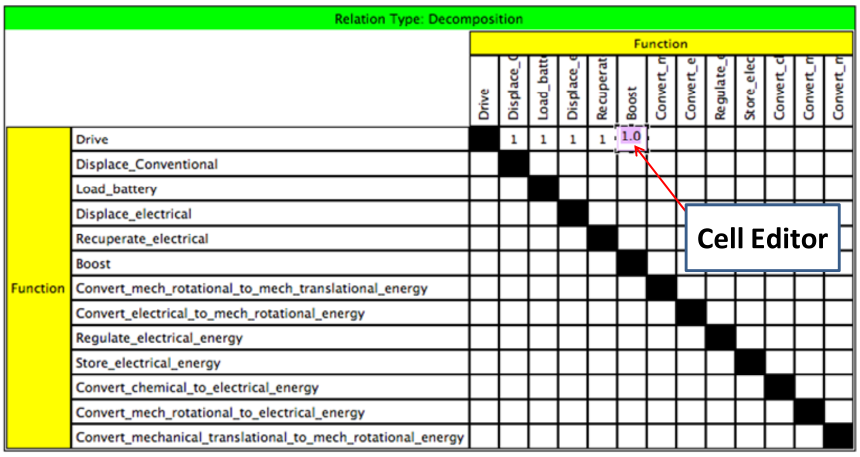


Figure 1. The main parts of the modeling tool

The first component which we will start with in this example is the domain. A domain is simply illustrated as a node that is labeled by its name. It can be selected from the palette similar to all other components and located everywhere and freely moved on the canvas (see arrow no. 1 in Figure 1). A domain consists of a number of elements. An element can be created from the palette (similar to the domain) and can be assigned to a domain afterwards. Therefore, the user can drop the element on the area of an already created domain (see arrow no. 2 in Figure 1). In order to change the name of a component, the user can double-click the component or select the component and type the name directly. Pressing the Enter key will close the small text editor, quit the editing session and update all other occurrences of this element.

The main component of the presented tool is a matrix frame. A matrix frame is a container of relations between elements of two domains and enables the user to define values for each relation between the elements of the domains. In order to create a matrix frame, the user can choose a matrix frame component from the palette and drop it on the canvas (see arrow no. 3 in Figure 1). A matrix frame area consists of three smaller parts, namely relation type, top domain and left domain. The title on the top of a matrix frame presents the type of the relation between the elements of the modeled domains. This relation type is represented as a relation type component. In analogy to previous components, a relation type is also illustrated as a node on the canvas. To define the relation type of a matrix frame, the user has to drag the relation type component on the top area of the matrix frame (see arrow no. 4 in figure 1). The title of the matrix frame changes subsequently. The user can replace this type by dropping another relation type component on the matrix frame anytime. The other two areas on the matrix frame are labeled with Top and Left and can intuitively be recognized as the dropping areas of two domains. To do so, the user can drag an available domain component from the canvas and drop it on the intended area (see arrow no. 5 in Figure 1). After dropping the second domain component the user will notice that a cell editor for the relations is created automatically at the center of the matrix frame (see Figure 2).


Relation Type: Decomposition

		Function												
		Drive	Displace_C	Load_batt	Displace_e	Recuperat	Boost	Convert_m	Convert_e	Regulate_e	Store_elec	Convert_c	Convert_m	Convert_m
Function	Drive	■	1	1	1	1	1.0							
	Displace_Conventional		■											
	Load_battery			■										
	Displace_electrical				■									
	Recuperate_electrical					■								
	Boost						■							
	Convert_mech_rotational_to_mech_translational_energy							■						
	Convert_electrical_to_mech_rotational_energy								■					
	Regulate_electrical_energy									■				
	Store_electrical_energy										■			
	Convert_chemical_to_electrical_energy											■		
	Convert_mech_rotational_to_electrical_energy												■	
	Convert_mechanical_translational_to_mech_rotational_energy													■

Figure 2. A matrix frame DSM editor with integrated domain "function" and relation type "decomposition"

Depending on which domains are dropped, a DMM cell editor or a DSM cell editor will be shown. The relation value between two elements can be edited directly by typing the value. Modifications (rename, remove and add) applied to components (domains, relation types and elements) on the canvas will lead to modifications of this component in all matrix frames that contain this modified component. This is possible since matrix frames have a number of pointers to the objects of relation type, domain and element and listen to their changes.

5 CONCLUSION AND OUTLOOK

During the planning and development process of innovative products diverse changes of the product architecture occur. Changes are often discovered in the late phases of the product development process and lead to costly modification of all affected parts of the system and consequently into change impacts of used models. The widely used hierarchically modeling approach is one particular limitation of common modeling techniques. Promising combinations of domains which are not considered in early modeling phases cannot easily integrated in the model. Therefore, the authors presented a more flexible modeling technique. This technique allows for modeling components of different level of abstraction, such as domains, relation types or elements, at the same time. Moreover, promising combinations of domains can be designed by adding a new matrix frame to the model at anytime. Furthermore, in the proposed modeling tool modifications of components are propagated to all instances of the components by the use of pointers to all occurrences. In this way the user can model system's structure in a more flexible way. Changes of the system's structure can be integrated in the model at anytime. Further and deeper analysis can be performed by considering potential combinations of domains which come up in late phases of the product development process. To do so, the user must not change the whole model. The user can easily add further views.

In future work the authors will enhance the presented modeling tool with transformation techniques. In this way the authors want to enable different people modeling on same model instances of different modeling paradigms in different times.

ACKNOWLEDGEMENTS

We thank the Deutsche Forschungsgesellschaft (DFG) for funding this research as a part of the collaborative research centre “Managing cycles in innovation processes – Integrated development of product service systems based on technical products” (SFB 768). ‘

REFERENCES

Abramovici, M. & Schulte, S. (2007). Optimising customer satisfaction by integrating the customer’s voice into product development. In: *Proceedings of International Conference on Engineering Design, ICED’07*, Paris, France.

Browning, T. (2001). Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management*, 48(3), 292-306.

Danilovic, M., Browning, T.R. (2007). Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, 25(3), 300-314.

Kohn, A., Lindemann, U. (2010). Approach towards a more flexible handling of domains in complex systems. In: *Proceedings of 12th International DSM Conference*, Cambridge, UK, Hanser.

Leimeister, J. M. & Glauner, C. (2008). *Wirtschaftsinformatik*, 50.

Lindemann, U. (2006). *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden*. Springer, Berlin.

Lindemann, U. & Reichwald, R. (1998). *Integriertes Änderungsmanagement.* Springer, Berlin.

Lindemann, U., Maurer, M. & Braun, T. (2008). *Structural Complexity Management: An Approach for the Field of Product Design*. Springer, Berlin.

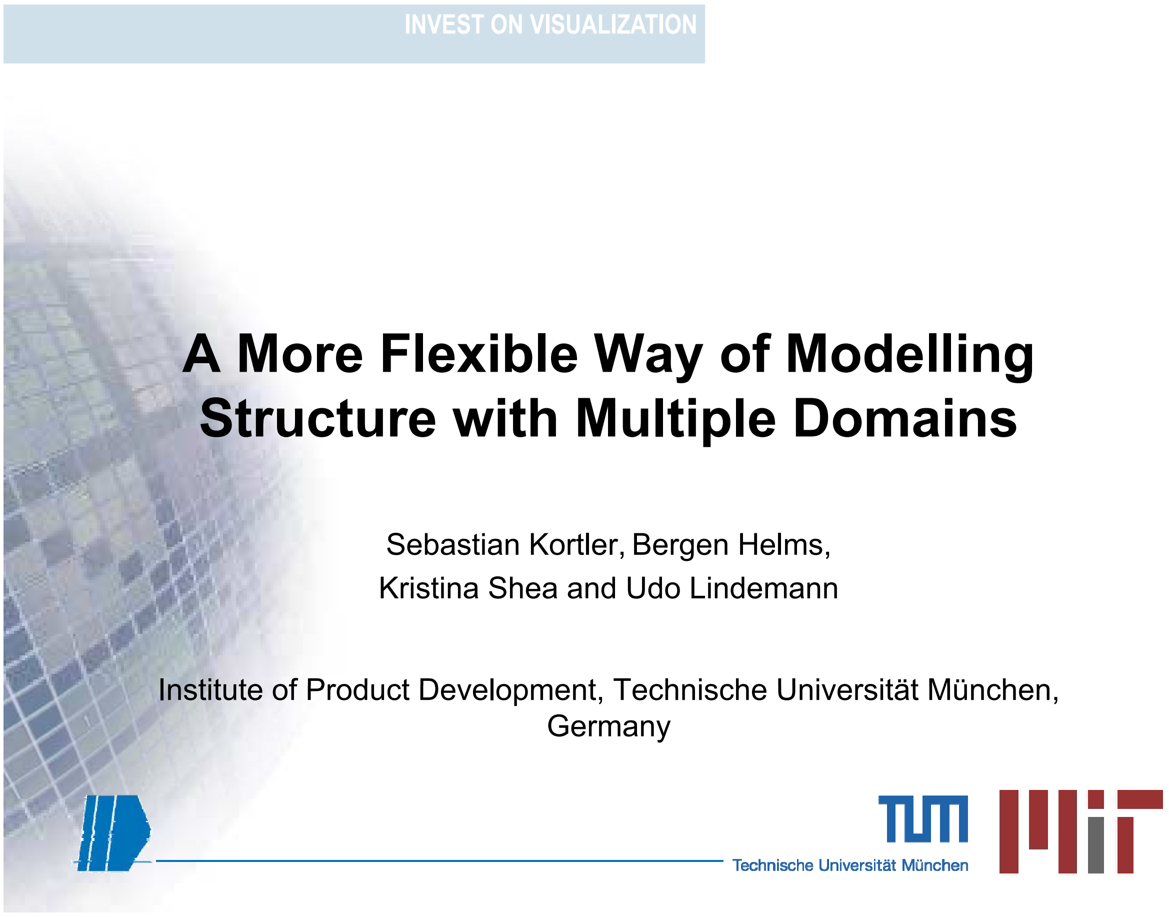
Maurer, M. (2007). Structural Awareness in Complex Product Design. Lehrstuhl für Produktentwicklung, TU München, München.

Pulm, U. (2004). Eine systemtheoretische Betrachtung der Produktentwicklung. Fakultät für Maschinenwesen, Technischen Universität München, München.

Steward, D.V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 28, 71-74.

Yassine, A., Whitney, D., Daleiden, S., Lavine, J. (2003). Connectivity maps: modeling and analysing relationships in product development processes. *Journal of Engineering Design*, 14(3), 377-394.

Contact: Sebastian Kortler
Institute of Product Development
Technische Universität München
Boltzmannstraße 15
D-85748 Garching
Germany
Phone: +49 89 289 151 53
Fax: +49 89 289 151 44
e-mail: kortler@pe.mw.tum.de
http://www.pe.mw.tum.de



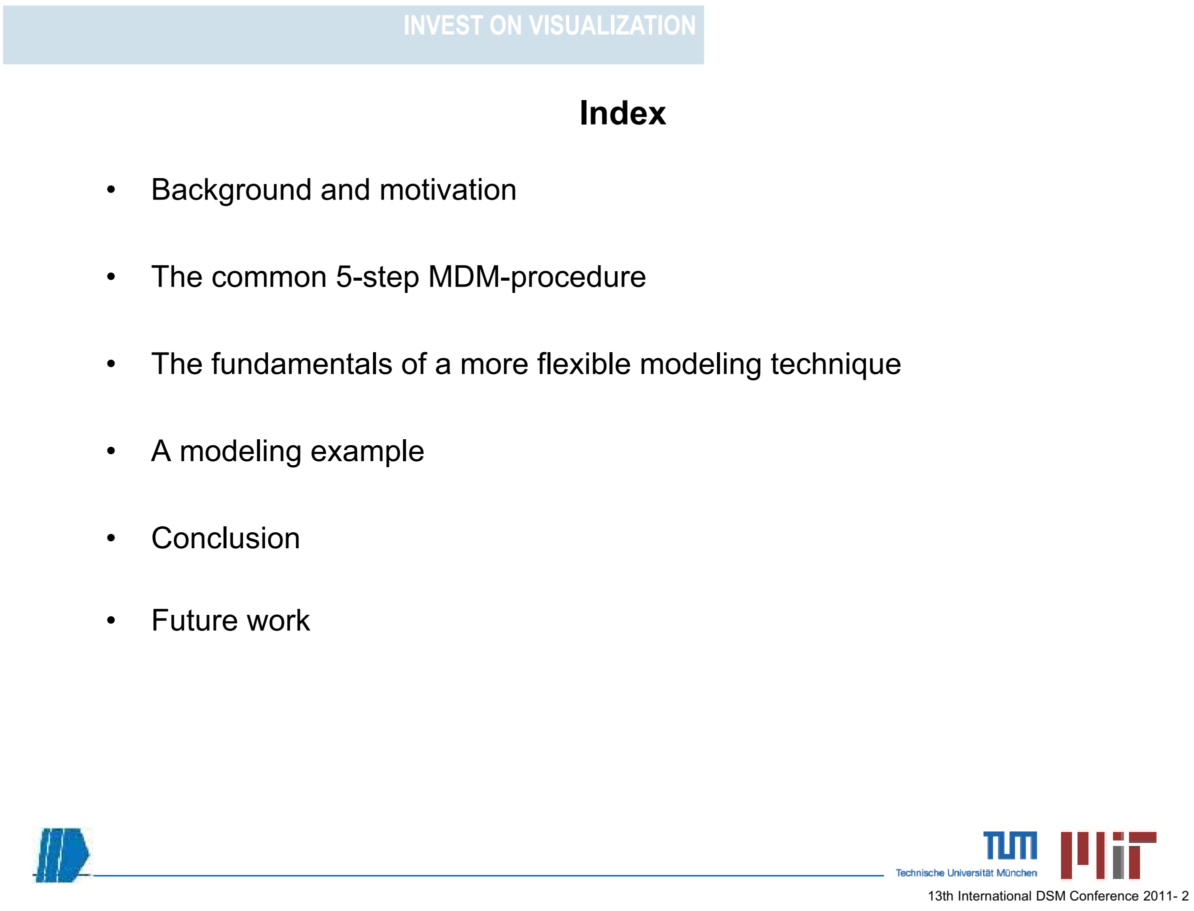
INVEST ON VISUALIZATION

A More Flexible Way of Modelling Structure with Multiple Domains

Sebastian Kortler, Bergen Helms,
Kristina Shea and Udo Lindemann

Institute of Product Development, Technische Universität München, Germany

Technische Universität München



INVEST ON VISUALIZATION

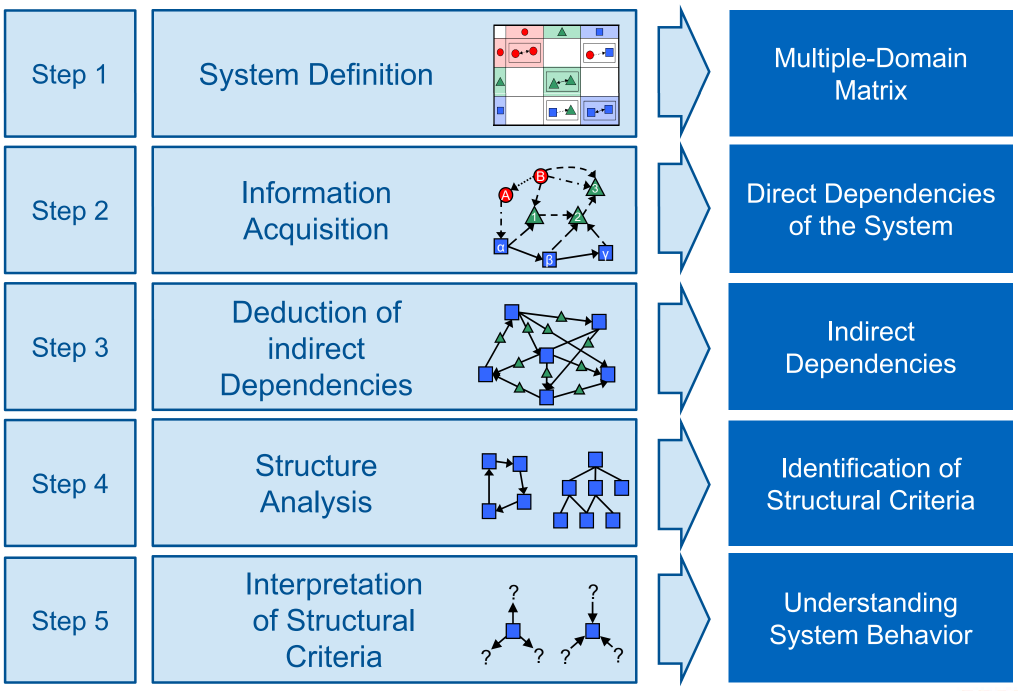
Index

Technische Universität München

Background and Motivation

- During the planning and development process of innovative products diverse changes of the product architecture occur
- Changes are often discovered in the late phases of the product development process and lead to costly modifications
- Changes of the product architecture lead to change impacts of used models
- The widely used hierarchically modeling approach is one particular limitation of common modeling techniques not allowing for the required degree of flexibility
- The initial assigning of used elements to domains and the combination of domains in early modeling phases may lead to restrictions in later analysis of the system's structure
- The research presented in this paper aims to tackle this challenge with a more flexible way of modeling structure with multiple domains

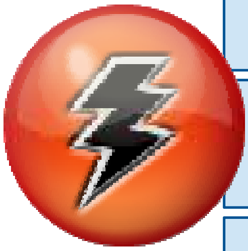
The Common 5-Step MDM-Procedure

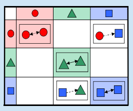
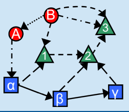
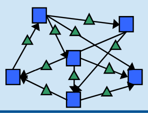

Step 1
System Definition
Multiple-Domain Matrix
Step 2
Information Acquisition
A
B
α
β
γ
1
2
3
Direct Dependencies of the System
Step 3
Deduction of indirect Dependencies
Indirect Dependencies
Step 4
Structure Analysis
Identification of Structural Criteria
Step 5
Interpretation of Structural Criteria
?
Understanding System Behavior

(Lindemann, U., Maurer, M. & Braun, T. (2008). *Structural Complexity Management: An Approach for the Field of Product Design*, Springer, Berlin)

The Common 5-Step MDM-Procedure

- If changes occur during the product development process, such as requirements' changes in the late phases of the development process, the user needs to apply changes in all phases of the five-step procedure



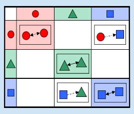
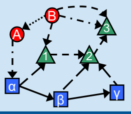
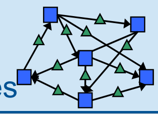
Step 1	System Definition 
Step 2	Information Acquisition 
Step 3	Deduction of indirect Dependencies 

- A more flexible modeling approach would be useful

The Fundamentals of a More Flexible Modeling Technique

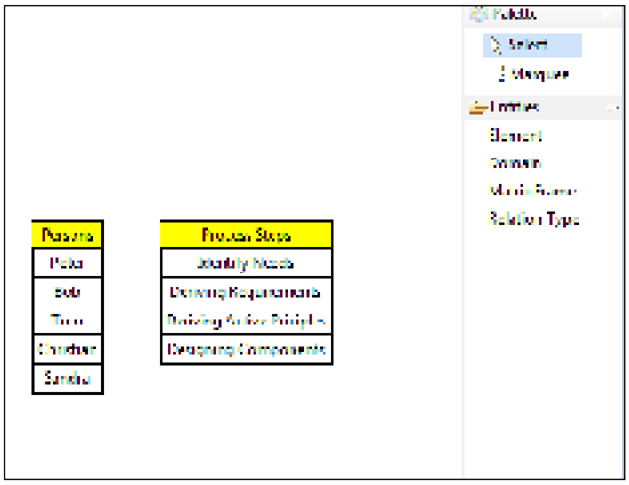
- Offering the possibility to model elements of different levels of abstraction at the same time
  - Defining domains, relations between domains, relation types and relations between elements independently during the whole modeling process
  - Instances of the modeled components (domains, elements and relation types) shall be (re-)used in different matrices, i.e. in different contexts
  - Changes of the elements in one occurrence shall be propagated via update functions to all other corresponding instances
  - Further combinations of domains shall get integrated at anytime

Performing all the modeling steps at the same time

System Definition 	Information Acquisition 	Deduction of indirect Dependencies 

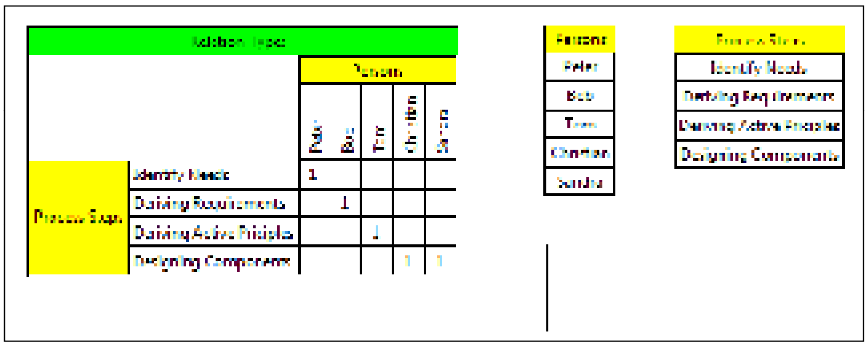
A Modeling Example

- In the first step of this example model the designer started with:
  - The domain persons and process steps
  - 5 persons
  - 4 process steps



A Modeling Example

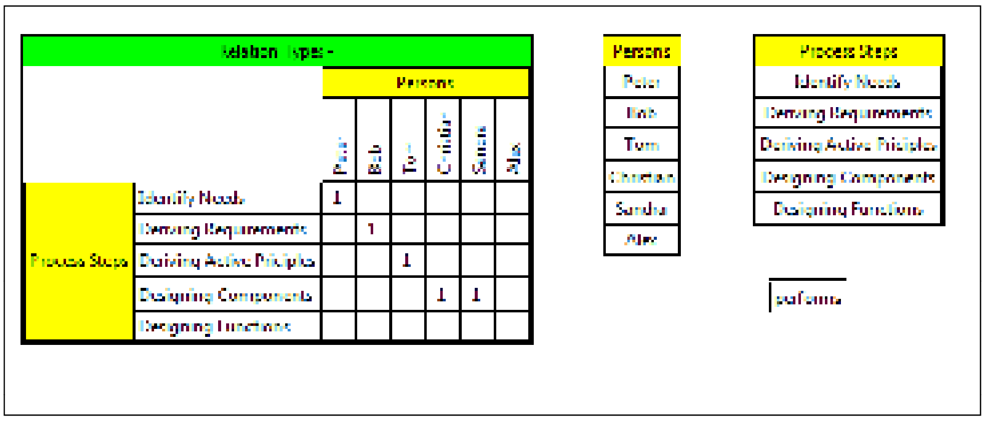
- In the second step the designer performed some updates:
  - a matrix frame connecting process steps and persons
  - 5 dependencies between process steps and persons



Relation Types		Persons				
		Peter	Bob	Tom	Christian	Sandra
Process Steps	Identify Needs	1				
	Deriving Requirements		1			
	Deriving Active Principles			1		
	Designing Components				1	1

A Modeling Example

- In the third step the designer performed some updates:
  - a further person (Alex) – the matrix frame is updated
  - a further process step (Designing Functions)) – the matrix frame is updated
  - 1 relation type (performs)



Relation Types -							
		Persons					
		Peter	Bob	Tom	Christian	Sandra	Alex
Process Steps	Identify Needs	1					
	Deriving Requirements		1				
	Deriving Active Principles			1			
	Designing Components				1	1	
	Designing Functions						

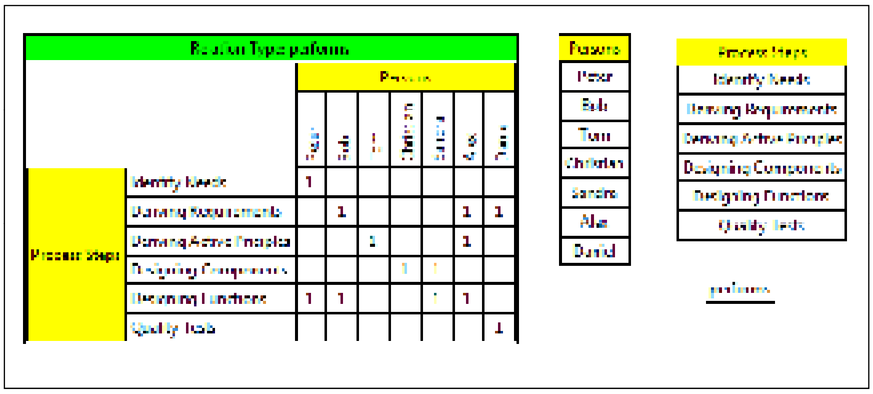
Persons
Peter
Bob
Tom
Christian
Sandra
Alex

Process Steps
Identify Needs
Deriving Requirements
Deriving Active Principles
Designing Components
Designing Functions

performs

A Modeling Example

- In the fourth step the designer performed some updates:
  - a further person (Daniel) – the matrix frame is updated
  - a further process step (Quality Tests) – the matrix frame is updated
  - 8 dependencies between process steps and persons
  - the matrix frame is connected to relation type (performs)



Relation Type performs								
		Persons						
		Peter	Bob	Tom	Christian	Sandra	Alex	Daniel
Process Steps	Identify Needs	1						
	Deriving Requirements		1				1	1
	Deriving Active Principles			1			1	
	Designing Components				1	1		
	Designing Functions	1	1			1	1	
	Quality Tests							1

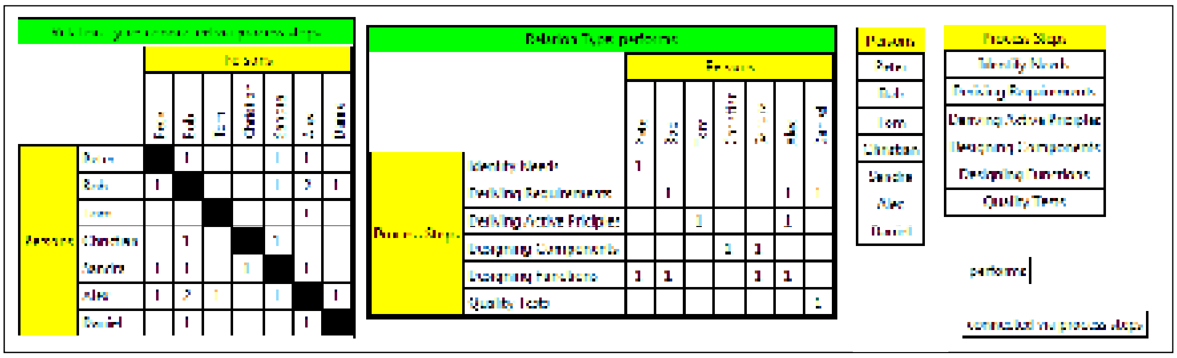
Persons
Peter
Bob
Tom
Christian
Sandra
Alex
Daniel

Process Steps
Identify Needs
Deriving Requirements
Deriving Active Principles
Designing Components
Designing Functions
Quality Tests

performs

A Modeling Example

- In the fifth step the designer performed some updates:
  - a further matrix frame connecting (Persons, Persons)
  - a further relation type (connected via process steps)
  - the new matrix frame is connected to relation type (connected via process steps)
  - indirect dependencies (persons, persons)



Relation Type connected via process steps

		Persons						
		Peter	Bob	Tom	Christian	Sandra	Alex	Daniel
Persons	Peter	■	1			1	1	
	Bob	1	■			1	2	1
	Tom			■			1	
	Christian		1		■	1		
	Sandra	1	1		1	■	1	
	Alex	1	2	1		1	■	1
	Daniel		1				1	■

Relation Type performs

		Persons						
		Peter	Bob	Tom	Christian	Sandra	Alex	Daniel
Process Steps	Identify Needs	1						
	Defining Requirements		1				1	1
	Defining Active Principles			1			1	
	Designing Components				1	1		
	Designing Functions	1	1			1	1	
	Quality Tests							1

Persons
Peter
Bob
Tom
Christian
Sandra
Alex
Daniel

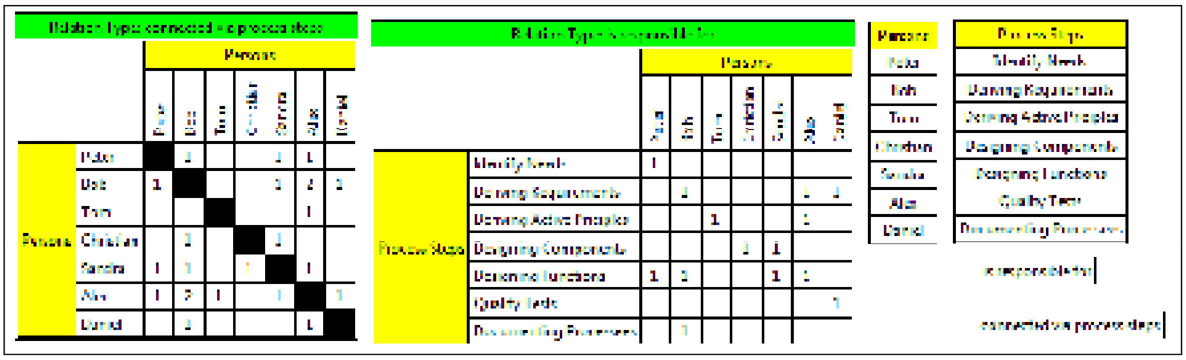
Process Steps
Identify Needs
Defining Requirements
Defining Active Principles
Designing Components
Designing Functions
Quality Tests

performs

connected via process steps

A Modeling Example

- In the sixth step the designer performed some updates:
  - a further process step (Documenting Processes) – the matrix frames are updated
  - 1 dependency between process steps and persons
  - renaming of relation type (performs) -> (is responsible for) – the matrix frames are updated



Relation Type connected via process steps

		Persons						
		Peter	Bob	Tom	Christian	Sandra	Alex	Daniel
Persons	Peter	■	1			1	1	
	Bob	1	■			1	2	1
	Tom			■			1	
	Christian		1		■	1		
	Sandra	1	1		1	■	1	
	Alex	1	2	1		1	■	1
	Daniel		1				1	■

Relation Type is responsible for

		Persons						
		Peter	Bob	Tom	Christian	Sandra	Alex	Daniel
Process Steps	Identify Needs	1						
	Defining Requirements		1				1	1
	Defining Active Principles			1			1	
	Designing Components				1	1		
	Designing Functions	1	1			1	1	
	Quality Tests							1
	Documenting Processes		1					

Persons
Peter
Bob
Tom
Christian
Sandra
Alex
Daniel

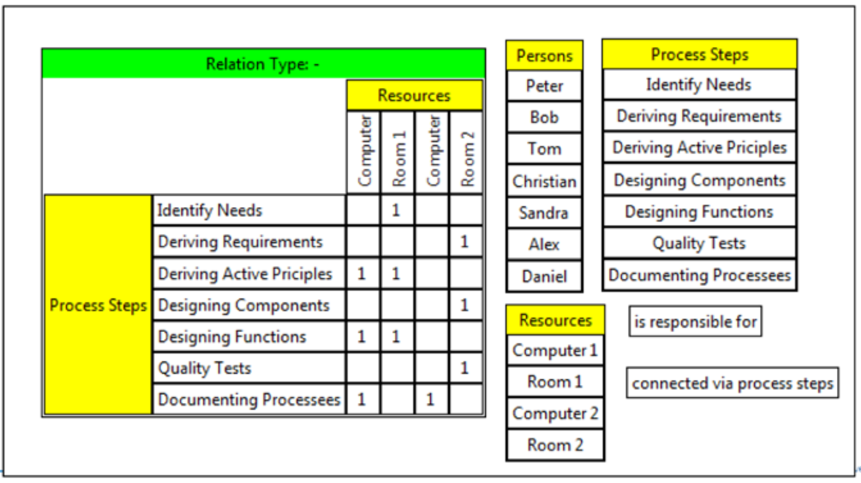
Process Steps
Identify Needs
Defining Requirements
Defining Active Principles
Designing Components
Designing Functions
Quality Tests
Documenting Processes

is responsible for

connected via process steps

A Modeling Example

- In the seventh step the designer performed some updates:
  - a further domain (Resources) with 4 elements
  - a further matrix frame (Process Steps, Resources)
  - 10 dependencies



Relation Type: -

		Resources			
		Computer	Room 1	Computer	Room 2
Process Steps	Identify Needs		1		
	Deriving Requirements				1
	Deriving Active Priciples	1	1		
	Designing Components				1
	Designing Functions	1	1		
	Quality Tests				1
	Documenting Processees	1		1	

Persons
Peter
Bob
Tom
Christian
Sandra
Alex
Daniel

Process Steps
Identify Needs
Deriving Requirements
Deriving Active Priciples
Designing Components
Designing Functions
Quality Tests
Documenting Processees

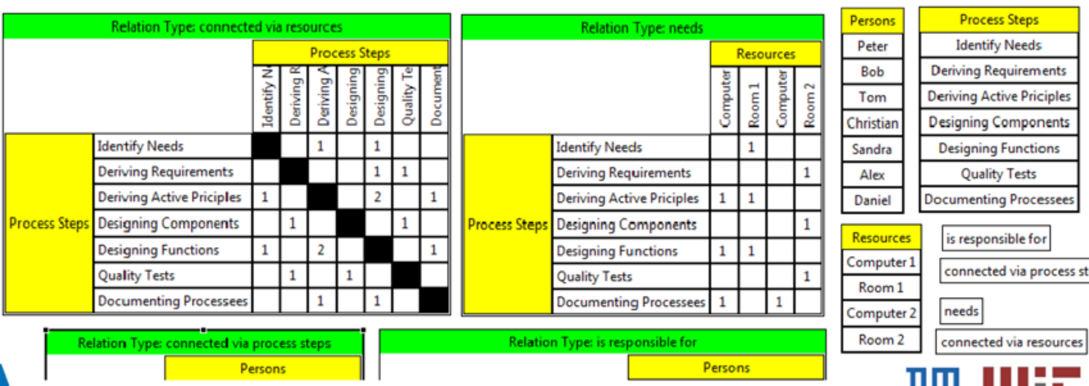
Resources
Computer 1
Room 1
Computer 2
Room 2

is responsible for

connected via process steps

A Modeling Example

- In the eight step the designer performed some updates:
  - 2 further relation type (needs, connected via resources)
  - the matrix frame (process steps, resources) is connected to relation type (needs)
  - the matrix frame (process steps, process steps) is connected to relation type (connected via resources)
  - indirect dependencies (process steps, process steps)



Relation Type: connected via resources

		Process Steps						
		Identify N	Deriving R	Deriving A	Designing	Designing	Quality Te	Document
Process Steps	Identify Needs	■		1		1		
	Deriving Requirements		■			1	1	
	Deriving Active Priciples	1		■		2		1
	Designing Components		1		■		1	
	Designing Functions	1		2		■		1
	Quality Tests		1		1		■	
	Documenting Processees			1		1		■

Relation Type: needs

		Resources			
		Computer	Room 1	Computer	Room 2
Process Steps	Identify Needs		1		
	Deriving Requirements				1
	Deriving Active Priciples	1	1		
	Designing Components				1
	Designing Functions	1	1		
	Quality Tests				1
	Documenting Processees	1		1	

Relation Type: connected via process steps

Persons

Relation Type: is responsible for

Persons

Persons
Peter
Bob
Tom
Christian
Sandra
Alex
Daniel

Process Steps
Identify Needs
Deriving Requirements
Deriving Active Priciples
Designing Components
Designing Functions
Quality Tests
Documenting Processees

Resources
Computer 1
Room 1
Computer 2
Room 2

is responsible for

connected via process steps

needs

connected via resources