INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

EXTENDING MULTIPLE DOMAIN MATRICES TO ALLOW FOR THE MODELING OF BOOLEAN OPERATORS IN PROCESS MODELS

Matthias Kreimeyer, Stefanie Braun, Matthias Gürtler, Udo Lindemann
Institute of Product Development, Technische Universität München

ABSTRACT

This paper suggests an approach to combine the analytical advantages of matrix-based notation with the modeling capabilities of graphical notation in order to model process flows including logic operators. Matrix-based notation is exemplified by Design Structure Matrices (DSM) and Multiple Domain Matrices (MDM), graphical notation by Event driven Process Chains (EPC), both being established standards in industry.
DSM and MDM offer algorithmic methods for analyses, but so far without a means of modeling decision points (AND, OR, XOR). EPC allows for an easy modeling of process flows with parallel and alternative paths. However, it does not feature comprehensive analyses, making it difficult to systematically analyze a process model.
In this research, MDM was therefore extended to include logic operators, thus combining the two approaches to transfer EPC and similar models into a matrix-based description and vice versa; this makes any graphical model that uses Boolean logic accessible to algorithmic analysis available in MDM. Generally spoken, the proposed modeling scheme opens up a way of generic modeling of logic operations in matrix-based notations.

Keywords: Boolean operator, Design Structure Matrix (DSM), Multiple Domain Matrix (MDM), process model

1 INTRODUCTION

With growing market demands, technical systems become more and more complex. This is both true for the products themselves as for the organizations that bring such products to the market. To support this development, systems engineering has been introduced a long time ago. With it, many methods of engineering complex systems have become available. Among them, the Design Structure Matrix (DSM) [1], introduced to better plan processes, has turned out to be a suitable tool for planning, modelling and analyzing all kinds of architectures [2]. In particular, this is the case because DSM allows for a systematic approach in both analysis and synthesis based on sound algorithmic support.

1.1 A Challenge to Model Complex Systems using Matrices

However, such matrices still face the challenge of being unable to model logic operators such as AND, OR and XOR [3], although these are commonly used in process flowcharts. For a process, this would e.g. refer to parallel or alternative process paths. In turn, logic operators in Design Structure Matrices could enable engineers to widen their modelling approach while being able to use classic DSM-based analysis methodology. So far, only binary matrices (representing the existence or non-existence of a dependency) or numerical matrices (introducing a weight to characterize the dependency in terms of its importance or probability) are available (see table 1 later in this paper).

1.2 Goals of this Research and Requirements

To overcome the disadvantage of not being able to transfer a common process model that includes logical operators into a matrix-based description without losing the information on these operators, the goal of this research was to find a way of extending matrix-based notation to include logic operators within process modelling. The focus was put on design processes, as these depend largely on decision points; furthermore, classic “flow-oriented” process models have made good progress on logics modelling. However, the authors believe that the basic description should also enable the better modelling of e.g. product architectures, although this field of application has not been further investigated in this research.

As a modelling standard for processes, Event-driven Process Chains (EPC) [4] were selected, because they are very similar to many other process modelling methodologies, allowing thus a comprehensive application of the extended modelling scheme that was developed. In fact, EPC embodies many features that can be found in other process modelling languages, while it is also well-formalized. This enables the development of a methodology that can be transferred to other process models easily.

As a matrix-based notation Multiple-Domain Matrices (MDM) [5] were chosen, as these represent a generic case of both DSM and other possible matrices, providing thus the maximum vocabulary that is available when using design matrices. All the while, there has been extensive research on the application of MDM, thus making it a generic tool that can be adopted for various kinds of applications.

In summary, the main requirements to the new solution are thus:

- Compatibility to existing MDM-notation
- Integration of logic operators
- Unambiguous description of processes

1.3 Research Methodology and structure of this paper

This paper is based on experiments on parts of real processes. Using these process models, in a first step, Elementary Building Blocks (EBB) consisting of the basic logic operations were modelled in EPC and transferred into MDM-notation. Then, single EBBs were combined to model complete processes. In the end, the validation on a complex real process served to test the applicability of the approach proposed.

This paper is structured accordingly: To motivate this research, first, a state of the art is given, explaining matrix methods and the prevailing gap in logic modelling. Equally, process modelling and its ability to include logic operators is explained; finally, existing ways of converting processes with such operators into matrices are lined out. In the following, a solution of how to model logic operators in matrices to better describe processes is generated. The paper concludes with a study of an industrial case of a design process.

2 STATE OF THE ART

2.1 Matrix-based systems modeling

Originating from a process focus with the first publication [1] of a Design Structure Matrix (DSM)[2], a whole community has developed around this research. The DSM is able to model and analyze dependencies of one single relationship type (e.g. “process step A activates process step B”) within one single domain (e.g. process steps) [2]. Thus, such matrices are necessarily square. There are numerous algorithms to analyze the overall structure of the relationships within a DSM [5]. DSM later was extended to Domain Mapping Matrices (DMM) [6]. The goal was to include more than one domain at a time by allowing for the mapping between two domains, including analyses [7] thereof. Thus, such matrices are rectangular.

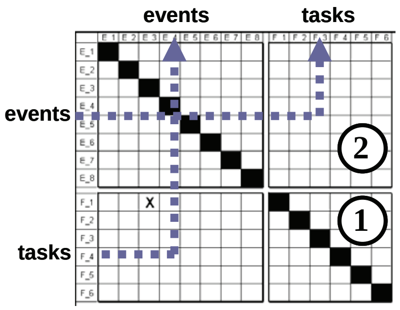


Figure 1: DSM, DMM and MDM: example of an EPC process

Multiple-Domain Matrices (MDM) [5] combine DSM and DMM to form MDMs as a way of systematically describing and analysing a whole system, including multiple domains and several relationship types concurrently. This way, a complete system can be described through partial DSMs and DMMs. Thus, both kinds of matrices are a subset of MDM. Also, MDM allows for systematic application of all available algorithms by compiling several matrices into aggregated views [5] [8].
Figure 1 illustrates the concept of a MDM. It shows how a MDM basically is a DSM [8] with several DSMs along its diagonal (e.g. (1) in figure 1) and DMMs outside the diagonal (e.g. (2) in figure 1). Additionally, each place of a DSM or DMM can be filled with several matrices at a time, allowing for different types of relationships in a system (e.g. flow of energy and forces). Every matrix is read as "row activates column".

2.2 Including information on the dependencies within a MDM

Unfortunately, matrices can only represent the existence of edges between nodes (i.e. the rows/columns). There are two types of such matrices – binary and numerical ones. Binary matrices only model the existence of a dependency, e.g. by an "x" in a cell, and they do not further differentiate the quality of the dependency [9]. Numerical matrices attribute a weight to each dependency, typically by assigning a value from 0 to 100%, or as e.g. a five-point scale to quantify the strength of each dependency as suggested e.g. by [10]. In fact, most matrices that serve as input for simulation are numerical, e.g. in Signposting [11].
Further detailing a system element is relatively easy using a DMM. In such a case, the domain to be explained is simply related to another domain using a DMM. Figure 2 shows how system element A is linked to attribute 1.
To relate an attribute to an edge in MDM notation, [3] suggested a way to model attributes to a dependency as an independent domain, and then relate two nodes via this third 'attribute'-domain. Transferred to a process, this would enrich e.g. the dependency (= edge) between two tasks (= nodes) with additional information. Such a piece of information could, of course, also be the logic operation that takes place in between two or more nodes. Figure 2 shows this as attribute 2, which is linked *not* to another system element *but* to the dependency between two system elements.

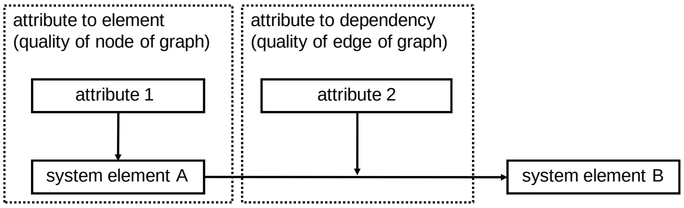


Figure 2: Attributes to elements and to dependencies according to [3]

2.3 Process modeling with EPC/ARIS

Event-driven Process Chains (EPC) [4] are a common process description in European industry, with at least 6000 licenses in place [12]. It is, in many aspects, similar to other process models such as SADT, IDEF or Petri Nets [13]. The basic elements of the 'primary workflow' are events, tasks and decision points, constituting a bipartite graph interspersed with Boolean operators. Figure 3 shows the basic semantics of EPC with a focus on logic operators: The process is a recombination of alternating sequences of events and tasks, between which logic operators can be inserted to describe the dynamics of a workflow where necessary. Other static elements, which are not of further interest to this paper, can complement tasks with additional information, e.g. data-objects, IT-resources and organisational units [4].

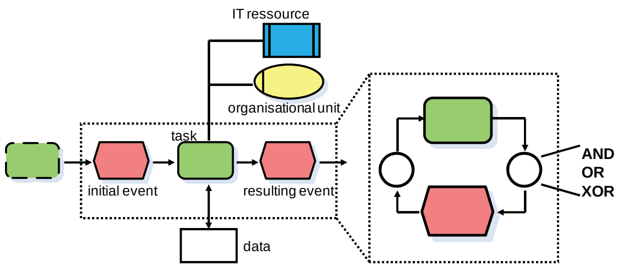


Figure 3: EPC process metamodel

An *event* describes an entered status which controls and influences the following process flow. On the one hand, events are triggers for functions. On the other hand, they are the result of a precedent function. *Tasks* are a neutral characterisation of an intended change of state or action of an object. *Connectors* are used if an event has several following functions or a function has different events as results. A connector represents a decision point and can be of the types AND, OR, and XOR. As shown in figure 4, *Join*-Connectors bring several process paths together. *Split*-Connectors divide the process into different paths [14]. This implies that one side of the logical connector always has only one incident or outgoing edge; n:m relationships need to be modelled using at least on join- and one split-connector, therefore.

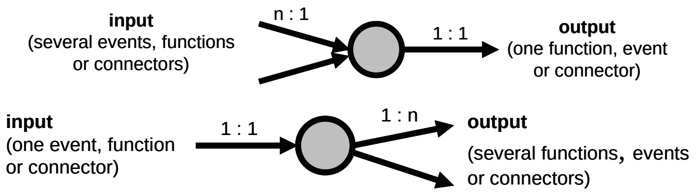

input
(several events, functions or connectors)
n : 1
1 : 1
output
(one function, event or connector)
input
(one event, function or connector)
1 : 1
1 : n
output
(several functions, events or connectors)

Figure 4: Join- (above) and split-connector (below)

There are a number of Boolean operators; the operators AND, OR, and NOT are the basic constituents [15] that can be used to set up all possible combinations, among others XOR [16]. However, in EPC and other process models, the following are commonly applied to model decision points [14]:

- **AND** (logical conjunction): $Y = X_1 \wedge X_2$
  Join: all incident edges are required to continue / Split: all outgoing edges have to be followed
- **OR** (logical disjunction): $Y = X_1 \vee X_2$
  Join: activated by at least one incident edge / Split: one or more outgoing edges are to be followed
- **XOR** (exclusive disjunction): $Y = (X_1 \wedge n\_X_2) \vee (n\_X_1 \wedge X_2)$
  Join: one incident edge is allowed at a time / Split: exactly one outgoing edge has to be followed at a time

The following semantic rules apply with EPCs [4] and guide the development of the approach presented in this paper:

- Each EPC starts and ends with an event.
- The sequence event-task-event-… has to be observed. Any number of connectors can be inserted.
- Process paths have to be split and joined with the same type of connector.
- Connectors have either the relation 1:n or n:1.
- Events are a special kind of status and cannot make a decision; therefore OR- and XOR-connectors are not allowed after an event.

In practical application, these rules are often not adhered to; therefore, one additional requirement on the solution presented here is robustness towards non-compliance of individual rules.

2.4 Converting Processes Logics into Matrices

So far, there has been little research into converting processes comprising these operators into matrix descriptions [17, 18]. Table 1 sums up all conversion rules with their advantages, disadvantages and recommended adaptations.

Table 1: Converting logic operators (process to matrix description)

Rule	Concept	Advantages	Disadvantages	U se for…
1	• Creating alternate process matrices	• Consistent notation with simple matrices • Characteristics of structure are kept	• Many different networks are generated • Complex conversion	• Simple models • Models with few logic operators • High relevance of logic operators
2	• Neglecting the operators	• Structure easy to analyze • Sufficient for many applications	• Neglects dynamic information • Structural analysis impossible for decision points	• Big and complex models • Dynamic information less important
3	• Decisions as probabilities for related paths	• Detailed description of model quality • Model is good for simulation	• Needs run-time information • Results highly susceptible to changes in data	• Numerical optimization of decisions • Limited focus on general structure
4	• Logic operators as additional entity in DSM	• No algorithmic conversion necessary	• Limited analysis of logic operators possible	• High number of entities in matrix • Analysis difficult
5	• Logic operators modeled as additional domain with "type of" attribute	• Conversion back to flow-chart possible • Model for basic analysis (without operators) and extended analysis (with operators)	• Complex MDM is generated	• Extensive use of logical operators • Combination of several process models into one • Impact analysis of decision points

Overall, five conversions are possible that transfer the operators in different manners:

Rule 1: Resolve all logical connections [17]

Logical operators are eliminated by creating different graphs and matrices for each alternate process given by each decision in the process. Because of the large number of different matrices eventually obtained, this rule is only of theoretical interest, while its application is little practicable. The number n of all possible graphs amounts to $n = 3k * 2m$ with k the number of binary OR-operators and m the number of binary XORs.

Rule 2: Neglect the operators [18]

By dropping all decision points and turning their connections into simple edges, only the basic structure of the process remains. This way, flow characteristics can be analyzed, while e.g. a critical path across different decision points (Critical Path Method) cannot be looked at. Thus, only analyses that are based on pure structural characteristics, i.e. those that do not rely on decision points, are possible.

Rule 3: Translate operators into probabilities

By resolving all possible paths into or after a decision point as numerical values that correlate to the probability for taking each path, it is possible to evaluate the sequence of decisions that take place numerically. As such, the decision points are basically modeled like a Bayesian network. However, the appropriate numerical data (e.g. as a numerical DSM) is necessary, which often is not the case.

Rule 4: Logical operators as additional entities in the process domain

The operators are kept as an additional entity, losing information on the type of operator. Nevertheless, the operators lose their meaning, and only the pure existence of a relationship is transferred, as if all operators were AND operators. This approach extends the simple disregard of the operators as in rule 3 and integrates an additional number of entities into the network that can be analyzed using common methodology. A process network with entities will therefore grow to a network with entities with as the number of distinct logic operators. The approach mainly is useful if the process model only consists of a single DSM and if decision points are of little importance.

Rule 5: Carry along the logical operators and their characteristics

Extending rule 4, this approach (explained in the following) extends the process MDM by a new domain that models the existence and connectivity of connectors (i.e. connectors are modeled as nodes of a new domain “connector”) and that uses another additional domain to model the type of the connector (i.e. each connector node is attributed with its type using a DMM). Although rather complex in both execution and result, the resulting matrices can transfer the structure of any process model without loss of information bi-directionally.

In this paper, rule 5 is developed and detailed in the following to extend the modeling capabilities to not convert but keep the logic operators.

3 MDM-BASED MODELLING OF LOGIC OPERATORS

3.1 Approach

The directed flow of information of an EPC model represents a bipartite graph with nodes of two domains (events and tasks) [4]. These dependencies can be converted into a MDM. As already illustrated in figure 1, events in rows start new tasks in columns, and tasks in rows result in events in columns. Such a matrix (being similar to the EPC) furthermore allows to be enriched by additional information, e.g. other dependencies or further domains (e.g. the organizational structure), allowing for a more complete model [5]. E.g. [19] shows a comprehensive way of modelling various aspects of a design process in MDM notation.

To introduce logical operators, the approach of the Three Domain Concept as shown in [3] is used. Decision points are handled as an additional domain; the nodes of this domain serve as an intermediate connection between events and tasks. To characterise the type of each of these connectors, each node that represents a connector is attributed with its type (AND, OR, XOR) using an additional DMM (the “characteristic” domain). Figure 5 explains how the two domains of the primary workflow “tasks” and “events” form the topmost two domains in the according Multiple-Domain Matrix. Then, the connectors are added as an additional domain. They represent a different kind of node present in the EPC model, thus introducing a new domain. Ultimately, the type of the node is inserted as an additional fourth domain, which, in fact, only adds a “type of” attribute to each connector node.

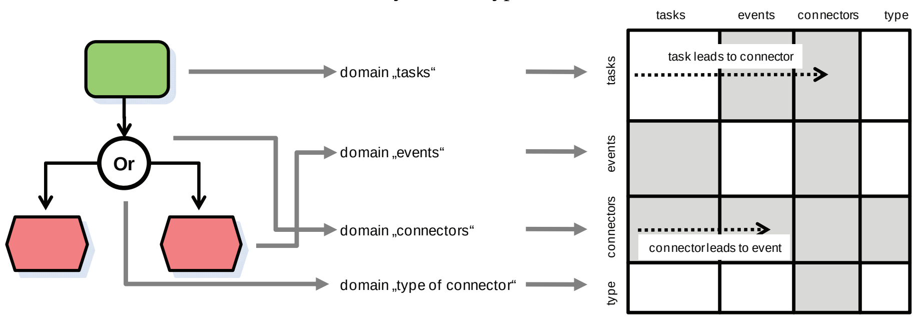


Figure 5: MDM with appropriate domains to translate primary workflow of EPC model (domains that are potentially not empty are shaded)

Theoretically, all 16 matrices within the emerging MDM could contain entries representing dependencies. If, however, the EPC model that is used as a basis is semantically correct, only the shaded matrices in the MDM are actually needed while the others remain empty. Yet, it is often possible that process models in practice do not fully adhere to all rules that are set in the process meta-model. Therefore, dependencies can also occur in the empty matrices, e.g. if tasks are directly linked. The following section explains the possible cases that can occur in detail.

3.2 Elementary Building Blocks

Elementary Building Blocks (EBB) embody the smallest units of a process, similar to workflow patterns [20]. Interlinking them results in the entire process. Representing the elementary connection types, they consist of two process elements (either tasks or events), and one logical connector.
Altogether, there are ten possible EBBs: six for tasks with AND, OR, XOR for splits and joins. Events have just four EBBs, as splits following an event can only be of the type AND, because events cannot take decisions: in strict EPC notation, events only represent a static state of the process that does not process any information and is thus unable to decide if a process flow follows one path or another [4]. However, the notation developed here would also be able to represent cases with decision points such as OR and XOR after an event.
Figure 6 and 7 illustrate two of the ten EBBs: an XOR-join EBB for events and an AND-split EBB for tasks. All other EBBs are formed likewise. Each EBB is first displayed on the left hand side in EPC and then transferred into MDM-notation on the right hand side.

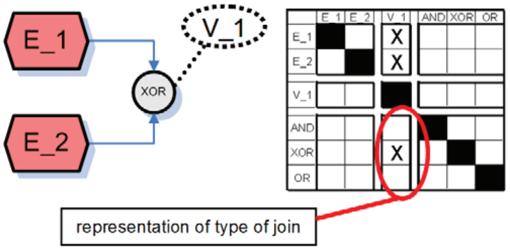


Figure 6: Example for EBB of XOR-join

Figure 6 shows one of the three possible join-connectors for tasks. All joins have an equal structure in EPC as well as in MDM. Both events E_1 and E_2 are followed by connector V_1. In EPC, this is expressed by the directed edges. Accordingly, the MDM is read as “row activates column”.
While the different operators are displayed by different objects in EPC, in MDM they are modelled by different entries within the characteristic domain. In other words, the characteristic domain puts a “type of” attribute to each connector-node. This means that the relation between two nodes (e.g. nodes E_1 and E_2 to the following node F_1, which is not shown in figure 6) that is established by the connector is separated from the type of relation (e.g. the XOR in figure 6).

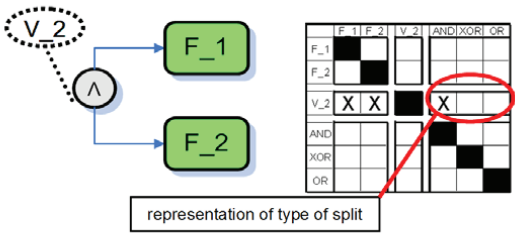


Figure 7: Example for EBB of AND-split

Figure 7 similarly shows a split-connector. The connector V_2 has task F_1 and task F_2 as successors. Thus, F_1 and F_2 have V_1 as precedent node. In MDM, the split-connector is recognisable by the entries within one row in the characteristic domain.

3.3 Recombining EBBs to describe a complete process

As an example for a possible recombination of EBBs, the case of one join- and one split-EBB is considered. As illustrated in figure 8, event E_1 and event E_2 are connected via a XOR-join-operator V_1, which has the AND-split-connector V_2 as successor. V_2 splits the process flow into two paths with functions F_1 and F_2.

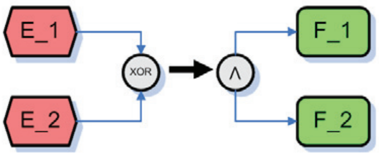


Figure 8: Recombination of join- and split-connector

Figure 9 displays the process in MDM-notation. The fact that V_1 has more than one entry in the column shows that it is a join-connector. The entries in the row of V_2 point to a split-connector. Similarly to the EBBs in figures 6 and 7, the directed flow of information can be recognized from the dependencies (“row to column”).

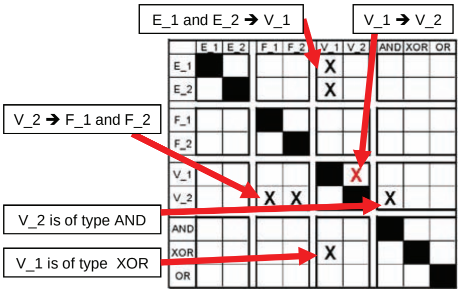


Figure 9: Process from previous figure in MDM-notation

4 INDUSTRIAL EXAMPLE

To explain the notation, a part of a design process of an automobile manufacturer was modelled (the overall process is explained e.g. in [21, 22]). It contains several logical connectors and typical features of a design process, e.g. parallel activities, different kinds of decisions, a rework iteration, and different input and output objects. In the case presented, EPC was used as a modelling scheme; however, the events were used to represent different business objects that serve as input and output for the tasks that constitute the process. This adaptation of EPC is commonly found in an industrial context, as it enables easier reading and reduces the number of entities that have to be modelled. The case study was used to test the suitability of the modelling scheme for an industrial model. Also, the model was then used for the application of DSM based analyses. Results are found in detail in [22].

The process that is used as an example represents the collaboration between embodiment design and simulation during the development of a premium class sedan. Basically, it shows how a simulation model for vibrational analysis of the body-in-white is set up, run through simulation, and the results are checked to be optimized or integrated into the overall design.

The following list gives a more detailed insight into the actual procedure taken. To remain compliant with the notation used in the paper, tasks are named F_x, and business objects are handled like events, named E_y. The process is graphically represented in figure 10.

1. The process starts when the events (E_1 and E_2) or E_3 or E_7 (circle) appear; here, different possible inputs are selected to create a simulation model from either a predecessor model (E_1), a scanned clay model (E_2), a CAD assembly (E_3) or a previous simulation model (E_7).
2. In the next step both task F_1 (set-up of simulation geometry model) and F_2 (preparation of boundary conditions for simulation) are executed. While F_1 just generates E_6, task F_2 can generate E_4 (modified input file) or E_5 (new simulation file).
3. Function F_3 is induced when E_6 and (E_4 or E_5) eventuate. F_3 (simulation) has either E_7 (simulation to be optimized) or E_8 (sufficient results) as an output.

4. In case of E_7, the process jumps back to V_2.
5. In case of E_8, task F_4 is performed (integrate results into design of automotive body).
6. E_9 is the final result of the process (optimal design).

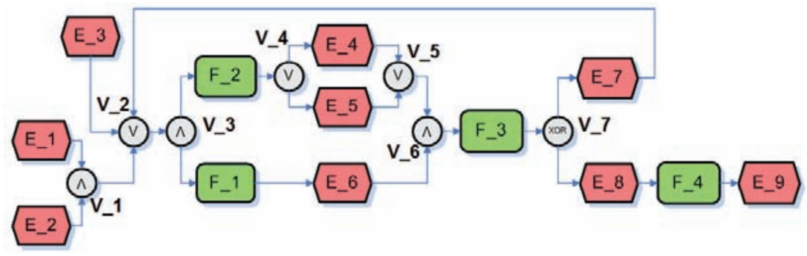
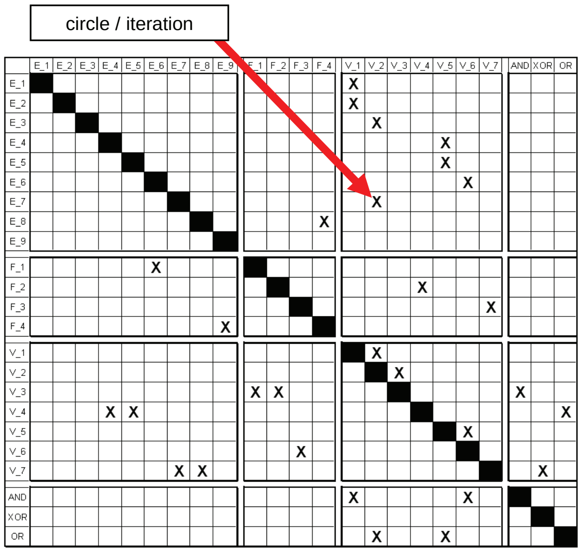


Figure 10: Industrial example (part of larger process)

Using the EBBs that were developed, the process description was translated into MDM notation, shown in figure 11. The arrow marks the point where the circle in the process flow is entered. As can be seen, the nine business objects form a 9x9 DSM that remains empty because none of them are connected directly to each other. The same is true for the 4x4 DSM for the tasks F_1 through F_4. However, the DMMs between tasks and events contain the dependencies for the process flow across E_8, F_4 and E_9. The connector-DSM is not empty because connectors V_1, V_2 and V_3 are directly related among themselves, combining two different joins and one split. The same is true for V_5 and V_6. Ultimately, as the connectors are related to both incident and outgoing events and tasks, these DMMs contain the majority of dependencies.


circle / iteration

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	F_1	F_2	F_3	F_4	V_1	V_2	V_3	V_4	V_5	V_6	V_7	AND	XOR	OR
E_1	■													X									
E_2		■												X									
E_3			■												X								
E_4				■														X					
E_5					■													X					
E_6						■													X				
E_7							■								X								
E_8								■					X										
E_9									■														
F_1						X				■													
F_2											■						X						
F_3												■								X			
F_4									X				■										
V_1														■	X								
V_2															■	X							
V_3										X	X					■					X		
V_4				X	X												■						X
V_5																		■	X				
V_6												X							■				
V_7							X	X												■		X	
AND														X					X		■		
XOR																						■	
OR															X			X					■

Figure 11: Example in MDM-notation

With the overall matrix description, all algorithms applicable for DSM/DMM/MDM analysis are possible. Different analyses become possible, now. For example, the empty event-DSM can be computed to show the dependencies between the events, even though these exist only in an indirect manner across the tasks and connectors, as suggested by [5]. Equally, the model is now accessible to analyses using e.g. structural metrics to identify various weak spots or to further determine the structural characteristics of the process (e.g. robustness, redundancies, central entities, drivers for cycles, propagation of errors and more, see [23]). The results of an analysis that closely regards the decision points can be found in [22].

5 CONCLUSION

This paper provides an approach for an improved matrix based-notation to solve deficits in modelling logic operators. It uses Multiple-Domain Matrices (MDM) that form an overall framework based on Design Structure Matrices (DSM) and Multiple Domain Matrices (DMM). The approach integrates logic operators such as splits or joins as a new domain, keeping the type of the operator (AND, OR, XOR) as an additional attribute in a characteristic domain. This way, it is compliant to the "classic" matrix-based description of systems, thus extending them to carry along logical operators (c.f. requirement 1).

In the presented case of describing directed graphs used for process modelling, the extension of MDM by logic operators allows for more complete process reviews. While typical flow-oriented process models only have limited means of systematic analysis, an MDM allows for integrating additional information as well as methodical analysis using various algorithms. The approach therefore supports process management by making process models better accessible to systematic analysis (c.f. requirement 2).

While MDM and EPC are rather different forms of notation, it was shown that their content is compatible in terms of the structure of the process, i.e. the interaction and dependencies of the various entities that prevail in the process. With the solution suggested here, a direct interconversion between both notations is therefore possible. This could support a consolidation in the form of clearly defined interface between the modelling methods. Thus, the goal is not to create one single notation, but rather an extension of the still existing ones with well-defined interfaces.

The definiteness of the approach shown allows to model iterations and other structural characteristics commonly found in processes. It is, in fact, fully suitable to represent complex process unambiguously. As long as all process entities are uniquely named (and this must be done also for all logical operators), no ambiguous cases can occur (c.f. requirement 3).

Also, as could be shown, the matrix-based description is flexible enough to also model dependencies that are not originally intended by the EPC process specification, making the conversion robust also to non-compliances in the flow model. While this is most useful for "dirty" models in industry, it also makes it possible to quickly detect any modelling errors by simply checking for dependencies that are "not supposed to be there".

As the approach that was presented is fully compliant with the existing MDM notation, all algorithms or metrics that are available to describe complex systems can be used. For example, the activity or passivity of any entity can be easily computed [22] to show how tightly an entity (e.g. a task or an event) or a logical operator is embedded into the process. Also, metrics that describe decision points, e.g. McCabe complexity or Log-based complexity [14], can be computed to evaluate the decisions that drive the process.

As algorithms remain applicable, so does their interpretation; however, when certain patterns in a process are interpreted while using logical operators, the type of these operators need to be closely examined. Figure 11 shows how to seemingly similar structural characteristics of a process may have different implications due to two different logical operators.

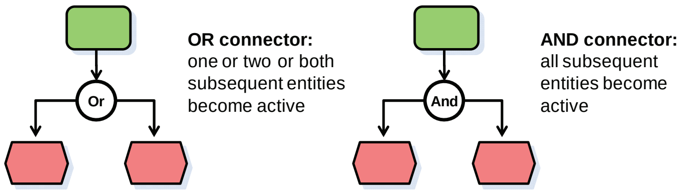

Or
OR connector:
one or two or both subsequent entities become active
And
AND connector:
all subsequent entities become active

Figure 11: Two similar structures with different operators necessitate different interpretations

While the approach has turned out to be sufficient for process modelling, the question whether this form is suitable to model logics in general matrix descriptions needs to be further investigated. As it allows representing decision points, a possibly other field of application could be the synthesis of variants within a product architecture. This as well as further case studies is part of the current work being undertaken.

REFERENCES

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

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

[3] Kreimeyer, M., Braun, S., Gürtler, M. and Lindemann, U. Relating Two Domains Via A Third – An Approach To Overcome Ambiguous Attributions Using Multiple Domain Matrices. *ASME 2008 IDETC/CIE 2008* (American Society of Mechanical Engineers (ASME), Brooklyn, New York, 2008).

[4] Scheer, A.-W. *Business Process Engineering, Reference Models for Industrial Enterprises*. (Springer, Berlin, 1994).

[5] Maurer, M. *Structural Awareness in Complex Product Design*. (Dr.-Hut, Munich, 2007).

[6] Danilovic, M. and Browning, T.R. Managing Complex Product Development Projects with Design Structure Matrices and Domain Mapping Matrices. *6th International Design Structure Matrix Workshop*Cambridge, UK, 2004).

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

[8] Biedermann, W. and Lindemann, U. Cycles in the Multiple-Domain Matrix – Interpretation and Applications. In Kreimeyer, M., Lindemann, U. and Danilovic, M., eds. *10th International DSM Conference* (Hanser, Stockholm, 2008).

[9] Warfield, J.N. Binary matrices in system modeling. *IEEE Transactions on Systems, Man, and Cybernetics*, 1973, 3, 441-449.

[10] Pimmler, T.U. and Eppinger, S.D. Integration Analysis of Product Decompositions. *ASME Design Theory and Methodology Conference*, pp. 343-351 (ASME, Minneapolis, USA, 1994).

[11] Wynn, D.C. *Model-based approaches to support process improvement in complex product development*. (University of Cambridge, Cambridge, UK, 2007).

[12] Davis, R. and Brabänder, E. *ARIS Design Platform: Getting Started with BPM*. (Springer, London, 2007).

[13] Langer, S., Kreimeyer, M., Müller, P., Lindemann, U. and Blessing, L. Entwicklungsprozesse hybrider Leistungsbündel – Evaluierung von Modellierungsmethoden unter Berücksichtigung zyklischer Einflussfaktoren. In Thomas, O. and Nüttgens, M., eds. *Dienstleistungsmodellierung,* pp. 71-87 (Physica, Berlin, 2009).

[14] Gruhn, V. and Laue, R. Complexity Metrics for Business Process Models. *9th International Conf. on Business Information Systems*Klagenfurt, Austria, 2006).

[15] Pahl, G., Beitz, W., Feldhusen, J. and Grote, K. *Konstruktionslehre*. (Springer, Berlin, 2004).

[16] IEC 60617-12 (International Electrotechnical Commission, Geneva, 2005).

[17] Belhe, U. and Kusiak, A. Modeling Relationships Among Design Activities. *ASME Journal of Mechanical Design*, 1996, 118(4), 454-460.

[18] Kreimeyer, M., Eichinger, M. and Lindemann, U. Assessment Of Process Networks Using Graph And Network Theory Based Key Figures. In Van Velden, D., ed. *FUBUTEC*, pp. 13-19 (Eurosis-ETI, Delft, The Netherlands, 2007).

[19] König, C., Kreimeyer, M. and Braun, T. Multiple-Domain Matrices as a Framework for Systematic Process Analysis. In Kreimeyer, M., Lindemann, U. and Danilovic, M., eds. *10th International DSM Conference* (Hanser, Stockholm, 2008).

[20] van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. and Barros, A.P. Workflow Patterns. *Distributed and Parallel Databases*, 2003, 14(1), 5-51.

[21] Kreimeyer, M., Eichinger, M. and Lindemann, U. Aligning Multiple Domains of Design Processes. *International Conference on Engineering Design, ICED'07*Paris, 2007).

[22] Kreimeyer, M., Gürtler, M. and Lindemann, U. Structural metrics for decision points within Multiple-Domain Matrices representing EPC processes. *2008 IEEE International Conference on Industrial Engineering and Engineering Management* (IEEE, Singapore, 2008).

[23] Kreimeyer, M., König, C. and Braun, T. Structural Metrics to Assess Processes. In Kreimeyer, M., Lindemann, U. and Danilovic, M., eds. *10th International DSM Conference,* pp. 245-259 (Hanser, Stockholm, 2008).

Contact: M. Kreimeyer
Institute of Product Development, Technische Universität München
Boltzmannstr. 15
85748 Garching
Germany
Phone	+49.89.28915136
Fax	+49.89.28915144
E-mail	matthias.kreimeyer@pe.mw.tum.de

Matthias Kreimeyer is a scientific assistant at the Institute of Product Development at the Technische Universität München, Germany. His research is focused on structural complexity and the management of engineering design processes; as part of his work, he is co-chair of the Special Interest Group “Managing Structural Complexity” and has organized the DSM Conference since 2007.

After finishing her studies in Mechatronics in May 2005 Stefanie C. Braun started to work as scientific assistant at the Institute of Product Development of the Technische Universität München. There she is occupied with a research project on optimized target costing of complex mechatronic products together with other projects that focus on the improvement of multidisciplinary design processes.

Matthias Gürtler is a graduate student at the Institute of Product Development, pursuing his diploma in mechanical engineering. Over a long period of time during his studies, he supported the research at the institute as student worker and has co-authored three papers on this topic so far.

Udo Lindemann is a full professor at the Technische Universität München, Germany, and has been the head of the Institute of Product Development since 1995, having published several books and papers on engineering design. He is committed in multiple institutions, among others as president of the Design Society and as an active member of the German Academy of Science and Engineering.