SPECIFYING PROCESS ACTIVITIES FOR MULTI-DOMAIN MATRIX ANALYSIS USING A STRUCTURED TEXTUAL FORMAT

Knippenberg, S.C.M.; Etman, L.F.P.; Wilschut, T.; van de Mortel-Fronczak, J. A.

Eindhoven University of Technology

ABSTRACT

This paper proposes a method to automatically generate a multi-domain matrix (MDM) from textual activity specifications. The format for specifying these activities is based on a structured grammar derived from natural language and consists of two types of activities: goal activities and transformation activities. A goal activity describes the purpose of an action performed by an actor for the benefit of another actor in the system. A transformation activity describes an activity from the viewpoint of a single actor, who receives, generates, and outputs information or artifacts. If one describes activities using these two types of activity specifications, dependencies can be automatically derived between actors, activities, and parameters of the system and visualized in an MDM. Thus the generated MDM presents an organization DSM (actors), a process DSM (activities), and a parameter DSM (flows of information or objects), as well as the mapping matrices coupling the different domains. An illustrative house construction example demonstrates the effectiveness of the proposed activity specification format. The method may provide an outcome in understanding and managing complex systems.

Keywords: Design methodology, Design structure matrix, Multi-domain matrix, Process modelling, Systems Engineering (SE)

Contact:
Knippenberg, S.C.M.
Eindhoven University of
Technology Mechanical Engineering
The Netherlands
s.c.m.knippenberg@tue.nl

Cite this article: Knippenberg, S.C.M., Etman, L.F.P., Wilschut, T., van de Mortel-Fronczak, J.A. (2019) 'Specifying Process Activities for Multi-Domain Matrix Analysis Using a Structured Textual Format', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.167

1 INTRODUCTION

In modern-day engineering of technical systems, many actors are involved in the design, development, and manufacturing activities. The actors usually have to perform a series of interdependent activities. Since activities usually depend on each other, management of the various activities and their precedence relations is key to arrive at an optimal sequence and clustering of activities. The coordination of the dependencies between these activities, different actors, and deliverables is key to ensure a timely and on budget delivery of the system.

One may approach the engineering design process as a business process and use established (graphical) modeling standards to this end such as Business Process Model and Notation (BPMN) (Chinosi and Trombetta, 2012), (Lee *et al.*, 2015). This is particularly useful to arrive at a well-defined and traceable design process. Nevertheless, many day-to-day design and development projects are rather informally organized using simple spreadsheets or written documents listing what activities need to be performed, who is involved, and who needs what from whom. Obviously, such documents and activity sheets are often incomplete, and also may not show a proper sequence of activities such that rework arises. An effective way to visualize and analyze the dependencies between activities is the Design Structure Matrix (DSM) method (Eppinger and Browning, 2012). Given such documents and activity sheets, developing a process DSM model typically requires a manual approach and additional interviews to gather the various dependencies.

However, if the specifications of the activities can be subjected to a predefined textual format this would support consistency and completeness regarding the flow of information between actors. What is more, this may enable automated text processing and DSM model generation. Wilschut *et al.* (2018), for example, developed a method to specify design functions following a restricted grammar in terms of goal and transformation function specifications. This allowed them to derive the various dependencies between components, functions, and parameters from the function specifications and automatically generate a multi-domain matrix (MDM) visualization, including their mutual relations. Our paper follows up on this idea, and develops a similar grammar for activities, introducing a textual format to specify goal and transformation activities. From the activities specified according to this format, an actor DSM, an activity DSM, and a parameter DSM are generated. Together with the corresponding domain mapping matrices (DMMs), an actor-activity-parameter MDM is built automatically. The proposed textual format for specifying activities textually has been validated for a couple of test problems of which one illustrating example is included in this paper.

The outline of this paper is as follows. Section 2 discusses related work regarding process modeling methods and process modeling languages. Section 3 presents our method to specify activities following a developed textual format and to automatically generate a multi-domain matrix from these specifications. An easy to understand example problem is presented in Sections 3.5 and 4 to illustrate the method. Finally, conclusions are presented in Section 5.

2 RELATED WORK

Model-Based Systems Engineering (MBSE) supports the modeling of systems during the entire life cycle (Vaneman, 2016). A popular method to model and analyze dependencies of a system is the design structure matrix (DSM). The DSM can be used to model networks in which the system elements and their interactions are presented (Eppinger and Browning, 2012), (Steward, 1981). A DSM can be used to represent different types of systems and their architectures, including product, organization, and process architectures in the various life cycle phases. Examples of these systems are racecars (Maurer, 2007), (Van Beek *et al.*, 2010), navigation locks (Wilschut *et al.*, 2018), IC design activities (Eppinger *et al.*, 1994), or project plan activities (Sharon *et al.*, 2009). One of the benefits of a DSM, compared with other network modeling methodologies, is the matrix display format that presents the complex network in a compact, easily scalable, and intuitively interpretable representation (Eppinger and Browning, 2012).

The process DSM was the first DSM method developed in de 1960s (Steward, 1965), (Eppinger and Browning, 2012). A process DSM models the network of activities and their interactions. One of the main considerations of the process DSM is the order in which the activities should be executed, for minimizing iteration steps within the entire process architecture.

By combining multiple DSMs, a multi-domain matrix (MDM) (Maurer, 2007) can be created that represents relations between different domains. In this MDM, each type of DSM is placed along the diagonal

of the MDM while the off-diagonal matrices represent the mapping between the various domains (i.e. DSM types) (Eppinger and Browning, 2012).
Building a process DSM and identifying the dependency relations between the various activities may be quite an elaborate task. In particular, for larger development processes the number of activities may grow rapidly, resulting in a DSM of considerable size.
Alternatively, a dedicated modeling language may be utilized to support the modeling of the processes and their relations, for instance the Unified Modeling Language (UML) (Vaneman, 2016), the Executable Unified Modeling Language (xUML) (Sharon *et al.*, 2009), the graphical Systems Modeling Language (SysML) (Vaneman, 2016), (Dori, 2016), (Sharon *et al.*, 2009), the Business Process Model and Notation (BPMN) standard (Chinosi and Trombetta, 2012), (Lee *et al.*, 2015), the process-based Lifecycle Modeling Language (LML) (Vaneman, 2016), and the Object-Process Language (OPL) (Dori, 2002), (Dori *et al.*, 2003a), (Sharon *et al.*, 2009). The latter OPL is a part of the Object-Process Methodology (OPM) (Dori, 2016), which combines the object-oriented (structure) and process-oriented (behavior) paradigms into a single framework. The outputs of OPM are for instance DSMs (Sharon *et al.*, 2009), (Dori, 2002), (Dori *et al.*, 2003a). An integrated systems engineering environment, designed for supporting the system development life cycle by applying OPM, is the Object-Process CASE Tool (OPCAT). Main features of OPCAT are its bimodal graphic-textual single view representation and its capability to perform simulations (Dori *et al.*, 2003b).
Our paper presents a textual format for specifying activities in a table or spreadsheet-like form which allows to automatically generate a DSM display of the dependencies between the specified activities. This enables checks for consistency and completeness of activities specified in a basic spreadsheet environment, as well as sequencing of the activities to minimize the risk of rework.

3 METHODS

The main difference between the visual process specification languages mentioned above and the approach presented in this paper is that we seek to specify the process elements textually, staying as close as possible to the commonly used written documents and activity sheets.
In this section, a textual format is presented for specifying process activities such that an MDM can be generated automatically. Activities are described in terms of actors carrying out actions. The information or artifact that is received, generated, and forwarded is represented by parameters. The basic elements of these activities are actors and parameters.
Similar to the textual format for specifying functions developed by Wilschut *et al.* (2018), the textual format for specifying activities contains verbs, parameters, and prepositions. Verbs are introduced to describe the actions that have to be performed by the actors of the system using the parameters. While components in the function specifications of Wilschut *et al.* (2018) are only capable of receiving parameters from other components of the system, actors in activity specifications can both receive as well as generate parameters by conducting activities themselves. Subsequently, an actor is able to transform the received and generated parameters into output parameters. Therefore, multiple verbs are introduced for describing actions performed on different types of parameters: a verb for receiving information or artifacts from other actors, a verb for generating a parameter by the actor itself, and a verb for transforming the aforementioned parameters into output parameters. In combination with actors and conjunctions, the activities can be specified according to a fixed format, resulting in a sentence. This sentence can either be a goal activity or a transformation activity, which are both presented in the forthcoming subsections. The textual ingredients corresponding to both types of activity specifications are listed in Table 1.

Table 1. Textual ingredients for specifying goal activities and transformation activities.

Goal activities	**Transformation activities**
Actors	Actor
Verb	Verbs
Parameters	Parameters
Preposition	Conjunction

https://doi.org/10.1017/dsi.2019.167 Published online by Cambridge University Press

3.1 Goal activities

A goal activity represents the purpose of an activity performed by an actor for another actor in the process, as presented in the textual format below.

$$Active\ actor + Verb + Parameter(s) + Preposition + Passive\ actor$$

In this format, the active actor of the goal activity indicates the actor who accomplishes the described activity and the passive actor represents the actor who receives the parameter(s) from the active actor. The verb and preposition describe the performed action, while the parameter specifies the flow between the active and passive actor.
Illustrative goal activities in the context of the development of a full electric racecar are presented in Table 2.

Table 2. Examples of goal activities, illustrating the proposed format for specifying goal activities.

ID	**Active Actor**	**Verb**	**Parameter Set**	**Prep.**	**Passive Actor**
ga-1	Battery-engineer	provides	battery-output-power	to	Inverter-engineer
ga-2	Inverter-engineer	provides	inverter-output-power	to	Motor-engineer
ga-3	Motor-engineer	provides	motor-output-power	to	Final-drive-engineer

3.2 Transformation activities

A transformation activity describes the activity performed by an actor to receive, generate, and output parameters, as presented in the format below.

$$Actor + Verb\ \#1 + Parameter(s)\ \#1 + Verb\ \#2 + Parameter(s)\ \#2 + Conjunction + Verb\ \#3 + Parameter(s)\ \#3$$

In this format, the actor executes the described transformation function. The verbs describe the action that is performed with respect to the parameters. The conjunction serves to improve the readability of the text. In this format, the first parameter set provides information about the input parameters, the second parameter set provides information about the parameters that are (internally) generated by the active actor, and the third parameter set describes the output parameters of the specified activity.
Table 3 depicts the transformation activities that correspond to the example goal activities of Table 2.

Table 3. Examples of transformation activities, illustrating the proposed format for specifying transformation activities.

ID	**Actor**	**Verb #1**	**Parameter Set #1 (input)**	**Verb #2**
ta-1	Inverter-engineer	requests	battery-output-power	consults
ta-2	Motor-engineer	requests	inverter-output-power	consults

ID	**Parameter Set #2 (generate)**	**Conj.**	**Verb #3**	**Parameter Set #3 (output)**
ta-1 (ctd.)	inverter-datasheet	and	determines	inverter-output-power
ta-2 (ctd.)	motor-datasheet	and	determines	motor-output-power

3.3 Generating the process multi-domain matrix

From the presented goal and transformation activities, an MDM can be built following the layout described by Maurer (2007), containing three DSMs and three DMMs, as visualized in Figure 1.

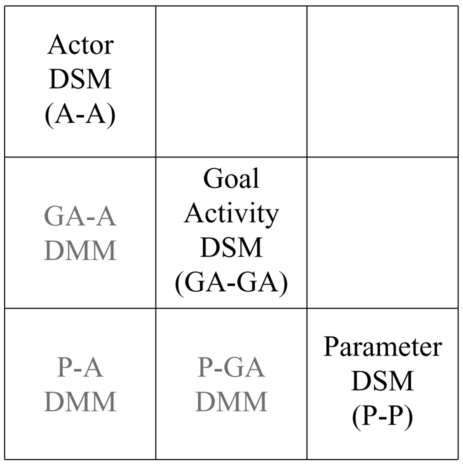

Figure 1. Schematic overview of the multi-domain matrix (MDM), derived from goal activity and transformation activity specifications.

The MDM combines the actor (A-A) DSM, the goal activity (GA-GA) DSM and the parameter (P-P) DSM in a single display, with additional mapping matrices. It can be noticed that the transformation activity (TA-TA) DSM is not displayed in the MDM since these dependencies are also covered in the P-P DSM when considering input-output relations between parameters. Therefore, it is chosen to display the P-P DSM at the cost of the TA-TA DSM with the intention to generate an MDM that is comprehensible for extensive networks, without displaying double information.

3.4 Dependencies in the process multi-domain matrix

In this section, the derivation of dependencies between actors, goal activities, transformation activities, and parameters for generating an MDM is described. In the sequel, the following conventions are used:

G	–	set of goal activities	A	–	set of actors
T	–	set of transformation activities	B	–	set of dependency types
AA	–	actor DSM (A-A)	C	–	set of parameter types
$GAGA$	–	goal activity DSM (GA-GA)	D	–	set of conjunctions
PP	–	parameter DSM (P-P)	N	–	set of parameter names
GAA	–	goal activity - actor DMM (GA-A)	P	–	set of parameters
PA	–	parameter - actor DMM (P-A)	Q	–	set of prepositions
PGA	–	parameter - goal activity DMM (P-GA)	V	–	set of verbs

From the specifications of the goal activities, dependencies between actors (A-A DSM), goal activities and actors (GA-A DMM), parameters and actors (P-A DMM), and parameters and goal activities (P-GA DMM) can be derived.

The derived dependencies are characterized by parameter p, as defined as a collection of 3 tuples (n_p, c_p, b_p) of which the formal notation is given in Equation (1), where n_p represents the name of the parameter, c_p the type of parameter, and b_p the type of dependency to which c_p belongs. Each of the dependencies in a DSM is represented with a distinct color corresponding to its dependency type b_p.

$$P = \{(n_p, c_p, b_p) \mid n_p \in N \ \wedge \ c_p \in C \ \wedge \ b_p \in B\} \qquad (1)$$

The set G of goal activities specified for the system is defined as a collection of 5 tuples (a_1, v, p, q, a_2), where a_1 represents the active actor, v the verb, p the parameter, q the preposition, and a_2 the passive actor of the specified goal activity. The formal notation is given in Equation (2).

$$G \subseteq \{(a_1, v, p, q, a_2) \mid a_1, a_2 \in A \ \wedge \ v \in V \ \wedge \ p \in P \ \wedge \ q \in Q\} \qquad (2)$$

Using the goal activity specification, a dependency between actors a_i and a_j can be identified as defined in Equation (3).

$$AA[i,j] = \{(g.3).3 \mid g \in G \ \wedge \ \left((g.1 = a_i \ \wedge \ g.5 = a_j) \ \vee \ (g.1 = a_j \ \wedge \ g.5 = a_i)\right)\} \qquad (3)$$

Even though the activity itself expresses direction, the dependency between actors is often presented to be bidirectional in the DSM, following the work of Browning (2016), which then results in a symmetric A-A DSM.
Next, the specification of goal activities can be used to derive relations between goal activities and actors in the GA-A DMM. Whether goal activity g_i has a dependency with actor a_j is defined by Equation (4).

$$GAA[i,j] = \begin{cases} \{(g_i.3).3\}, & \text{if } \; g_i.1 = a_j \;\vee\; g_i.5 = a_j \\ \emptyset, & \text{otherwise} \end{cases} \tag{4}$$

The defined relations are mapped in an asymmetric GA-A DMM, with $g_i.1 = a_j$ indicated as an active relation with a color corresponding to the type of dependency, and $g_i.5 = a_j$ marked as a passive relation indicated in grey. For instance for the goal activity ga-2 in Table 2, a relation between the two actors the Inverter-engineer and the Motor-engineer follows.
The final mapping relations that can be derived from the specified goal activities only are those between parameters and goal activities, which are placed in the asymmetric P-GA DMM. These relations are defined for any parameter $p_i \in P$ and goal activity $g_j \in G$ according to Equation (5), assigned by dependency type b_p.

$$PGA[i,j] = \begin{cases} \{p_i.3\}, & \text{if } \; g_j.3 = p_i \\ \emptyset, & \text{otherwise} \end{cases} \tag{5}$$

Furthermore, the specifications of the transformation activities represent the DSMs with dependencies between transformation activities (TA-TA DSM) and parameters (P-P DSM), respectively. These specifications can also be used for determining dependencies between parameters and actors (P-A DMM), parameters and goal activities (P-GA DMM), transformation activities (TA-TA DSM), and parameters and transformation activities (P-TA DMM).
The set T of transformation activities specified for the system is defined as a collection of 8 tuples $(a, v_1, p_1, v_2, p_2, d, v_3, p_3)$, where a represents the actor, v_1 the first verb, p_1 the first set of parameters (input), v_2 the second verb, p_2 the second set of parameters (generate), d the conjunction, v_3 the third verb, and p_3 the third set of parameters (output). The formal notation is given in Equation (6).

$$T \subseteq \{(a, v_1, p_1, v_2, p_2, d, v_3, p_3) \mid a \in A \;\wedge\; v_1, v_2, v_3 \in V \;\wedge\; p_1, p_2, p_3 \subseteq P \;\wedge\; d \in D\} \tag{6}$$

In Equation (6) three sets of parameters can be identified, of which the first set ($t.3$) represents the input parameters, the second set ($t.5$) represents the (intermediate) parameters generated for the purpose of the transformation activity, while the third set ($t.8$) indicates the output parameters of activity t. Dependencies between two parameters p_i and p_j can be derived according to Equation (7), resulting in the asymmetric parameter (P-P) DSM.

$$PP[i,j] = \begin{cases} \{p_i.3\}, & \text{if } \; \exists\, t_k, t_l \in T: \begin{array}{l} \big((t_k.3 \in p_i \;\vee\; t_k.5 \in p_i) \;\wedge\; t_l.8 \in p_j\big) \;\vee \\ \big((t_l.3 \in p_i \;\vee\; t_l.5 \in p_i) \;\wedge\; t_k.8 \in p_j\big) \end{array} \\ \{p_j.3\}, & \text{if } \; \exists\, t_k, t_l \in T: \begin{array}{l} \big((t_k.3 \in p_j \;\vee\; t_k.5 \in p_j) \;\wedge\; t_l.8 \in p_i\big) \;\vee \\ \big((t_l.3 \in p_j \;\vee\; t_l.5 \in p_j) \;\wedge\; t_k.8 \in p_i\big) \end{array} \\ \emptyset, & \text{otherwise} \end{cases} \tag{7}$$

In this equation, a dependency between parameters p_i and p_j is identified if transformation activities t_k and t_l exist in the set of specified transformation activities T, such that either (a) p_i is part of parameter sets $t_k.3$ or $t_k.5$ while p_j is part of output parameter set $t_l.8$, or (b) p_i is part of output parameter set $t_k.8$ while p_j is part of parameter sets $t_k.3$ or $t_k.5$.
By combining both the goal activity specifications and transformation activity specifications, dependencies between parameters and actors can be derived. These mapping relations are described in Equation (8) and are mapped in the asymmetric P-A DMM. For any actor $a_j \in A$ and parameter $p_i \in P$ holds that:

$$PA[i,j] = \begin{array}{l} \{(g.3).3 \mid g \in G \wedge \left(g.1 = a_j \vee g.5 = a_j\right) \wedge g.3 = p_i\} \cup \\ \{p_i.3 \mid t \in T \wedge t.1 = a_j \wedge (t.3 = p_i \vee t.5 = p_i \vee t.8 = p_i)\} \end{array} \tag{8}$$

This condition holds, i.e. parameter p_i has a mapping relation with actor a_j, if a goal activity g exists in the set of all specified goal activities G in which either the active ($g.1$) or the passive ($g.5$) actor is equal to a_j and parameter $g.3$ is equal to p_i, and if a transformation activity t exists in the set of all specified transformation activities T in which actor $t.1$ is equal to a_j and parameter p_i is part of one of the parameter sets $t.3$, $t.5$, or $t.8$ respectively.
Furthermore, a dependency with type b_p can be derived for describing dependencies between goal activities g_i and g_j, which is presented in Equation (9).

$$GAGA[i,j] = \begin{cases} \{(g_i.3).3\}, & \text{if } \exists\, t \in T:\ g_i.5 = g_j.1 \wedge (g_i.3 = t.3 \vee g_i.3 = t.5) \wedge g_j.3 = t.8 \\ \{\left(g_j.3\right).3\}, & \text{if } \exists\, t \in T:\ g_i.1 = g_j.5 \wedge \left(g_j.3 = t.3 \vee g_j.3 = t.5\right) \wedge g_i.3 = t.8 \\ \emptyset, & \text{otherwise} \end{cases} \tag{9}$$

If one of the conditional statements evaluates true, a relation between goal activities g_i and g_j, both in the set of all specified goal activities G, is identified. The derived relations between specified goal activities are mapped in the asymmetric GA-GA DSM.

3.5 Generating an MDM from specified activities

To demonstrate how an MDM can be generated from goal and transformation activities, as an illustrative example a fragment of a house construction project is presented in Tables 4 and 5. Table 4 depicts the goal activities of this example and Table 5 presents the transformation activities corresponding to these goal activities. In this fragment, the manager of the project provides designs for the foundation and the garden of the house. Using these designs, the foundation and the garden are delivered.

Table 4. A few goal activities of the house construction project example, illustrating the proposed format for specifying transformation activities.

ID	**Active Actor**	**Verb**	**Parameter Set**	**Prep.**	**Passive Actor**
ga-1	Project-manager	provides	foundation-design	to	Foundation-worker
ga-2	Project-manager	provides	garden-design	to	Gardener
ga-3	Foundation-worker	provides	foundation	to	Gardener
ga-4	Foundation-worker	provides	foundation	to	Project-manager
ga-5	Gardener	provides	garden	to	Project-manager

Table 5. A few transformation activities of the house construction project example, illustrating the proposed format for specifying transformation activities.

ID	**Actor**	**Verb #1**	**Parameter Set #1 (input)**	**Verb #2**
ta-1	Project-manager	requests	maximum-budget, house-design	determines
ta-2	Project-manager	requests	maximum-budget, house-design	determines
ta-3	Foundation-worker	requests	foundation-design	-
ta-4	Gardener	requests	foundation, garden-design	-

ID	**Parameter Set #2 (generate)**	**Conj.**	**Verb #3**	**Parameter Set #3 (output)**
ta-1 (ctd.)	foundation-budget	and	delivers	foundation-design
ta-2 (ctd.)	garden-design	and	delivers	garden-design
ta-3 (ctd.)	-	and	delivers	foundation
ta-4 (ctd.)	-	and	delivers	garden

https://doi.org/10.1017/dsi.2019.167 Published online by Cambridge University Press

A multi-domain matrix (MDM) display can be generated from the specified goal and transformation activities in Tables 4 and 5, which is visualized in Figure 2 below. This MDM represents the dependencies derived from the activity specifications, consisting of an actor DSM, a goal activity DSM, a parameter DSM, and the three corresponding DMMs. The MDM should be interpreted such that the column elements of the matrix are input to the row elements, which is indicated by the directed arrow in the top-left corner of Figure 2. The involved dependency types are presented in the legend next to the figure and represent financial information (dark blue), design information (mid blue), realized objects (light blue), and passive dependencies (grey) between elements of the system.

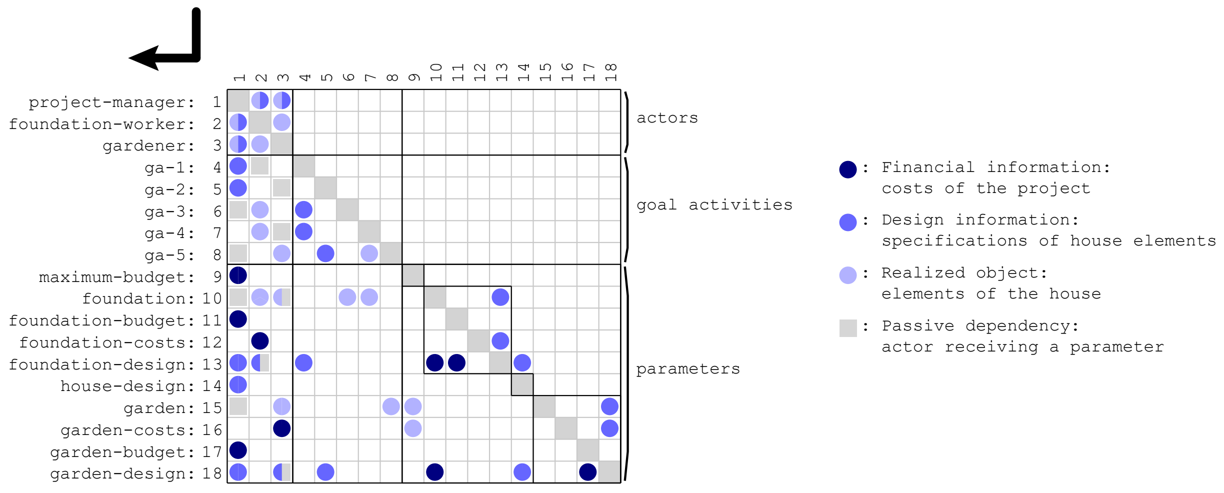

Figure 2. Multi-domain matrix of the fragment of the house construction project fragment, corresponding to Tables 4 and 5.

A clustering algorithm (we used Wilschut *et al.* (2017)) is applied to both the actor and the parameter DSM. This algorithm is able to group elements of the DSMs with multiple mutual dependencies and to detect "bus" elements. These define a group of elements of the system which have a large number of dependencies with other system elements. In Figure 2 no bus element is identified since each actor has dependencies with all other system actors.

In the goal activity DSM, four dependencies between goal activities are displayed. Goal activities ga-3 and ga-4 require design information from goal activity ga-1 for creating a foundation. Subsequently, activity ga-5 requires design information of the garden and the realized foundation from ga-2 and ga-4, respectively, for landscaping by the gardener. In the parameter DSM in the figure, two clusters of parameters are identified, regarding the foundation and the garden respectively.

The goal activity-actor DMM presents a single active actor and a single passive actor for each goal activity (i.e. row in the matrix). When the two clusters of the parameter DSM are mapped to the parameter-actor DMM, it is observed that the project manager and foundation worker are mainly related to the first cluster and the project manager and gardener to the second cluster. This reflects that the foundation worker is responsible for the foundation, the gardener for the garden, and the project manager for both. In the parameter-goal activity DMM, five dependencies are distinguished, corresponding to the goal activities.

4 HOUSE CONSTRUCTION PROJECT EXAMPLE

Figure 3 displays the MDM derived from all activities specified for the house construction project. The construction project is described by means of 26 goal activities, and 12 transformation activities with 7 different actors and 22 different parameters involved (activity specifications are not included in this document). The dependency types that correspond to the dependencies are presented in the legend next to the MDM. The seven actors in this house construction project example are the project manager, the architect, the resident, the bricklayer, the foundation worker, the roofer, and the gardener. The various activities range from designing the house by the architect to delivering the house to the residents. Again, the directed arrow in the top-left corner of the figure indicates the input in rows convention.

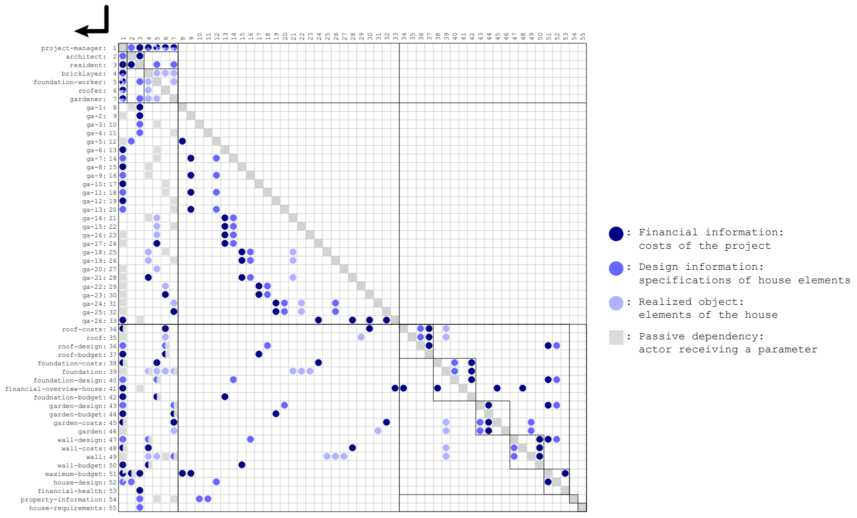

Figure 3. Multi-domain matrix of the example where a house is being constructed.

In the actor DSM, the Project Manager is identified as the bus element since this actor must coordinate the project and therefore has many dependencies with the other actors. In the actor DSM, two clusters are identified by applying a clustering algorithm (Wilschut *et al.*, 2017). The first cluster consists of the architect and resident, which are dependent in terms of resident wishes and house design. The second cluster consists of the bricklayer, foundation worker, roofer, and gardener, which together are responsible for the realization of the house. The goal activity DSM in the MDM displays the goal activities in a sequenced way. This follows from the natural sequence of specifying activities, beginning with the house design, via its construction, towards generating the financial overview of the project. The flow of activities displayed in this DSM starts by investigating the financial situation of the residents, which is used to set the financial budgets for realizing the house. After the design of the house has been realized, a foundation is made on which the walls of the house have to be built that form a basis for the construction of the roof. When the house itself has been realized the landscaping of the garden is performed by the gardener. Finally, all expenses are added up into a financial overview of the entire project. The process flow is displayed in the parameter DSM, where the clusters correspond to constructing the roof, constructing the foundation, landscaping the garden, constructing the walls, and designing the house from top to bottom, respectively. The clusters have been determined by the clustering algorithm. The fourth element in the foundation cluster represents the financial overview of the house, which has also mapping relations with other clusters and therefore could be assigned to different clusters.

The goal activity-actor DMM defines exactly one active and one passive actor for each of the goal activities, indicating properly specified goal activities. Furthermore, the parameter-goal activity distinguishes 26 dependencies, corresponding to the number of specified goal activities, as visualized in Figure 3.

5 CONCLUSIONS

For an increasing number of (design) activities, building a process DSM becomes time-consuming and error-prone. This paper shows that specification of activities according to a predefined textual format that is still readable like natural text, enables the automated derivation of the dependencies in the process DSM. When these activities are expressed in terms of goal and transformation activities with actors (people) as the source and purpose of the activities, the generated process activity DSM can be accompanied with an organization DSM (actor DSM) and a parameter DSM displaying the flow of information. Also, the mapping matrices between these three domains can be automatically generated. The generated MDM provides for an insightful display of the interplay of actors, activities, and flow of information.

The proposed method with the accompanying derivation of dependencies has been verified using a couple of smaller and larger test problems. The results show that various activities can be cast into the textual formats of Tables 4 and 5. Validation of the developed method is ongoing. For instance, we have been working on the specification of design, development, and testing activities for a fully electric racecar of University Racing Eindhoven, the Formula Student team at Eindhoven University of Technology. Also, other applications are planned where a team of engineers is using spreadsheets or text documents to manage their design or development activities.

https://doi.org/10.1017/dsi.2019.167 Published online by Cambridge University Press

When the prescribed textual form is applied the MDM display can be automatically generated from the tables with activity specifications. The MDM display is very helpful in revealing missing activities, or errors in the activity specifications. What is more, through the MDM activities can be easily sequenced to help to avoid unnecessary (design) iterations.

REFERENCES

Browning, T. R. (2016), "Design structure matrix extensions and innovations: a survey and new opportunities", *IEEE Transactions on Engineering Management*, Vol. 63, No. 1, pp. 27–52. https://doi.org/10.1109/TEM.2015.2491283

Chinosi, M., Trombetta, A. (2012), "BPMN: An introduction to the standard", *Computer Standards & Interfaces*, Vol. 34, No. 1, pp. 124–134. https://doi.org/10.1016/j.csi.2011.06.002

Dori, D. (2002), *Object-Process Methodology: A Holistic Systems Paradigm*, Springer-Verlag, Berlin-Heidelberg, Germany.

Dori D., Reinhartz-Berger I., Sturm A. (2003), "Developing Complex Systems with Object-Process Methodology Using OPCAT", *22nd International Conference on Conceptual Modeling*, Chicago, IL, USA, 13-16 October 2003, Conceptual Modeling - ER 2003, pp. 570–572. https://doi.org/10.1007/978-3-540-39648-2_46

Dori D., Reinhartz-Berger I., Sturm A. (2003), "OPCAT - A Bimodal Case Tool for Object-Process Based System Development", *5th International Conference on Enterprise Information Systems (ICEIS 2003)*, Angers, France 22-26 April 2003, Proceedings of the 5th International Conference on Enterprise Information Systems, pp. 286–291.

Dori, D. (2016), *Model-Based Systems Engineering with OPM and SysML*, Springer-Verlag, New York, NY, USA. https://doi.org/10.1007/978-1-4939-3295-5

Eppinger, S. D., Whitney, D. E., Smith, R. P., Gebala, D. A. (1994), "A model-based method for organizing tasks in product development", *Research in Engineering Design*, Vol. 6, No. 1, pp. 1–13. https://doi.org/10.1007/BF01588087

Eppinger, S. D., Browning, T. R. (2012), *Design structure matrix methods and applications*, MIT Press, Cambridge, MA, USA.

Lee, W., Ma, S., Lee, S., Law, P., Chang, T. (2015), "Extending the Multiple Domain Matrix by Using Business Process Model and Notation for Business Process Analysis", *2015 IEEE 12th International Conference on e-Business Engineering*, Beijing, China 23-25 October 2015, *Proceedings of the 2015 IEEE 12th International Conference on e-Business Engineering*, pp. 311–318. https://doi.org/10.1109/ICEBE.2015.60

Maurer, M. S. (2007), *Structural awareness in complex product design*, Verlag Dr. Hut, Munich, Germany.

Sharon, A., De Weck, O. L., Dori, D. (2009), "Is There a Complete Project Plan? A Model-Based Project Planning Approach", *INCOSE International Symposium*, Vol. 19, No. 1, pp. 96–109. https://doi.org/10.1002/j.2334-5837.2009.tb00940.x

Sharon, Am., Dori, D., De Weck, O. L. (2009), "Model-based design structure matrix: deriving a DSM from an object-process model",*Second International Symposium on Engineering Systems*, Cambridge, MA, USA, 15-17 June 2009, MIT ESD and CESUN, pp 1–12.

Steward, D. V. (1965), "Partitioning and Tearing Systems of Equations", *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, Vol. 2, No. 2, pp. 3445–365. https://doi.org/10.1137/0702028

Steward, D. V. (1981), "The design structure system: a method for managing the design of complex systems", *IEEE Transactions on Engineering Management*, Vol. EM-28, No. 3, pp. 71–74. https://doi.org/10.1109/TEM.1981.6448589

Van Beek, T. J., Erden, M. S., Tomiyama, T. (2010), "Modular design of mechatronic systems with function modeling",*Mechatronics*, Vol. 20, No. 8, pp. 850–863. https://doi.org/10.1016/j.mechatronics.2010.02.002

Vaneman, W. K. (2016), "Enhancing Model-Based Systems Engineering with the Lifecycle Modeling Language", *2016 Annual IEEE Systems Conference (SysCon)*, Orlando, FL, USA, 18-21 April 2016, Proceedings of the 10th Annual IEEE Systems Conference, pp. 1–7. https://doi.org/10.1109/SYSCON.2016.7490581

Wilschut, T., Etman, L. F. P., Rooda, J. E., Adan, I. J. B. F. (2017), "Multi-level flow-based Markov clustering for design structure matrices", *Journal of Mechanical Design: Transactions of the ASME*, Vol. 139, No. 12, p. 121402. https://doi.org/10.1115/1.4037626

Wilschut, T., Etman, L. F. P., Rooda, J. E., Vogel, J. A. (2018), "Generation of a function-component-parameter multi-domain matrix from structured textual function specifications", *Research in Engineering Design*, Vol. 29, No. 4, pp. 531–546. https://doi.org/10.1007/s00163-018-0284-9

https://doi.org/10.1017/dsi.2019.167 Published online by Cambridge University Press