Product, Processes and Organization Architectures Modeling : from strategic expectations to strategic competences.

Ghassen Harmel, Eric Bonjour, Maryvonne Dulmet

PRODUCT, PROCESSES AND ORGANIZATION ARCHITECTURES MODELING: FROM STRATEGIC EXPECTATIONS TO STRATEGIC COMPETENCIES

G. HARMEL, E. BONJOUR, M. DULMET
Laboratoire d'Automatique de Besançon
- UMR CNRS 65 96
ENSMM - Université de Franche-Comté - 24 rue Alain Savary - 25000 Besançon
gharmel@ens2m.fr ebonjour@ens2m.fr mdulmet@ens2m.fr

ABSTRACT: *Complex systems design and especially automotive design is facing continuous technological evolution that needs stronger integration. In this paper, a method for representing system architecture is presented. This method is based on dependencies and constraints propagation which allows us to focus on the co-evolution of product, organization and processes domains.*

KEY-WORDS: *product architecture, organization structure, processes structure, DSM, competencies*

1. INTRODUCTION

Nowadays, firms' survival depends on their ability to deal with the constant evolution of customers needs. The fast reactivity and adaptation of firms to this changing environment is strongly related with an advanced control of their internal mechanisms. Concerning product development situations, researchers (Allen, 1997) (Pimmler and Eppinger, 1994) have highlighted the interdependency between three domains which needs to be modeled. These relevant domains are: Product, Processes, and Organization. Product domain covers "what" to offer to customers and processes domain covers "how" to do to achieve it. While Organization reflects which actors and skills are engaged in the development process.

Product architecture choice depends on the innovation and standardization policy of the firm (eg. modular products, products families, unique products…). Process architecture depends on the design methodology adopted (eg. project management, system engineering) and is constrained by the physical decomposition of the firm into departments and teams. The choice of organization architecture is related to the other domains and so it may depends on their constraints.

In order to represent the above mentioned synergy, many tools and models were developed. Unfortunately, they capture only static and partial views of the development situations.

That is why, we develop here a method based on matrix tools allowing to represent each domain by a matrix and to link them in order to propagate constraints and dependencies.
Starting from the propagation of customers' needs, we obtain therefore a dynamic representation of the co-evolution of all subsystems domains. We also aim to show that DSM –Design Structure Matrix- is a relevant tool for modeling the architecture of each domain at different levels of details, it also allows to represent and explain the propagation of constraints and dependencies in a project.

In this paper, we explain the model by applying it to a real Robotized Gearbox development situation observed during our frequent collaborations with an automotive company.

In order to simplify the representation, organization domain includes competencies and actors. This hypothesis is frequently implicitly taken in many papers dealing with the restructuring of Organizations.

2. MODELING TOOLS

2.1. Design Structure Matrix: structure modeling tool

The design structure matrix (DSM) is becoming a popular modeling and analysis tool, especially for purposes of decomposition and integration. A DSM displays the relationships between components of a system in a compact, visual, and analytically advantageous format. A DSM is a square matrix with identical row and column labels. In the example DSM in Fig. 1, elements along the diagonal have no sense. An off-diagonal mark signifies the dependency of one element on another. Reading across a row reveals what other elements the element in that row provides to; scanning down a column reveals what other elements the element in that column depends on. That

is, reading down a column reveals input sources, while reading across a row indicates output sinks.
Thus, in Figure 1, element D provides something to elements C, E and G, and it depends on something from element E.

	A	B	C	D	E	F	G
Element A	A	■					■
Element B	■	B	■				■
Element C	■		C	■			
Element D				D	■		
Element E				■	E	■	■
Element F					■	F	
Element G		■		■		■	G

Fig 1. Product DSM

DSM is a tool widely used in different research fields:

- Project Management (Browning and Eppinger, 2002),
- Organization architecture (McCord et Eppinger, 1993),
- Exchange flows (Allen 1977) (Pimmler et Eppinger, 1994),
- Process architecture (Yassine, *et al.,* 1999) (Browning 2002) (David, *et al.*, 2002).

For a more detailed overview of the DSM method and its applications, the reader is referred to Steward (1981) and Eppinger, *et al*. (1994).
There are two main categories of DSMs: static and time-based. Static DSMs represent system elements existing simultaneously, such as components of a product structure or groups in an organization. Static DSMs are usually analyzed with clustering algorithms. In time-based DSMs, the ordering of the rows and columns indicates a flow through time: upstream activities in a process precede downstream activities, and terms like "feedforward" and "feedback" become meaningful when referring to interfaces. Time-based DSMs are typically analyzed using sequencing algorithms.
In this article, we straddle these various fields to deal at the same time with product, organization and processes architectures.

2.2. Mapping and allocation matrix

This second matrix based tool is an inter-domain DSM as classified by Malmqvist (2002), this type of DSM serves at mapping between heterogeneous domains like product, organization and processes domains. The matrix obtained is rectangular with different elements in rows and columns, the relations handled by this matrix are allocation type.

3. MODELING SYSTEM ARCHITECTURE WITH DSM

The development of complex products, such as cars, computers, or aircraft engines requires at the same time the use of a general approach –in order to deal with product, organization and processes domains- and a well structured approach - enabling to capture the interactions, the decomposition and the integration of elements composing the system.

DSM tool is an appropriate way to capture the structure of a heterogeneous system. The ease of use and the simplicity of DSM tool make possible to concentrate a huge amount of information in a single matrix. In the following part of this paragraph, we will see how to use DSMs to capture the structure of the development domains.

3.1. Product architecture

Ulrich (1995) defines product architectures as "the scheme by which the function of a product is allocated to physical components." A key feature of product architecture is the degree to which it is modular or integrative. In modular architectures, functional models of the product map one-to-one to its physical components. On the other hand, in integrative architectures a large subset of the functional model maps to a single or small number of components.

In the engineering design field a large stream of research has focused on methods and rules to map functional models to physical components. Researchers have developed several architecting rules to map functions to physical modules (Shapiro and Voelcker, 1989; Liu, *et al.*, 1999).

By using established concepts in the product architecture literature we can categorize systems as modular or integrative based on how their corresponding components share design interfaces within the system.
From an external perspective modular and integrative systems concept is based on the existence of design interfaces between components of the same product that belong to different systems. Then, modular systems are those whose design interfaces with other systems are clustered among a few physically adjacent systems, whereas integrative systems are those whose design interfaces span all or most of the systems that comprise the product due to their physically distributed or functionally integrative nature throughout the product.

Product architecture modeling. In order to study the structure of product architecture, in terms of product interactions, we use the Design Structure Matrix tool. Figure 2, illustrates how a manufactured product can be decomposed into modular and integrative sub-systems. Thus, we identify two modules (A,B,C) and (D,E,F) which share few external interfaces by comparison to G –integrative subsystem- who shares 3 external interfaces within the product.

	A	B	C	D	E	F	G
Element A	A	■					■
Element B	■	B	■				■
Element C	■		C	■			
Element D				D	■		
Element E				■	E	■	■
Element F					■	F	
Element G		■		■		■	G

Fig 2. Product DSM rearranged

3.2. New Product Development Processes structure

Most of the researches in this area assume that the design process has an underlying structure (Alexander, 1964; Smith and Morrow, 1999).
At an interesting level of detail, NPD processes do not proceed in a purely sequential way (Cooper, 1993). The activities in a NPD process interact by exchanging information (Clark and Fujimoto, 1991).
According to Hammer and Stanton (1999), a process is an organized group of related tasks that work together to create a result of value.
Process architecture—the elements of a process (tasks) and their pattern of interaction—is an important process variable (Von Hippel, 1990).

Process architecture modeling. Decomposition is the standard approach to address system complexity in the same way of what we did with the product domain. As a kind of system, a process is defined not only by its decomposition into tasks but also by how they work together (Browning, 2002, 2003).
A process is often modeled as an activity network. The visual representation is too busy, making it difficult to discern architectural differences. And since many process flowcharts capture only a single input and output for any given activity, the full range of information flow is seldom represented. Using a flowchart for this purpose would simply be too complicated and cumbersome.

A design structure matrix (DSM) can also be used to represent a process (Browning, 2003; Steward 1981; Eppinger, et al. 1994). The DSM shows activities and interfaces in a concise format. A Process DSM is a square matrix in which a cell on the diagonal represents each activity. A mark in an off-diagonal cell indicates an activity interface. For each activity, its row shows its inputs and its column shows its outputs. When activities are listed in temporal order, subdiagonal marks denote a feeding of deliverables forward in the process, from upstream activities to downstream activities, while superdiagonal marks indicate feedback. The DSM provides a simple way to visualize the structure of an activity network and to compare alternative process architectures.

3.3. Organization structure

Motivation. Organization is a very recurrent term in many research fields. Thus, we have to clarify this concept from our point of view.
Organization includes with no doubt the actors participating to the attainment of the firm' objectives, and depending in which research field we are, it may include the processes regulating and controlling its behavior. In this paper, organization structure is referred to the interactions between actors. Thus, the organization structure determines who works with whom and who reports to whom. Particularly, in development organizations we are interested in studying the interactions among the people conducting the technical development project.
Organizations are extremely complex systems (Donnadieu and Karsky, 2002). Better understanding of organizations enables innovation and improvement in organization design (Sosa, *et al.*, 2003). Complex system development requires the exchange of information among various groups or teams.
Relationships among people and teams are what give organizations their added value and build collective competencies. Analyzing a team-based DSM highlights interteams interfaces, which provide the greatest leverage for improving the organization. Better understanding of organizational interfaces supports the application of appropriate integrative mechanisms (Browning, 2001, 2003), and help identifying strategic competencies.
From an organizational viewpoint, complex development projects usually involve the efforts of hundreds of team members. A single team does not design the entire product at once (it is too complex). Rather, many teams develop the components, and work to integrate all of these components to create the final product (Alexander, 1964; Allen, 1977; Eppinger, 1997). We call *modular design teams* those which design modular systems while *integrative design teams* are those which design integrative systems and more generally those which cannot be mapped to a modular system.
Development organizations face an important challenge which is product integration (Meinadier, 2002). In order to win this challenge, we need to put the stress on coordination and collaboration interactions.
In order to investigate actors' interactions, we use the DSM tool applied to organization structure.

Organization structure modeling. By filling in DSMs with prescribed perspective (Figure 3), we obtain a clear representation of complex organizations' structures easy to analyze. Investigating the differences between prescribed and realized organization structure can help us identify communication barriers and characterize them.

		A	B	C	D	E	F
Motor Team	A		X				X
Transmission Team	B	X		X			
Drive train Team	C		X				X
External equipments Team	D						X
Internal equipments Team	E						X
structure and openings Team	F	X		X	X	X	

Fig 3. Organization structure DSM

We distinguish in organization architecture between individual and collective actor.

Individual actor: is a person who is in charge of performing a mission or a set of missions and who

acts permanently, temporarily or occasionally at the service of a considered company. This actor is characterized by his corpus of competencies and belongs to a recognized skill-network (design diesel engine, architecture of electronic systems, quality management...).

Collective actor: This is a team who is in charge of performing a mission or a set of missions and who acts permanently, temporarily or occasionally at the service of a company. A collective actor may be split up into smaller collective actors and/or individual actors. An individual actor may take part in different collective actors.
Figure 3 shows that, by investigating the need for communication interactions, we highlight the existence of a metateam (A, B and C). This team is considered as a collective actor and needs to be more in deep analyzed to address the collective competencies and mechanisms explaining its behavior and its performance.
The model of organization structure given above put the stress on his information processing function. In our research, we find this model insufficient, that's why we complete it with a process structure model that introduces the cognitive concept of competency.

Competency concept. Competency may be required and then, referred to a given task (or mission) that needs to be carried out in the future. Competency may be acquired and recognized, embodied by an actor (one or several persons) and then, available to perform a given similar mission.

Our research team proposes the following definition of competency (Bonjour and Dulmet, 03): Competency is the "mobilization" and dynamic organization of a set of heterogeneous cognitive resources that leads to the production of an acknowledged performance, in the framework of a finalized activity and a particular class of situations. "

However, in this paper we will only deal with prescribed competencies which are directly derived from tasks formulation.

In the following development of our paper, we assume that one task implies one mission, and one mission is derived from one task. A required competency is defined by reference to a mission, we obtain so a bijection between missions domain and competencies one.

4. SOCIO-TECHNICAL SYSTEM MODELING: ROBOTIZED GEARBOX DEVELOPMENT SITUATION

4.1. Motivation

A socio-technical system is a special system organized around an actor. Thus, a socio-technical system can be a firm or any subsystem having an objective and composed of actors.
In this paper we focus on a gearbox development situation. A Robotized Gearbox (RG) acts as an automatic gearbox without being too expensive. In fact, in a RG, there is an automated controller who shifts the speeds of a manual gearbox instead of the customer.
Our purpose is to model the dependencies and the co-evolution of the product, the organization and the processes structures. Starting from the product domain, we will propagate the voice of the consumer until the organization and competencies domains. Our purpose is to identify the interdependencies between all these domains and how these interrelations are handled in order to fulfill customer expectations.

4.2. Propagating the voice of the consumer to product architecture

From the consumer point of view, a car as any other product is an object providing a set of services. Each automotive company employs a panel of experts struggling to find out and to formulate consumer's expectations. These expectations concerns the entire vehicle, thus we need to decompose them into more precise expectations linked to car subsystems and components.
In this paper we will treat the example of Drivability expectation. Drivability is the general qualitative evaluation of a powertrain's operating qualities, including idle smoothness, cold and hot starting, throttle response, power delivery, and tolerance for altitude changes.
Drivability is assumed to be a strategic consumer expectation. When applied to a gearbox, we wish to determine which component is especially concerned.
The mapping between expectations domain and product domain is not detailed in this paper, the only information needed is that drivability is carried out by the shifting system function (SF COM).

4.3. Robotized Gearbox architecture modeling

In the Product architecture modeling paragraph, we have observed that product architecture can be decomposed into modular and integrative subsystems. We believe that the choice of product architecture impact the architecture of processes and the architecture of organization. After all, the development organization is executing the development process, which is implementing the product architecture.
Even though the RG is a new product with a new expectation -the automation of gears shifting- the architecture of the product do not evolve by comparison to a manual Gearbox, thus we obtain the RG DSM in figure 4. The DSM obtained is rearranged using clustering algorithms in order to highlight the existence of modules.

			CDI	SYN	ACT	EMB	CIE	DIFF	MEI	CART
SF VOL	SF COM	CDI		X	X					X
		SYN	X						X	
		ACT	X				X			X
	SF COU	EMB					X		X	
		CIE			X	X				X
	SF TPU	DIFF							X	X
		MEI		X		X		X		X
	SF EFF	CART	X		X		X	X	X	

ACT: actuators
EMB: clutch
SYN: Synchronizer
DIFF: Differential
CART: housing
CDI: internal control parts
SF COU: coupling
MEI: internal mechanical parts
CIE: internal clutch control
SF VOL: functional volumes
SF COM: shifting
SF TPU: power transmission
SF EFF: efforts recovering

Fig 4. RG product DSM

The RG is a complex system composed of 8 components and hundreds of parts. In order to overcome complexity barrier, we decompose it progressively into subsystems and so on. The first level of decomposition applied to our product reveals the principal system functions carried out. We obtain 5 principal functions explaining the internal function of RG. The identification of system functions (SF) and the mapping between functions and components are facilitated by the typicality of the architecture of the RG. In fact, as we can notice it in the figure 4, RG is composed of three modules and two integrative components. The modules are directly linked to system functions and the integrative components are either linked to a module or supporting directly a system function. We observe also that a system function can be integrative, it is the case of the functional volumes SF which covers all the other SFs.

Drivability, which is carried out by Shifting SF, can be linked to a modular subsystem (CDI, SYN and ACT). What need to be noticed at this point are the external interfaces of this module. In fact, if an alteration is noticed in drivability expectation or if we need to investigate the impact of changes carried out by the system function on the product, we need to link the module to other subsystems.
From the DSM in figure 4, we notice also that shifting module is linked to two integrative components who are MEI and CART and that ACT component belongs to two modules. Thus, if drivability causes modification in MEI or CART components, all the subsystems of the product may be impacted. And if ACT component is concerned, two modules are impacted: shifting (COM) and coupling (COU).

In the following paragraph, we link product architecture to design processes.

4.4. From Product to processes domain

In the left side of figure 6, we find a listing of prescribed tasks describing the NPD processes of the RG. These tasks can be detailed and decomposed in order to become usable by designers, but the level of detail proposed in this paper is sufficient to cover all design process and to be in cohesion with the level of detail adopted in the product architecture.
We notice in the mapping matrix obtained that the system functions are conserved. In fact, the RG is a complex system, so we need first to identify system functions then to map these functions to components. When system functions were designed, the components were not identified yet.
Throw the mapping obtained, we notice that drivability is linked principally to 4 prescribed tasks (2BV1, 6BV1, 6BV2 and 6BV3).

4.5. RG development processes architecture modeling

The processes DSM is a time-based matrix, the scope of this matrix is to represent the dependencies between tasks.
The DSM obtained in figure 7 allows us to analyze the dependencies of the tasks related to those linked to drivability expectation. Thus 2BV1 task is constrained by the specification of the other system functions but constrains kinematics choice and indirectly all the other system functions, this information is drawn by the fact that SF VOL is on the one hand linked to 3BV1 and on the other hand is integrative among all the other SFs.
At this point, we are able through the two matrices already constructed to link drivability to the concerned components and tasks.

4.6. From process domain to competencies domains

By reference to the paragraph (§3.3), we remind that processes and competencies domains are bijective, without reformulating tasks we can read directly the competence required from a task. For example the competence linked to task 0BV1 is the capability to carry out negotiations around the specifications of a gearbox.

4.7. From competencies domain to organization structure domain

As pointed out in Organization structure paragraph (§3.3), any organization can be decomposed into individual and collective actors. At this point, we don't take into account resources allocation problems.

RG development is organized as a project. The project organization structure call up actors having the following job positions: project manager, function system designer, component manager, calculus manager and designer.
According to the level of detail adopted in our example, the designers do not appear in our organization because they are related to parts design.
The matching between job positions and product architecture leads to the list of project roles in figure 8.

At this point we can match between competencies and project roles. The matching is based on the relation one role is the decision maker related to one competency.
Logically, in order to avoid conflicts, there is only one decision maker related to each task.
By continuing the propagation of drivability expectation impacts, we observe that the roles concerned are those of ARSF COM, CdP SYN, CdP CDI and CdP ACT.

4.8. RG development organization architecture modeling

The prescribed organization structure OS-DSM in figure 9 is obtained by reporting the prescribed interactions between actors during the RG development project.
We notice that the project manager (PM) will interact during the project with all the other actors, this information is based on the combination between the processes DSM and the mapping matrix between processes and organization structure. In fact, the PM is responsible of 0BV1, 5BV1 and 8BV1 tasks, and these tasks are related to others ones which are under the responsibility of all the actors.
By reorganizing Organization structure DSM and by introducing redundant SF roles, we obtain the particular architecture in figure 9. This DSM conducts us to make the following remarks:

- There are two integrative actors: the PM and the PMIV. During the RG development, they will interact with all the other actors;
- The architecture of the project is flower like (figure 5). There is a central collective actor regrouping the system functions managers. ARSF VOL is the integrative actor between all the ARSFs;
- Each system manager apart from ARSF VOL belongs to two collective actors;
- The actors linked to drivability expectation form a collective actor. In this collective actor like in the others, the ARSF interacts with all the components managers who do not interact with each others. This leads us to conclude that the ARSF has an integrative role inside a component team.
- The remarks made before give us an idea about the ideal physical organization of actors. All the actors belonging to the RG project must share the same space with respect to the flower like organization.

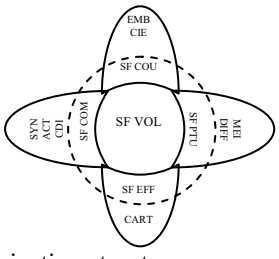

Fig 5. RG organization structure representation

5. DISCUSSION

Product, Organization and Processes domains are inter-related. This fact increases the level of complexity of social-technical systems and forces us to deal with the whole system in order to capture its real behavior. We have chosen to develop our method starting from consumer's expectation going through product domain, processes domain and arriving to competencies and organization structure.
The model proposed in this paper is simplified by the following important hypothesis:

- The level of detail adopted in product domain and which impacts the decomposition of the other domains is 8 components decomposition only, it allows us to handle partially the complexity of the global system. However, the explanation of some interactions and dependencies used in our example are derived from a more in deep knowledge of the system.
- The competency domain is assumed to be bijective with tasks domain. This is a good starting point for our example but competency concept is more complex and gain to be more in deep developed in future researches.

The partial modular architecture of the product impacts the structure of all the other domains. In our case, this special architecture facilitated the propagation of the voice of the consumer. In fact, we found a product module impacted by drivability expectation and a collective actor designing that module.
In drivability propagation case, we are able trough the example treated to identify in each domain the subsystems directly impacted and even those who may be impacted indirectly. From a manager point of view, this tool becomes essential for developing a global and detailed idea of the interactions of intra and inter-domains and gives a quick representation of changes applied on any element of the system.
More precisely, as drivability is identified as being a strategic expectation, then we are able to affirm that shifting module is a strategic component for the company and that the shifting team (collective actor) holds strategic competencies.

6. CONCLUSION

The DSM applications reviewed in this paper demonstrate the main strength of matrix-based approaches: concise, visual representation of complex systems. This paper emphasizes, first how DSMs facilitate intelligent system decomposition and representation, and secondly how an original representation and analysis model can be developed using the combination of DSMs and allocations matrices.
In industrial context, this model gives managers a global sight about the impacts of changes (reorganization, new element, etc.) from one domain to the others. Specially, in this paper we highlight the

necessary linking between the strategic drivability expectation and the strategic component, team and competencies which are related to that expectation.

REFERENCES

Alexander, C. (1964). *Notes on the Synthesis of Form*, Harvard University Press, Cambridge.

Allen, T.J. (1977). *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D Organization*. MIT Press, Cambridge, MA.

Browning, T.R. (2001). Applying the design structure matrix to system decomposition and integration problems: A review and new directions, *IEEE Trans. Eng. Mgt.*, **vol. 48,** pp. 292–306.

Browning, T.R. (2002). Process integration using the design structure matrix, *Syst. Eng.*, **vol. 5, no. 3**, pp. 180–193.

Browning, T.R. (2003). On customer value and improvement in product development processes, *Syst. Eng.*, **vol. 6, no. 1**.

Browning T.R., and S.D. Eppinger, (2002). Modeling the impact of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management*.

Clark, K. B. and T. Fujimoto (1991). *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*. Harvard Business School Press.

Cooper, K. G. (1993). The rework cycle: Why projects are mismanaged, *PMNETwork*.

David M., Z. Idelmerfaa. and J. Richard (2002). Organization Method for Complex Cooperative Design Projects. *IEEE 2002, Hammamet*.

Donnadieu G. and M. Karsky (2002). *La systémique, penser et agir dans la complexité*. Editions liaisons.

Steward, R. P. and V. Donald (1981). The Design Structure System: A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management* **28, No.3**, 71-74.

Eppinger, S. D., D. E. Whitney, R. P. Smith, and D. A. Gebala (1994). A Model-based Method for Organizing Tasks in Product Development. *Research in Engineering Design,* **6: 1**, pp. 1-13.

Eppinger, S. D. (1997). A Planning Method for Integration of Large-Scale Engineering Systems,'' *International Conference on Engineering Design(ICED 97)*.

Hammer, M. and S. Stanton (1999). How process enterprise really work. *Harv. Bus. Rev.*, 108-118.

Liu, Y. C., A. Chakrabarti, and T. P. Bligh (1999). Transforming Functional Solutions into Physical Solutions, *ASME*, *DETC/DTM-8768*.

Malmqvist J. (2002). A classification of matrix based methods for product modeling, *Design 2002*, 14-17, Cavtat-Dubrovnik, Croatia.

McCord, K. R. and S. D. Eppinger (1993). *Managing the Integration Problem in Concurrent Engineering*, M.I.T. Working Paper no.3594.

Meinadier, J.-P (2002). *LE METIER D' INTEGRATION DE SYSTEMES,* Hermes Science Publications.

Pimmler, T. U. and S. D. Eppinger (1994). Integration Analysis of Product Decompositions. *ASME (DTM'94)*, **Vol 68**, pp. 343-351.

Shapiro, V., and H. Voelcker (1989). On the Role of Geometry in Mechanical Design, *Res. Eng. Des.*, **1**, pp. 69–73.

Smith R. P. and J. A. Morrow (1999). Product development process modeling, *Design Studies*, **vol. 20**, **no. 3**, pp. 237–261.

Sosa, M. E., S. D. Eppinger, and C. M. Rowles (2003). Identifying modular and integrative systems and their impact on design team interactions. *ASME* **Vol 125**.

Ulrich, K. T. (1995). The Role of Product Architecture in the Manufacturing Firm, *Res. Policy*, **24**, pp. 583–607.

Von Hippel E. (1990). Task partitioning: An innovation process variable, *Res. Policy*, **vol. 19**, pp. 407–418,.

Yassine A., D. R. Falkenburg and K. Chelst, 1999. Engineering Design Management: An Information Structure Approach. *International Journal of Production Research*, **vol. 37, no. 13**, pp. 2957-2975.

	VOL	*COM*	COU	TPU	FFF	ACT	SYN	CDI	EMB	CIE	DIFF	MEI	CART
0BV1	X	X	X	X	X								
1BV1				X									
1BV2			X										
1BV3					X								
2BV1		**X**											
3BV1	X	X	X	X	X								
4BV1	X			X									
4BV2	X		X	X									
4BV3					X								
5BV1	X	X	X	X	X								
6BV1		**X**				**X**							
6BV2		**X**					**X**						
6BV3		**X**						**X**					
6BV4			X						X				
6BV5			X							X			
6BV6				X							X		
6BV7				X								X	
6BV8					X								X
7BV1													
8BV1												X	X
9BV1	X	X	X	X	X	X	X	X	X	X	X	X	X

0BV1 Negotiate specifications with motor DT
1BV1 Specify TPU function
1BV2 Specify COU function
1BV3 Specify FFF function
2BV1 Specify COM function
3BV1 Fix kinematics
4BV1 Fix axle spread
4BV2 Fix internal clutch control
4BV3 Fix lubricant
5BV1 Fix gearbox architecture
6BV1 Design ACT
6BV2 Design SYN
6BV3 Design CDI
6BV4 Design EMB
6BV5 Design CIE
6BV6 Design DIFF
6BV7 Design MEI
6BV8 Design CART
7BV1 Fix rolling bearings and casing
8BV1 Fix components design
9BV1 Edit validation and integration document

Fig 6. Product-Processes Mapping matrix

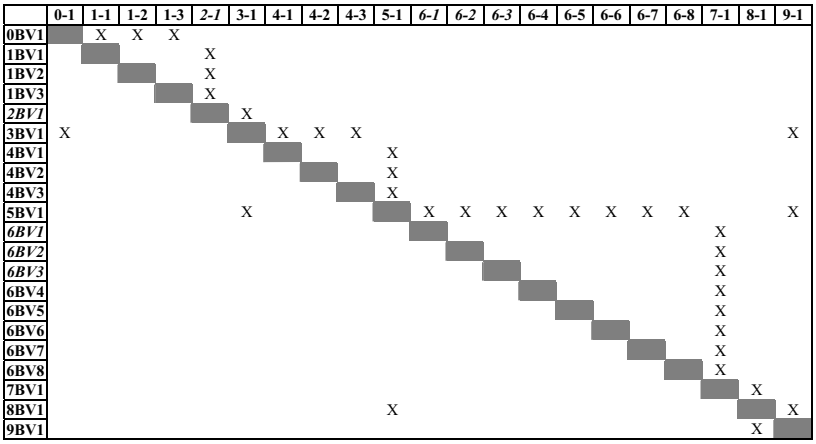

Fig 7. RG processes domain DSM

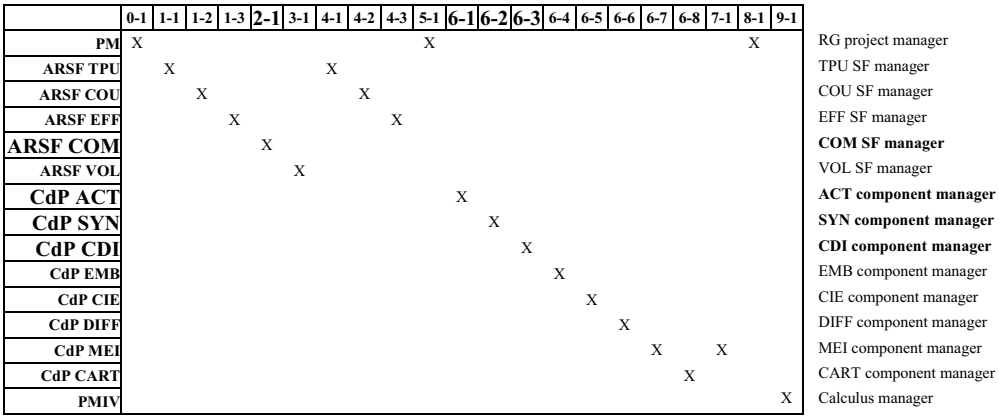

Fig 8. Competencies-Roles mapping

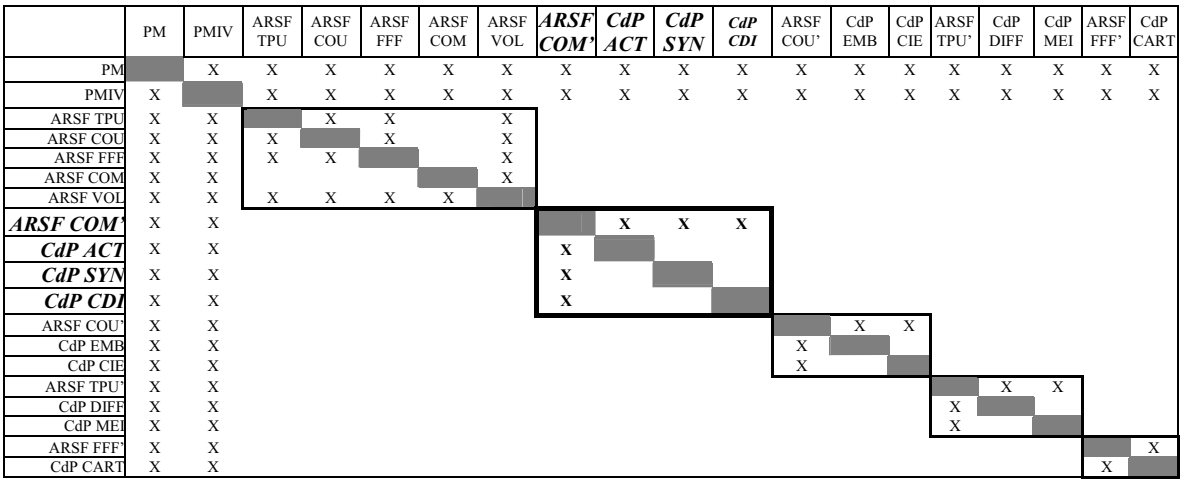

Fig 9. Organization structure DSM