7TH INTERNATIONAL WORKSHOP ON INTEGRATED PRODUCT DEVELOPMENT
IPD WORKSHOP | 17.-19. SEPTEMBER 2008 | MAGDEBURG - BAD SALZELMEN

PRODUCT DEVELOPMENT PROCESS SCHEDULING WITH MULTI-VARIABLE HEURISTIC METHOD

István Groma[1], Tibor Bercsey[2]

[1,2]Budapest University of Technology and Economics, Dept. of Machine- and Product Design

Keywords: Genetic Algorithms, Task Scheduling, Resource Allocation, DSM, Product Development

ABSTRACT

Nowadays, product development processes are typically specified as bounded work processes (as a workflow), and their treatment is performed by PLM systems. One of the benefits of this approach, as implemented by various PLM applications, is that it integrates PDM/EDM, and in some cases PPS, SCM tasks as well. However, development processes are unique and valid in a given context that changes over time. Therefore, it is suitable to use a DSM, through which one can get a more precise picture of the product during the development planning process. This approach supports the modelling of iterations that are quintessential in practice. With the help of genetic algorithm-based optimization, one can find the development schedule with the least duration and cost by changing the schedule of the product elements. As a result, a project plan can be obtained and for each element, the required resources can be assigned. For solving the problem of resource assignment, a heuristic method was worked out where the project plan is fitted to the enterprise resource environment by applying various strategies, focusing on minimizing the duration of the development project.

1 INTRODUCTION

The subject of product development is the product that is designed to fulfill some kind of needs. The process (between the need and an actual product that is realized physically) has been a popular subject of research and it is being actively investigated in recent years. This area of research can be divided into two main parts: descriptive and prescriptive definitions.

Descriptive definitions do not offer specific processes. Examples are Pugh's Total Design theory, Suh's Axiomatic Design, and Tomiyama and Yoshikawa's General Design Theory, the TRIZ created by Altschuller, and Linde's and Hill's WOIS – Contradiction-Oriented Innovation Strategy [3].

On the contrary, prescriptive definitions do give us the list of tasks that need to be done in order to perform planning. Theories yielding to such processes were formulated by Hubka, Koller, Roth, Rodenacker, and Pahl and Beitz, In Europe, the most widely accepted theory is the process suggested by VDI 2221 [7]. The advantage of VDI 2221 is that it can be applied to both the entire product and any of its components or smaller parts. On the other hand, the real processes are affected by various further factors, such as the number and type of the design (new, variant,

or fitted construction), and the duration and cost. Numerous approaches were developed to reduce product development time, such as that of concurrent engineering, simultaneous engineering and frontloading, each of which attempt to optimize development duration by parallel tasks, strengthening teamwork, and the early availability of information.

The theories and methods mentioned above do little to aid the planning and modelling of product-oriented design processes. We need a method that defines the design process based on and according to the structure and features of the product to be designed [4]. This method needs to be able to represent the necessary iterations of the product development process and to handle the planning processes of products with various size and complexity – in other words, it is able to represent the groups of related elements that are best to be assigned to the same development teams. It should also support the creation of a project plan and the handling and assignment of resources – both quintessential for management. One approach that makes all of these possible is the Design Structure Matrix (DSM) [6].

2 PAPER LAYOUT AND STYLES

The DSM approach is based on the idea that one can change the order of tasks based on the relationships among the development processes of the sub-components. By reordering, one can find a sequence of tasks that may contain fewer cycles and identify those tasks that can be performed in parallel. This optimum signifies the most favourable assignment of total person-hours and total cost, and the final project plan may take less time if there are tasks that can be performed in parallel.

The tasks A_i involved in a product planning process identify the matrix shown in Figure 1. The diagonal elements represent self-referencing, which is avoidable so they should be set to zero: $a_{ii}=0$. The remaining elements can be used to denote various relationships between the tasks. If A_i provides information to A_j, then $a_{ij}=1$, otherwise $a_{ij}=0$, which signifies that there is no direct relationship between A_i and A_j. If an element of the matrix holds that $a_{ij}=1$ where $i<j$, then we speak of a feed-forward relationship (the element lies above the diagonal), whereas if $i>j$ the entry denotes a feedback relationship (the element lies below the diagonal). In the case of cycles, the total number of loops should be determined in order to keep the design process finite.

Furthermore, numerous indicators can be assigned to the elements of the matrix that makes DSM more applicable to a wider range of problems. Our approach handles the cost and duration constraints specified by the management of the organization.

For optimizing the development process, a genetic algorithm was chosen. This algorithm makes it possible to quickly solve robust and extensive problems yielding an optimal solution that fits multiple criteria.

A_i	1	2	3	4	5	6	7
1						1	
2						1	
3				1			1
4					1		
5						1	
6							1
7						1	

Figure 1: Example of DSM

2.1 Iterations and optimization

Planning is an iterative process and this fact should be taken into account while planning the development process. We often come across the problem of having to repeat the planning of certain tasks because of unclear requirements or at other times the type of the design itself requires us to redefine the model. Such cases can be planned in advance and the development process can be tailored accordingly. It is possible to define repeated task as new versions of the same task. This way the cycles can be eliminated what is called unfolding the DSM.

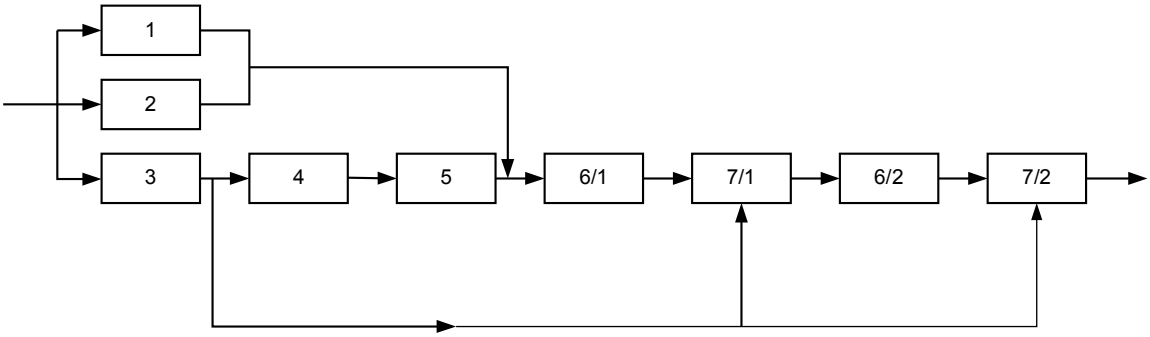

Figure 2: The unfolded graph of an iterative process segment

The unfolded representation shown in Figure 2 is more informative and in fact indispensable for process planning. Relations do not disappear during optimization, instead the best entry points of cycles should be found in the graph. One basic requirement of development projects is to be able to work with real information in every work phase. For instance, there can be two different ways for executing the iteration shown in Figure 3 and its DSM definition: either as *a-b-a-b* or *b-a-b-a*, thus ensuring that the right information is always available at the right time.

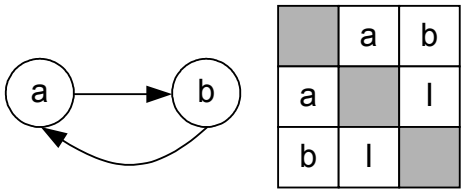

Figure 3: DSM for representing iteration

Figure 4 shows a DSM consisting of three tasks. In the first case, the order of task is *a*-*b*-*c*, where *b* and *c* can be executed in parallel, but they both tie down resources. In order to keep the condition that all tasks should be done according to the latest relevant information, we must perform nine activities. In the second case, by rationalizing the order of the task, only seven activities are needed. Doing so, time and money can be saved.

The aim of this ordering is to find the best entry points of cycles in order to minimize the cost and the duration of the development.

A DSM prescribed by a planning professional for a given development process is usually far from optimal: the duration and cost requirements are likely to exceed the optimal level, planning cycle iterations are unnecessarily complex and pessimistic, making the resource allocation unnecessarily involved. Therefore, as a first step a genetic algorithm is used in order to compute a DSM that reflects the optimal activity breakdown, which provides the best duration and cost combination. The inputs of this optimization are the tasks and their durations and costs. When defining the development order the goal is to reduce the duration and total cost of the process.

In the method described here, a chromosome (a potential solution instance in the genetic algorithm) corresponds to a particular task ordering where the genes contain one definite order of the tasks. The genetic algorithm uses the usual operators: selection (binary contender selection), crossing (partial crossover), and mutation (per gene). The parameters that drive these operators are discussed in [5].

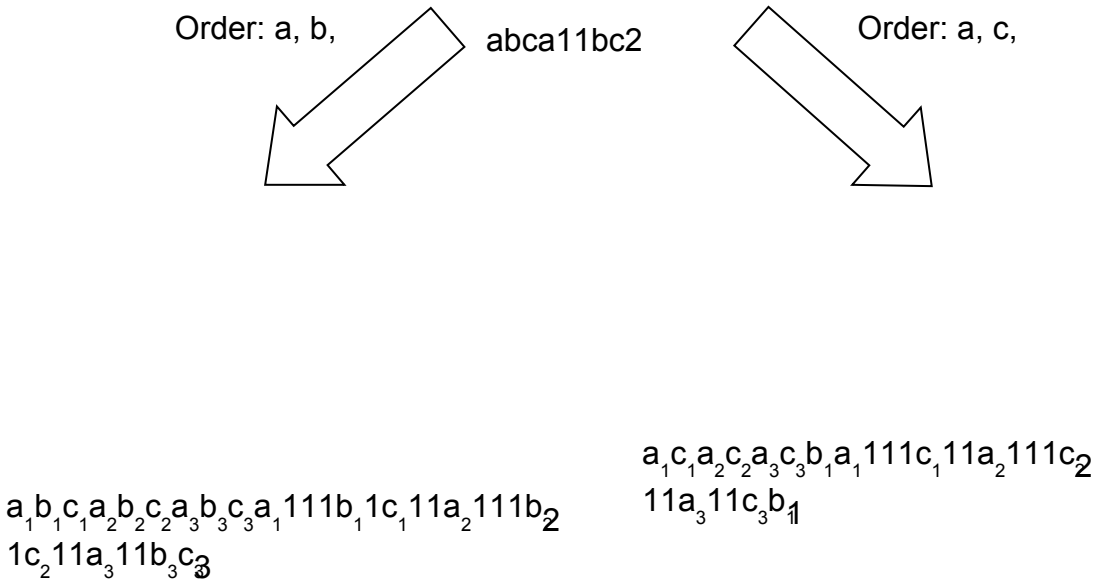

Figure 4: Iteration in a more complex case

3 RESOURCE ALLOCATION PROBLEM WITH HEURISTIC SIMULATION APPROACH

The basis of our approach is simulating the planning process in time. During the virtual planning process after each elementary allocation time unit some unfinished tasks using various allocation policies are activated in such a way that all previously discussed conditions still hold. We distinguish between two families of allocation policies. The first family is called as filtering, the second as ranking policies.

A filtering policy can activate or suppress a unit of unfinished tasks based on some strategy. A suppressed element cannot be scheduled at the given time. However, a favored task will get a higher priority and will be placed higher up in the list of schedulable tasks.

A ranking policy assigns a unique score between 1 to *n* to the *n* available tasks. The most important task will get the highest; the least important will get the lowest score. The available tasks are then activated based on their priority score. The tasks can be interrupted and there may be a delay in their execution [2].

There can be multiple versions of filtering and ranking policies, and they can be applied in combination in a given simulation context. In case of multiple filtering policies, we obtain the list of favored and suppressed elements by combining those

resulting from applying the individual policies. If a task appears in both the favored and suppressed lists (because of opposing policies) it will be remove from the favored ones and will be placed in the suppressed list. Finally, three disjoint sets are obtained: the favored, the suppressed and the normal tasks. Their semantics are identical to those we described in the case of filtering policies.

The case of multiple ranking policies is a slightly more difficult situation. One possible method for using multiple policies could be done by assigning weights to various policies, calculating the scores of tasks per policy and then summing these values with weights. These cumulative scores could then be used to rank the available tasks in a decreasing order. The main problem with this approach is that it can produce interference between to equally weighted policies: a given task may get an average score despite the fact that it was ranked as best by one policy and as worst by a conflicting other. This is not necessarily effective since despite being ranked on the top by one policy it is not activated due to averaging. The authors recommend another approach that uses only one policy at a given time unit. In this case, a policy would be chosen randomly after every time unit, although we may elect to assign different probabilities for each policy selection. This approach is free of interference between policies given that only one is applied at any time unit, and one can control the policy selection process by setting appropriate probabilities to used policies. However, this procedure, contrary to the previous approach, is not deterministic.

The simulation thus proceeds as follows. First, before each allocation time unit, the set of tasks that can be scheduled are determined. After this, using the filtering policies, the favored and suppressed tasks are selected and the ranking policies are applied to those that are not suppressed. Finally at first the favored than the normal tasks are selected for activation their order is based on the ranking. For each selected task, an attempt is made to allocate all the resources needed by the task. If there are not enough sufficient quantities of any resource category, the given tasks cannot be scheduled and will be skipped. If all resource needs can be fulfilled the available resources will be decreased accordingly and the given task will be activated. After this the algorithm proceeds to assign the remaining resources to the tasks with lower ranks.

The simulation can terminate in two ways: either all elements are scheduled within the total planning time or the total planning time is exceeded. In the former case a project plan can be created automatically, in the latter case the resource allocation problem cannot be satisfied, probably the conditions should be changed (more resources, longer total planning time, simpler DSM or perhaps applying different policies); this is always a management decision.

Number of interruptions filtering policy: This filtering policy favours those tasks that had been interrupted relatively more often than others. This way the fragmentation of a complex planning process can be avoided.

Completeness filtering policy: It favours those tasks that are almost (at least 90%, say) finished.

Critical path ranking policy: This algorithm ranks the available tasks based on their extent by which they can be shifted: the maximum amount of delay that does not extend the length of the critical path (longest predecessor chain).

Dominant resource needs ranking policy: Ranks the available tasks based on their resource needs where the task requiring the most resources is placed to the front.

Predictive policy: Assigns a probability to each task based on various considerations, expressing the likelihood that the task can be finished later on (it is able to obtain the resources needed to complete). If this probability is small, the task should be treated with a higher priority, as those with more chance for finishing are assumed to have more opportunities later on.

A multi-variable heuristic model for resource scheduling of constructional design processes have been worked out in a way that it can schedule the resource environment of development processes efficiently with the use of the examined policies – policies that filter the number of interruptions, completeness, as well as dominant resource needs ranking and predictive policy. The efficiency of the policy depends on the resource environment hence the possible solutions should always be checked with the combined use of policies [5].

4 CASE STUDY

For examination, a classical gear drive development process was chosen. A process consists of 24 tasks with some iterative sub-processes. The adequate DSM is shown in Figure 5. At this state, the order of the tasks is arbitrary thus it contains spare iterations. With the genetic algorithm-based optimization the optimal order of tasks can be achieved, the optimized DSM for gear drive development process can be seen on Figure 6.

Bearing calculations	Strength calculation with material properties	Drafting of parts	Requirement examination	Main design	First modification	End of project	Scaling review	Start of project	Bearing selection	Driven shaft checking	Controlling	Strength calculation shafts ...	Preconceptions	Secound modification	Design of gear parts	Main design review	Disposition versions	Market research	Documentation	Third modification	Validation	Disposition sketch	Detailed design of gear parts	Days	$	Σ
														1		1								14,06	937	6
														1		1								14,06	4685	6
						1																		34,39	6 878	4
																	1							1,9	950	2
														1		1								46,86	14 056	6
									1			1			1									3,8	1 900	2
																								5,695	0	8
1	1			1																			1	3,8	1 900	2
			1										1					1						1	0	1
																						1		10,32	1 719	4
																				1	1			15,65	4173	7
																				1	1			5,217	5 217	7
																						1		27,51	6 878	4
																	1							5	1 000	1
1	1			1																			1	4,686	2 342	6
																						1		27,51	6 878	4
										1	1									1	1			2,71	1 355	3
									1			1			1									9,5	2 850	2
																	1							5	1 500	1
						1																		27,51	3 439	4
										1	1										1			4,686	4685	6
		1																	1					13,55	5 420	3
					1		1																	7,6	1 900	2
														1		1								11,4	3 800	2

Figure 5: The DSM of gear drive development process with thriftless task ordered

Start of project	Market research	Requirement examination	Preconceptions	Disposition versions	Design of gear parts	Strength calculation shafts ...	Bearing selection	Disposition sketch	First modification	Scaling review	Detailed design of gear parts	Strength calculation with material properties	Bearing calculations	Main design	Secound modification	Main design review	Driven shaft checking	Controlling	Third modification	Validation	Drafting of parts	Documentation	End of project	Days	$	Σ
	1	1	1																					1	0	1
				1																				5	1 500	1
				1																				1	500	1
				1																				5	1 000	1
					1	1	1																	5	1 500	1
								1																15,2	3 800	2
								1																15,2	3 800	2
								1																5,7	950	2
									1	1														7,6	1 900	2
					1	1	1																	3,8	1 900	2
											1	1	1	1										2	1 000	1
															1	1								11,4	3 800	2
															1	1								5,7	1 900	2
															1	1								5,7	380	2
															1	1								19	5 700	2
											1	1	1	1										1,9	950	2
																	1	1	1	1				1	500	1
																			1	1				5,7	1 520	2
																		1	1					1,9	1 900	2
																	1	1		1				1,9	1 900	2
																					1	1		5	2 000	1
																							1	10	2 000	1
																							1	8	1 000	1
																								1	0	1

Figure 6: The DSM of gear drive development process with optimized task order

The optimized DSM is suitable to be used for automatic project planning fitting the process to a given human resource environment provided by the heuristic algorithm described before. Figure 7 shows a conceivable human resource environment for a month period. Each bar represents the cardinality of the related human resource on the given day.

The human resource environment shown in Figure 7 was elaborated in such a way that it results three days shift with respect to the beginning of the project, since we prescribed a FEM (finite element method) specialist for the first task, but from this human resource category, there is none available in the first three days. Further artificial resource problem was induced by limiting the amount of some necessary resources for a few days in the middle of the month. This way the affected task had to be interrupted and when these resources were once again available the task than could be resumed (Figure 8; Market research, Bearing selection (1)).

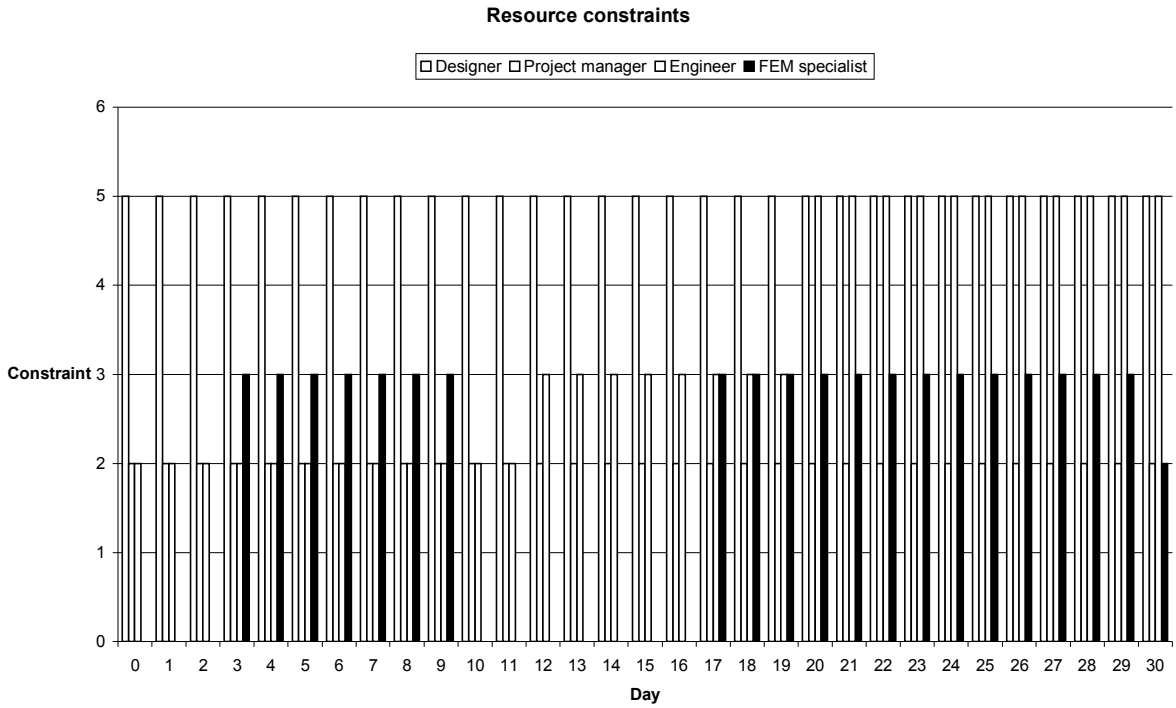

Figure 7: The diagram of an imaginary industrial human resource environment used in the case study

Furthermore, it is worth mentioning that our algorithm recognized those tasks that can be scheduled in parallel (e.g. Design of gear parts (1) and Strength calculation shaft and shaft couplings (1)), given that the available resources allow for such parallel execution. From the synthesized process plan a Gantt-diagram is produced: Figure 8.

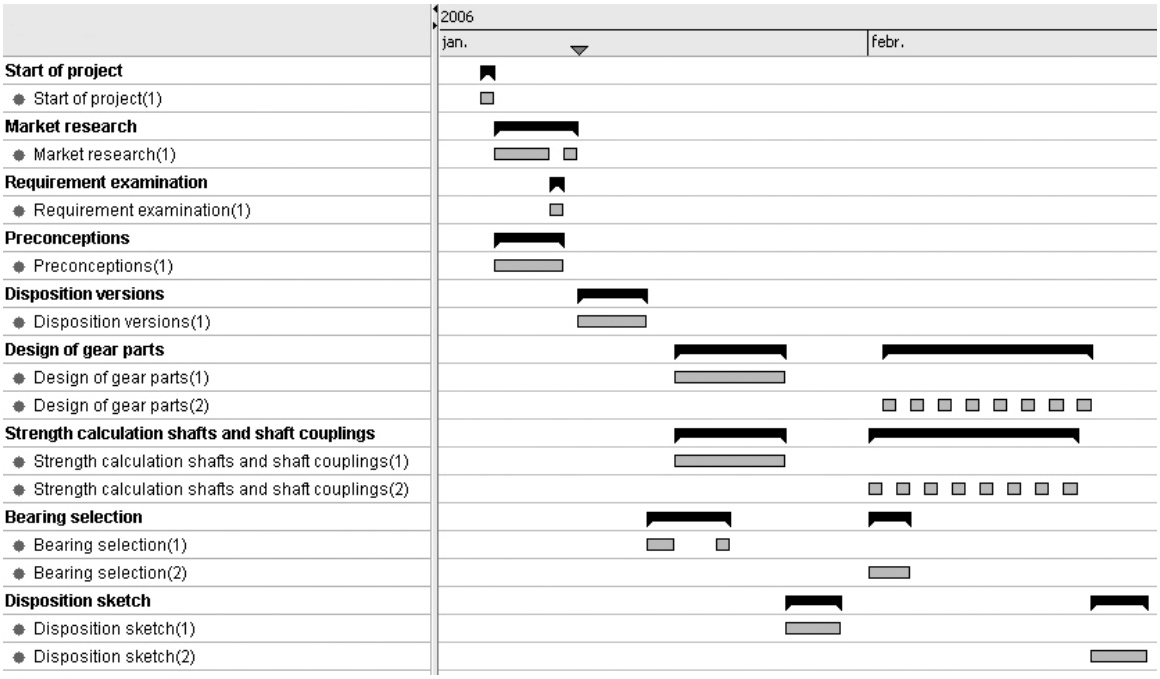

Figure 8: The final Gantt-diagram for the gear drive development process

ACKNOWLEDGEMENT

This research has been supported by the Hungarian Scientific Research Fund grant no. 68773 and the Hungarian Scholarship Board - German Academic Exchange Service Inter-University Partnerships grant no. 7.

REFERENCES

Bercsey, T., Groma, I., Rick, T., “Flexible and Effective Task Scheduling and Resource Allocation“ 17. Symposium „Design for X“, Merkamm, H., (Ed.), Neukirchen, 2006, pp. 95-105.

Buddhakulsomsiri, J., Kim, D., S., “Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting”, European Journal of Operational Research, vol. 175(1), 2006, pp. 279-295

Clement, S., „Erweiterung und Verifikation der Autogenetischen Konstruktionstheorie mit Hilfe einer evolutionsbasierten und systematisch-opportunistischen Vorgehensweise“, Dissertation, Buchreihe Integrierte Produktentwicklung, Band 7, Otto-von- Guericke-Universität Magdeburg, 2001

Reinertsen, D., G., „Die neuen Werkzeuge der Produktentwicklung“, Carl Hanser Verlag, München, Wien 1998

Rick, T., „Gépipari terméktervezsési folyamatok erőforrás- és költségszempontú optimálása a termékstruktúra figyelembe vételével“ Dissertation, Budapest University of Technology and Economics, 2007.

Steward D., V., “The Design Structure System: A Method for Managing the Design of Complex Systems”, IEEE Transactions on Engineering Management, vol. 28, 1981, pp. 71-74.

VDI-Richtlinie 2221 Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte, 1986 (Düsseldorf, VDI-Verlag)

Contact: István Groma
Budapest University of Technology and Economics
Department of Machine- and Product Design
Műegyetem quay 3.
1111, Budapest
Hungary
+36 (1) 463-40-81
+36 (1) 463-40-81
groma.istvan@gt3.bme.hu
http://gt3.bme.hu/