INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITÉ DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

TASK SCHEDULING AND RESOURCE ALLOCATION WITH MULTI-VARIABLE HEURISTICS

Tamás Rick[1], István Groma[2], Ádám Gránicz [3] and Tibor Bercsey [4]
[1,2,4]Budapest University of Technology and Economics, Institute of Machine Design
[3]IntelliFactory Ltd.

ABSTRACT
Nowadays product development processes are typically specified as bound 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. On the other hand, the workflow-based process description is only applicable to modeling simple recurring development activities that can only be treated as a whole, and it is impossible to specify and treat iterations that often accompany the development process in a dynamic way. Furthermore, the assignment of resources to the tasks of the given process is not automated, and it is not possible to adapt the process in case of a change in the requirements due to the rigid, pre-defined structure. Therefore, the use of work processes is only effective if a pre-defined standard, repetitive processes was used. However development processes are unique and valid in a given context that changes with time. Therefore it is advised to use a DSM, through which one can get a more precise picture of the product and the relationship of its structural elements during the development planning process. This approach supports the modeling of iterations that are quintessential in practice. With the help of genetic algorithm-based optimization one can find the one development schedule with the least development time and cost by changing the schedule of the product elements. As a result a project plan can be obtained – and for each element the needed resources can be assigned. But in most cases, the enterprise resource profile does not fully cover the requirements; therefore it becomes necessary to create a modified project plan keeping the optimized schedule, one that is suitable for the enterprise resource profile. For solving the problem of resource assignment a heuristic method has been created where the project plan is fitted to the enterprise resource setting by applying various strategies, focusing on minimizing the time requirements of the development project.

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

1 INTRODUCTION
The subject of product development is the product that is designed to fulfill some kind of need. 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.
Those definitions that are descriptive do not offer specific processes. Examples are Pugh's [1] Total Design theory, Suh's [2] Axiomatic Design, and Tomiyama and Yoshikawa's Genereal Design Theory [3], the TRIZ created by Altschuller [4], and Linde's and Hill's [5] WOIS – Contradiction-Oriented Innovation Strategy.
On the contrary, prescriptive definitions do give us the list of activities that need to be done in order to perform planning. Theories yielding to such processes were formulated by Hubka [6], Koller [7], Roth [8], Rodenacker [9], and Pahl and Beitz [10]. In Europe, the most widely accepted theory is the process suggested by VDI 2221 [11]. 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 time and cost. Numerous approaches were developed to reduce product development time, such as that of concurrent engineering [12], simultaneous engineering [13] and

frontloading [14], each of which attempt to optimize development time by parallel tasks, strengthening team work, and the early availability of information.
The theories and methods mentioned above do little to aid the planning and modeling of product-oriented design processes. A method is necessary that defines the design process based on and according to the structure and features of the product to be designed [15]. 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) [16].

2 DESIGN STRUCTURE MATRIX (DSM)

The DSM approach is based on the idea that one can change the order of activities based on the relationships among the design/development processes of the sub-components. By reordering, one can find a sequence of activities that may contain fewer cycles, and identify those activities that can be performed in parallel. This optimum signifies the most favorable assignment of total man hours and total cost, and the final project plan may take less time if there are activities that can be performed in parallel. When planning the relationships among product activities the following are useful.
The main activities A_i (for i=1,…,n) involved in a product identify the matrix shown in Figure 1. The diagonal elements (activities) represent themselves, and these are set to zero: a_{ij}=0 (for i=j). The remaining entries can be used to denote various relationships between the main activities. 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 és A_j . If for an element of this matrix it holds that a_{ij}=1 for i<j is called a feed forward relationship (and the entry lies above the diagonal), whereas for i>j the entry denotes a feedback relationship, representing a cycle (and lies below the diagonal). In the case of cycles, one can supply the planned number of cycles based on the actual ordering.
Furthermore, the elements of the matrix can be assigned numerous placeholders that make it more applicable to a wider range of problems. The approach of this paper, in its current form, handles the cost and time constraints specified by the management of the organization.
For scheduling and optimization of the various steps of the product development process a genetic algorithm has been chosen that makes it possible to quickly solve robust and extensive problems yielding an optimum solution that fits multiple criteria.

	A1	A2	A3
A1			1
A2	1		1
A3		2	

Figure 1. An example DSM

2.1 Iterations

Planning is an iterative process, and this should be taken into account while planning the development process as well. It often happens that one must re-plan certain elements because of unclear requirements, or at other times the type of the design itself requires to redefine the model in case of an FEM analysis. In such cases it is possible to plan in advance and tailor the development process accordingly. Figure 2 shows a process that contains one iteration which can be unfolded in a straightforward manner as shown in Figure 3.

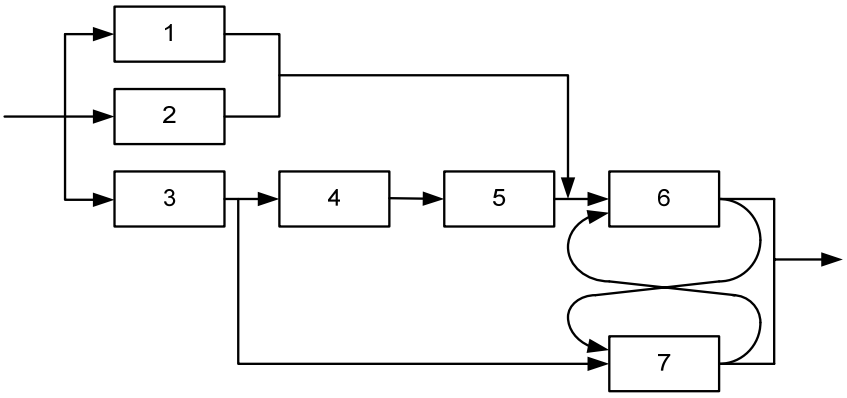

Figure 2. A fragment of an iterative process

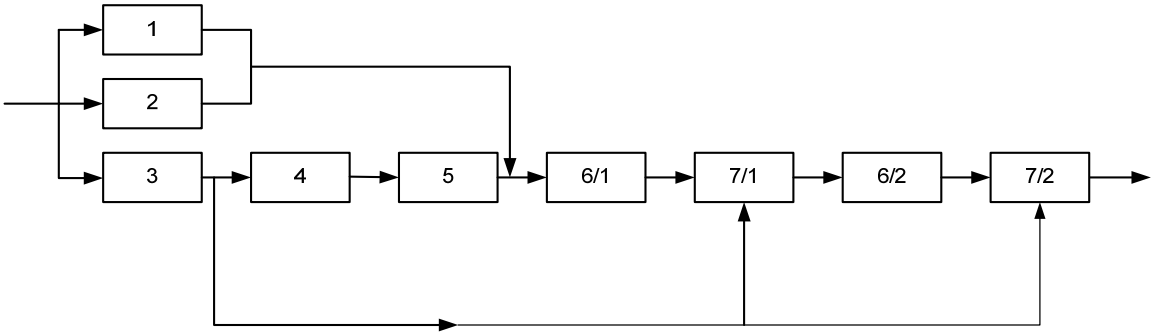

Figure 3. The unfolding of an iterative process segment

This unrolled, unfolded representation shown above is more informative and in fact indispensable for creating a Gantt diagram. Graph edges do not disappear during optimizing the activity order; instead one must locate the optimal place for the starting point of iteration found in the graph. One basic requirement of development projects is to be able to work with real information in every work phase. Therefore, there can be two different modes of executing the iteration in the graph shown in Figure 4 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.

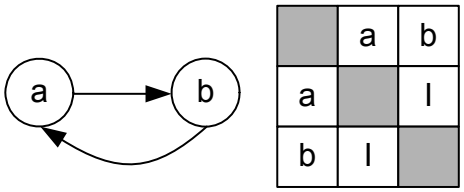

Figure 4. Representing an iteration

Figure 5 shows a DSM consisting of 3 activities. In case A, the order of task execution 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, nine activities must be performed.

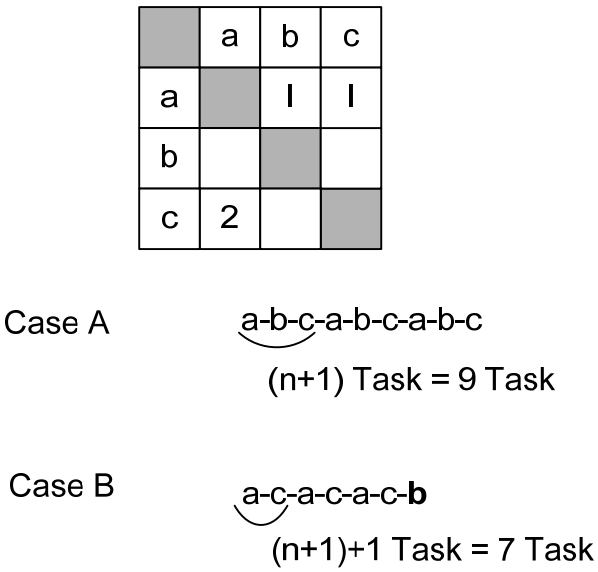

Figure 5. Representing an iteration in a more complex case

In case B, by rationalizing the order of activities to be executed only seven activities are needed. By doing so, time and money can be saved. The number 2 in the exercise designates the number of iterations needed. The first iteration step starts after finishing task *a* and *c*, the remaining number of iterations is one, and the process continues until the number of remaining iterations reaches zero.
The aim of this ordering is to find the optimal starting point of cycles in order to minimize the cost and time required for development. For this task a robust genetic algorithm was applied.

3 SCHEDULING USING GENETIC ALGORITHMS

We consider the basis of process modeling those main components that are developed either at the beginning (US Model) or the end (EU Model) of the conception phase. In the conception design phase, the relationships between product components are already understood, thus these can be represented in the DSM mentioned earlier. When defining the development order, the main goal is to reduce the total time and cost of the process.
The basic prerequisites of using the developed model are as follows:

- Knowledge of the product's structural make-up;
- Clarifying the relationships between the various structural elements, components;
- Knowledge of estimated development time detailed according to components or parts;
- Knowledge of estimated development costs detailed according to components or parts;
- The probability and number of possible development iterations.

Our model neglects the following aspects:

- Costs do not contain material costs, and only the estimated cost of expenditures of renewable resources are handled;
- The treatment ignores relationships between various components, and the detailed decomposition of the material-, energy-, and information flow of classic machine design.

During planning the product design process there is no need to list the product's components in a defined order based on the model developed, since this can not be projected in the case of a complex product. The random order set up as a result is represented in the first column and row of the DSM. It is a prerequisite that an order is assign to each activity so in order to identify them easily. Additionally, the cost and time can be specified intended for the development of each part or component.

3.1 Encoding and applied settings

Genetic algorithms typically do not work with the parameters to be optimized directly, but rather in a coded form (e.g. binary or gray coding), but in the case of order optimizing it is impractical [17]. In our case, a gene (a given order, a particular solution in the search space) is made up of the number of structural element indices (chromosomes) without encoding (as shown in Figure 6).

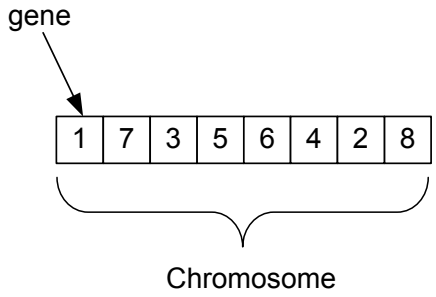

Figure 6. Encoding and naming

The size of the chromosome or the number of genes depends on the number of activities in the development process. The value of a particular gene is the index of the given task.
A tournament process was used for making selections [18]. The crossing used is a full crossing, and the mutation applied is a one-gene mutation. The definition of the given parameters and probabilities are discussed in [18]. In our treatment the following values are used:

- probability of crossing: 0.6
- probability of mutation: 0.8
- full crossing
- the size of population is ten times the size of the dimensions of the DSM

3.3 Evaluation

Defining the right target function is one of the necessary conditions for the success of any search task. In this case, the objective is to minimize the total cost and time of the development processes without disrupting the relationships between the elements of the process (the development of parts, or components). This can be reached by summing up the time and cost requirement of each task executed. One can expect different ordering depending on whether cost or time plays a more important factor in a given process. While totaling the effort, one should not forget about tasks that were executed multiple times due to iterations - and it is important that the learning rate was taken into account in such cases. In order to make a better approximation of reality, a learning rate modifying factor is introduced for calculating the time requirements of iterations. This means that if an activity is to be executed multiple times, its execution for the second or the any subsequent time can be accomplished faster than the first time. Another assumption is that it was never concluded that a given task is invalid during the development process. The previous condition reduces development time, as the development process doesn't have to start at a given element again from scratch but instead it is sufficient to make minor modifications to it.
The target function can be defined in a general form as follows:

$$f\left(\underline{n}_{\langle s\rangle}, \underline{l}, \underline{c}, \underline{t}, w_c, w_t\right) := \left[\frac{1-l_1^{n_1}}{1-l_1} \quad \cdots \quad \frac{1-l_m^{n_m}}{1-l_m}\right] \left[\underline{c} \quad \underline{t}\right] \begin{bmatrix} w_c \\ w_t \end{bmatrix} \qquad (1)$$

The explanations for the symbols in the above equation are summarized in Table 1 below:

Table 1.Symbols

$m \in N$	The number of development tasks
$\langle s \rangle \in Perm([1..m])$	The sequential order of the development tasks
$w_c, w_t \in R$	The weight of the cost (1/€) and time (1/h) in the target function
$\underline{c} \in R^m$	The vector of the costs associated with the development tasks
$\underline{c} \in R^m$	The vector of the time lengths associated with the development tasks
$l \in [0,1)^m$	The vector of learning rates relative to each structural element
$\underline{n}_{\langle s\rangle} \in N^m$	The vector containing the repeat count of the development tasks, given a particular production order $\langle s \rangle$

The formulation of the problem then is:

$$\langle s_0 \rangle \in Perm([1..m]) : f\left(\underline{n}_{\langle s_0 \rangle}\right) \approx \min_{\langle s \rangle} f\left(\underline{n}_{\langle s \rangle}\right) \quad (2)$$

$$\left|Perm([1..m])\right| = m! \quad (3)$$

As the genetic algorithm proceeds it determines using the above formula the fitness value of all elements in a given population. The order found in each element defines the order of development. This order is random, thus each element contains a different order. As a consequence, there are different cycle numbers in each element, which means that the same development task should be executed a different number of times as encoded by one element or another. Let us consider the cost value and development time associated with each component. Based on the value of the target function, the GA selects the best (best fitness), and the worst one (worst fitness). In our treatment, the best value is represented by the lowest "best fitness" value. This function determines the total number of man hours and costs, and there is no connection made with the real running time of project, as parallel tasks and repetitions caused by iterations are not handled by date, e.g. if the time value of the target function was added to given a date, that would lead to an unrealistic result.

4 THE HEURISTIC METHOD OF RESOURCE ALLOCATION

The genetic algorithm computes a development process matrix optimized by total time that can be turned into a predecessor graph by applying an unfolding strategy. The nodes of this graph are numbered by the version numbers of the various development processes, each denoting the number of repetitions of the given task. The directed edges between the nodes designate the information flow between tasks as defined above. Cycles from the original graph disappear as a result of the unfolding process. The main goal of the preliminary, genetic algorithm-based optimization is to reduce the repetitions of various tasks by optimizing their order – and as a result to reduce the number of nodes in the predecessor graph.

By exploiting the scheduling of parallel activities the total time of the development project can be decreased. Two development tasks can be scheduled to run in parallel if neither of them provides information to the other. This can be checked in the predecessor graph by examining whether a directed path exists from one task to the other. Ultimately, our main goal is to shorten the total development time by achieving an optimal scheduling of tasks where the number of parallel activities is maximized, but still adhering to the relationships contained in the predecessor graph.

It is guaranteed that there exists an optimal solution to the above problem, one that produces the best scheduling possible. By the use of the critical path algorithm [19], it is possible to construct the optimum solution in polynomial time – and we can refer to this as the shortest scheduling. It is important to note that the shortest scheduling is not unique; as tasks can be delayed in their free buffer time without prolonging the project as shown in Figure 7.

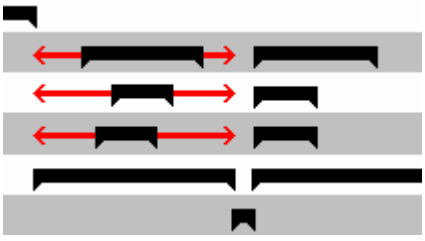

Figure 7. Shifting activities within their buffer time.

One can get an even more realistic and practical model if - when planning parallel tasks – it is taken into account that development tasks done in parallel need a larger resource volume. In our treatment, resources stand for renewable resources, mainly human ones. To simplify matters, our model do not deal with concrete individual resources, but instead treat them as various types of resources, and it is also assumed that two resources of the same type can substitute each other, but two different types cannot.

If the resource needs of the development activities are known, namely those belonging to different types of activities, then one can determine from the schedule calculated by the critical path algorithm how much of a given resource type is needed at any given time in order to run our product development project without complications. Here our treatment is simplify by assuming that a given development task has a constant resource need for its entire duration.
Now the minimum level of each resource type can be easily defined for every time period. On the other hand, it is possible that the needed quantities of a given resource type can not be either fully or partially fulfilled from the resource profile of the enterprise. In this case, the scheduling calculated by the critical path algorithm must be sacrificed, but this raises the problem of generating the shortest time scheduling obeying the enterprise resource availability.
In the case of scheduling without resource assignments, the problem of having to abort tasks does not surface since interruption of them are not forced due to some constraint. On the other hand, in a given enterprise resource context one may find it practical or even unavoidable to interrupt tasks at various time intervals.
Therefore, the scheduling problem is defined as follows: the time-wise shortest scheduling is required on a discrete time scale allowing for the tasks to be interrupted, but the resources assigned to each time step can not be exceed the available limits, and each subsequent task has to be fulfilled on time. This is defined formally below:

Table 2. Symbols

$[1..T], T \in N$	A discrete time scale; the scheduling range in focus
J	The set of versioned activities
$K \in N$	The index of the different resource types. Each resource type represents a new dimension in K-dimension space.
$\mathbf{R}:[1..T] \to (N_0 \cup \{\infty\})^K$	The vector of resource assignments for each time step
$\mathbf{r}: J \times [1..T] \to {N_0}^K$	The vector of resource needs of the given activity.
$a: J \times [1..T] \to \{0,1\}$	The activity function; for a given activity it returns 1 if active, and 0 otherwise
$d: J \to [1..T]$	The time length of a given activity
$\to \subset J \times J$	The follow relation (transitive, irreflexive, and anti-symmetric)

$$\sum_{j \in J} \mathbf{I}^{a(j,t)} \mathbf{r}(j) \le \mathbf{R}(j)$$

$$A_j^a := \{t \in [1..T] \mid a(j,t) = 1\} \qquad (4)$$

$$\left|A_j^a\right| = d(j)$$

$$\forall j,k \in J : j \to k \Rightarrow \max A_j^a < \min A_k^a$$

$$g(a) := \max_{j \in J}(\max(A_j^a)) \quad g(a^*) = \min_a g(a) \qquad (5)$$

Two different approaches were applied to arrive at optimal scheduling. First the problem was defined as a 0-1 discrete linear programming problem (ILP), where the search space is made up of indicator variables [21]. Finally our decision was to reject this solution due to the large size of ILP matrix and the NP-complete nature of the approach [20].
Our second approach is based upon temporal simulation of the development project. Each time period it was examined that which activities are the most feasible targets for scheduling. This is done by having various scheduling policies compete with one another. Since each policy represents a different heuristic approach, this method is a heuristic one. In this treatment, it was assumed that the tasks that are candidates for scheduling are competing for resources available in the next given time period. The "distribution" of resources happens according to priority; first the algorithm tries to schedule those tasks with the highest priority. If the right amount resources are available, the task becomes active and the needed resources are assigned to it; otherwise it becomes inactive in the next time period. The

method attempts to schedule all activities according to their priorities; therefore the ones with the highest priority are more likely to become active in the next time period. Scheduling policies determine the priorities of each activity. During the simulation multiple policies can be applied at any given time step. Every policy scores the available activities based on the importance of their scheduling; that with the highest priority gets the highest score, the second gets one point less, and so on. Two different methods were applied for mixing policies. One sums the weighted scores given by the various scheduling policies to arrive at a particular ordering, the other method uses a random technique to choose a policy at every time step to create the scheduling order. This latter approach produces a stochastic result.
The simulation ends successfully if all available tasks are scheduled entirely, or it exits unsuccessfully if the given project time constraint is exceeded.
The pseudo code for the algorithm using the symbols outlined in our discussion is shown below. The `sort`$_{p1,...,pn}$`()` function generates the scheduling order based on policies p_1,...,p_n and the `finished()` function examines whether a given activity has been finished or not. The `schedule(S,t)` function schedules a given set of activities for the next time period.

```
A <- J
t <- 1
while (A != {}) and (t < T)
        F <- A
        for j,k in A
                if j -> k then
                        A <- A \ {k}
        S <- {}
        rl[] <- R(t)
        for f in sortp1,...,pn(F)
                nr[] <- r(f)
                if nr[] < rl[] then
                        S <- S + {f}
                        rl[] <- rl[] – nr[]
        schedule(S,t)
        for s in S
                if finished(s) then
                        A <- A \ {s}
        t <- t + 1
if A = {} then
        found()
else
        error()
```

Figure 8. Pseudo code for the main algorithm

It is provable that given an unconstrained resource profile, all scheduling policies will produce the same solution using the critical path algorithm. Nevertheless, it is worth noting that it is not necessarily true that if a resource profile satisfies the shortest scheduling it will be found by the algorithm, as this is dependant on the policy used.
It is no surprise that the essence and success of the algorithm is determined by appropriately defining policies. The ones used by us with a short description are shown in table 3.

Table 3. Policies

Critical path policy	Ranks the activities based on their free buffer time. It favors those that have less buffer time since if these have to be postponed they are likely to affect the total project time.
Predictive policy	Assigns a probability to each activity based on various considerations, expressing the likelihood that the activity can be finished later on (it is able to obtain the resources needed to complete). If this probability is small the activity is treated with a higher priority, as those with more chance to finish are assumed to have more opportunities later on.

Optimal policy	The ideal policy is a theoretical one that is always able to schedule optimally, and adjusts the ordering at any given time to this optimal schedule. It provides the best results of any policy, but there is no known concrete and efficient algorithm that implements it, thus it remains only a reference.

5 CASE STUDY

For examination a classic gear drive development process was chosen. A process consists of 24 tasks with some iterative sub-processes as shown in Figure 9. With the genetic algorithm-based optimization the optimal linear order of the design tasks can be achieved (Figure 10 and Figure 11). The resulting order can be used for automatic creation of a project plan fitting the defined resource environment as shown in Figure 12, provided by the heuristic algorithm described before.

The defined resource environment was prepared in such a way that it resulted in a three-day shift with respect to the project start at Jan 1, 2006, since a FEM specialist was prescribed for the start of the project, but from this resource category there was no one available in the first three days (Figure 12). The second artificial resource problem was induced by taking out some needed resources for a few days. This way, the affected activity was interrupted, and when the needed resources were once again available the activity was resumed (Figure 13; Market research, Bearing selection (1)).

Furthermore, it is worth mentioning that our algorithm recognized those activities that can be scheduled in parallel, and it shows them in the Gantt diagram (Figure 13; Design of gear parts (1) and Strength calculation shaft and shaft couplings (1)) accordingly, given that the available resources allow for such parallel execution. The critical path ranking policy algorithm can also be effectively applied. The resource allocation essentially reveals a time unit-based simulation, where the completion times of each simulation can be supplied, which in our case is one day. Upon completing a given simulation the algorithm re-computes the critical path and assigns resources to those elements that are on this critical path before any other. A good example for such is the Design of gear parts (2) and Strength calculation shaft and shaft couplings (2).

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 9. The DSM of an unstructured ordered gear drive development process

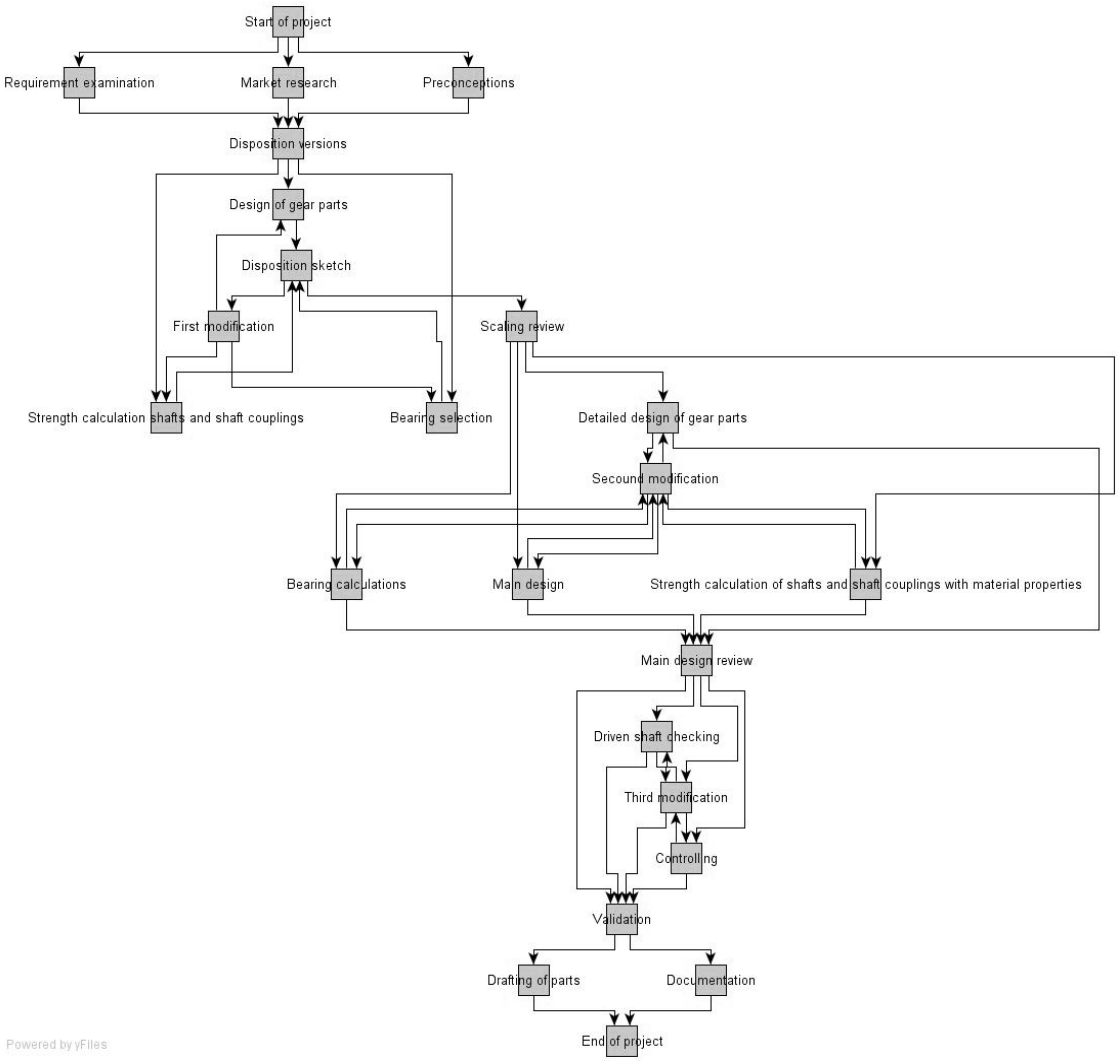

Figure 10. The process graph of the optimal development process

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						1,9	1 900	2
																					1	1		5	2 000	1
																							1	10	2 000	1
																							1	8	1 000	1
																								1	0	1

Figure 11. The DSM of the optimal development process

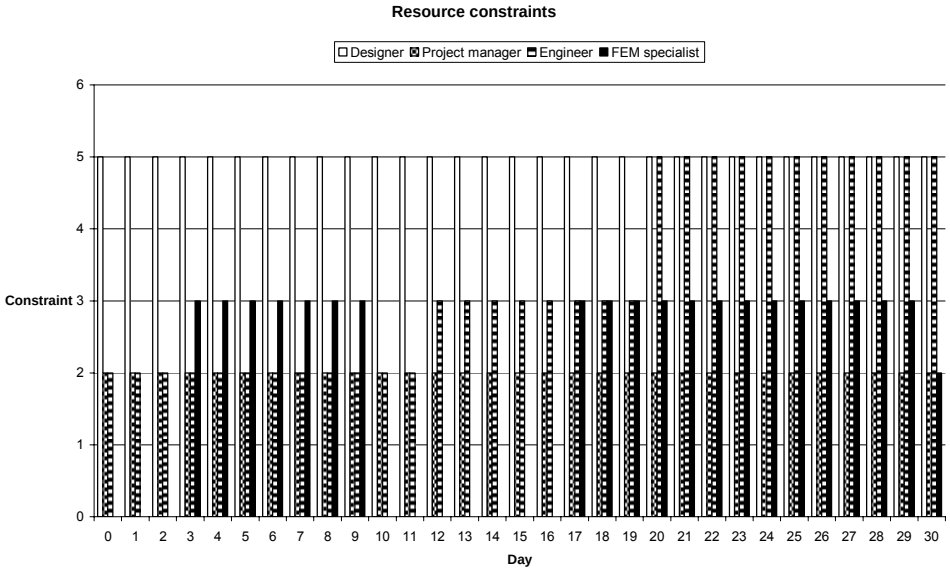

Figure 12. A possible resource profile diagram of a design company

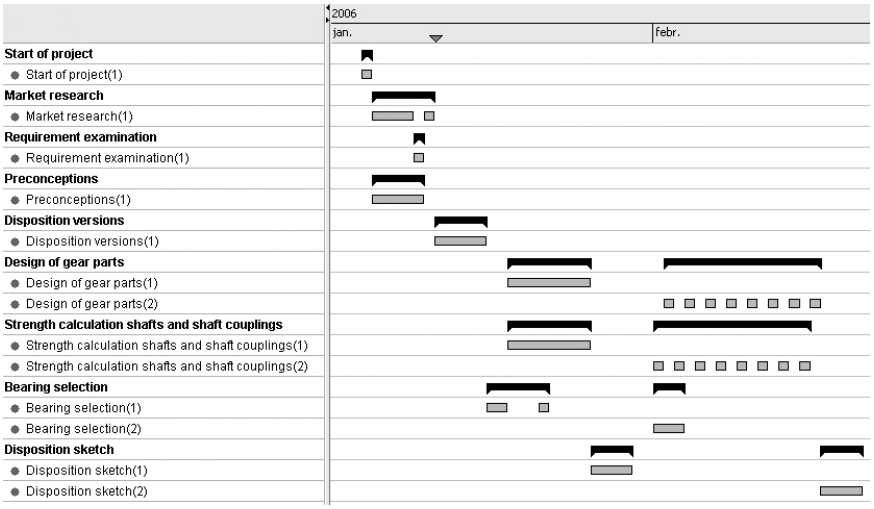

Figure 13. The final project plan for the gear drive development process

6 CONCLUSION AND FURTHER WORK

Current work is underway to automate the assignment of the resource profile (which contains resource categories that participate in the allocation process) to the human resource registry of the business. Further goals are to consider additional resource parameters and to devise refined allocation policies that simulate more realistic scenarios. Another fruitful endeavor is reducing the DSM complexity based on cluster analysis.

REFERENCES

[1] Pugh S. *Total Design, Integrated Methods for Succesful Product Engineering*, 1990 (Addison-Wesly).

[2] Suh N. P. *Axiomatic Design: Advances and Applications*, 1999 (Oxford University Press).

[3] Tomiyama T. and Yoshikawa H. *Extended General Design Theory*, 1988 (Reports CS-R8604, Centre of Mathematics and Computer Science Amsterdam).

[4] Altschuller G.S. *Erfinden – Wege zur Lösung technischer Probleme*, 1998 (VEB Verlag Technik Berlin).

[5] Linde H. and Hill B. *„Erfolgreich erfinden, Widerspruchsorientierte Innovationsstrategie für Entwickler und Konstrukteure“*, 1993 (Hoppenstedt Technik Tabellen Verlag Darmstadt).

[6] Hubka V. *Allgemeines Vorgehensmodell des Konstruierens*, 1980 (Fachpresse Goldach Zürich).
[7] Koller R. *Konstruktionslehre für den Maschinenbau*, 1985 (Springer, Berlin).
[8] Roth K. *Konstruieren mit Konstruktionskatalogen*, 1982 (Springer, Berlin).
[9] Rodenacker W. G. Neue Gedanken zur Konstruktionsmethodik, *Konstruktion*, 1991, 43, pp.330-334.
[10] Pahl G. and Beitz W. *Konstruktionslehre*, 1986 (Springer-Verlag Berlin, New York, Heidelberg).
[11] VDI-Richtlinie 2221 *Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*, 1986 (Düsseldorf, VDI-Verlag).
[12] Young R. E. and O'Grady P. Constraint Nets for Life Cycle Engineering: Concurrent Engineering. In *Proceedings of the 1992 NSF Grantees Conference*, Atlanta, January 1992.
[13] Eversheim W. and Bochtler W. and Laufenberg, L. *Simultaneous Engineering*, 1995 (Springer-Verlag, Berlin, Heidelber).
[14] Eigner M. and Zagel M. and Weidlich R. The Conceptual Product Structure as Backbone of the Early Product Development Process. In *Proceedings ProSTEP iViP Science Days 2005*, Darmstadt, September 2005.
[15] Donald G. Reinertsen *Die neuen Werkzeuge der Produktentwicklung*, 1998 (Carl Hanser Verlag, München, Wien).
[16] Steward D. V. The Design Structure System: A Method for Managing the Design of Complex Systems, *IEEE Transactions on Engineering Management*, 1981, 28, pp.71-74.
[17] Altus S. S. and Kroo I. M. and Gage P. J. A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems, *Journal of mechanical design*, 1996, 118(4), pp.486-489.
[18] Rick T. and Horváth M. and Bercsey T. Desgin Tasks Scheduling Using Genetic Algorithms, *Periodica Polytechnica, Mechanical Engineering*, 2006, 50(1), pp.41-55.
[19] Reichert O. *Netzplantechnik*, 1994 (Vieweg).
[20] Buddhakulsomsiri J. and Kim D. S. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting, *European Journal of Operational Research*, 2006,175(1), pp.279-295.
[21] Bercsey T. and Rick T. and Groma I. and Gránicz Á. GA-Based Flexible and Effective Task Scheduling and Resource Allocation. In *WMSCI 2006 Proceedings: CD and Vols. 1*, Orlando, July 2006, pp.119-124.

Contact: Tamás Rick
Institute of Machine Design/Budapest University of Technology and Economics
Department of Industrial Product Design and Agricultural Engineering
Műegyetem rkp. 3.
1111, Budapest
Hungary
+36 (1) 463-40-81
+36 (1) 463-40-81
rick.tamas@gszi.bme.hu
http://www.gszi.bme.hu.