17. SYMPOSIUM „DESIGN FOR X“
NEUKIRCHEN, 12. UND 13. OKTOBER 2006

GA-BASED FLEXIBLE AND EFFECTIVE TASK SCHEDULING AND RESOURCE ALLOCATION

Tibor Bercsey, István Groma, Tamás Rick, Ádám Gránicz

Abstract

The paper deals with the resource limited optimization of optimal sequence constructional design processes represented in a Design Structure Matrix (DSM). The method based on genetic algorithms, applied for the optimization of the sequence of constructional processes is introduced. Then a discrete linear programming and a heuristic simulation method is detailed to solve the resource allocation problem.

1 Introduction

Work process management sub-systems in current process control and Product Data Management (PDM) systems are shaped by various quality assurance standards. As a result, these sub-systems are inherently bound and limited, ignore efficiency, and often are not well-suited to a given planning task [8], [9]. Furthermore, a typical shortcoming is that they do not adapt easily to activity-based scheduling and human resource allocation. As a result, for the typical project planning parameters (time, cost, and resources) they can provide coarse estimations only [14].

The subject of the planning activity (product development, software engineering, construction planning) is usually a complex and semi-structured problem that involves iterations (it requires repetitious execution of the same tasks or group of tasks) in contrast to production planning processes where there are no repetitive work steps. The planning problem at hand should be dissected into sub-problems in such a way that these reflect the structural make-up of the desired outcome and match the organizational structure of the business [8], [9]. In view of the previous two conditions, our approach makes it possible to schedule renewable resources in an optimal and flexible way driven by multiple considerations. On one hand, optimal, because the total expenditures and total time that can be spent on the planning phase are derived from the available resources. On the other, flexible, because the planning process breakdown, the grouping of available resources, their needs and constraints and the overall resource allocation strategy can be fully customized.

We use the Design Structure Matrix (DSM) approach to represent the structural breakdown of the planning problem (a given product, for instance) itself. This choice is made because DSM is widely used with a long history of applicability in engineering (it is part of various CAD systems), it is adequately versatile, and it can be embedded in IT solutions easily [2], [10].

2 Design Structure Matrix

The DSM approach is based on the idea that one can change the order of activities based on the relationships among the design processes of 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 favourable 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 (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 and A_j. If for an element of this matrix it holds that a_{ij}=1 for $i<j$, then we speak of 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 we present in 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 planning process we had chosen a genetic algorithm that makes it possible to quickly solve robust and extensive problems yielding an optimum solution that fits multiple criteria.

▨	A	b	c
a	▨	1	
b		▨	1
c		1	▨

Fig. 1: An Optimal DSM.

3 Activity scheduling via genetic algorithms

The DSM developed by a typical planning professional is far from optimal: the time 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 using a genetic algorithm we compute a DSM that reflects the optimal activity breakdown which provides the best time and cost combination. The inputs of this optimization step are the activities and their time, cost and resource needs.

In the method described here a chromosome (a potential solution instance in the genetic algorithm) corresponds to a particular activity ordering where the genes contain the index of the activity. The genetic algorithm uses the usual operators: selection (binary contender selection), crossing (partial crossover), and mutation (per gene). The probability values that drive these operators are discussed in [2].

3.1 The Unfolded DSM

A DSM optimized by order may still contain feedback entries (this is inevitable in the case of cycles). We would like to eliminate these before the resource allocation process, as they complicate the task of resource assignment, and furthermore no project plan can be made in their presence. Therefore, we extend the original DSM by a new transformation that yields an equivalent representation but without feedback links. The basic idea is to record into the

DSM multiple versions of those activities that are part of cycles based on how many times they repeat in the planning process [1]. These activities will become separate rows and columns in a new, larger than the original DSM. This algorithm is referred to as DSM unfolding. Naturally, in an unfolded DSM each version of an activity will supply information to the version that follows it, since repeating a given activity can only be performed upon finishing the previous one. The feed forwards are embedded in the new DSM as is, whereas the feedbacks are transferred between the appropriate versions of linked elements only.

▨	a(1)	b(1)	c(1)	b(2)	c(2)
a(1)	▨	1			
b(1)		▨	1	1	
c(1)			▨	1	1
b(2)				▨	1
c(2)					▨

Fig. 2: The corresponding unfolded DSM.

This approach essentially dissects a given planning step into multiple activities according to how many times it is repeated in the original formulation. As a result, we obtain a new DSM that contains only feed forwards that basically constitute the predecessor network used during resource scheduling. For the sake of simplicity, we will uniformly call each version of an activity a separate activity, or simply DSM elements.

4 Resource allocation problem

The resource allocation problem deals with assigning the necessary amount of renewable (primarily human) resources to the various activities. We assume that the starting time *S*, the total planning time *T* (the practical time constraint on the planning process), the duration of each activity ($t_1,\dots,t_n$), the resource constraints, and the predecessor network (in our case the unfolded DSM) is given. The resources are supplied in resource categories, these being the basic units of the resource allocation process, in other words the output will not be a person-based allocation. Introducing resource categories decreases complexity as we are dealing with fewer entities, and they also yield to a more flexible project plan. Management can freely decide how the resource needs appearing at a given time in the project plan will be served from the available human resource pool. The resource category needs for each activity, and the quantity of resources in each category are pieces of input as well. These constraints can vary in time, and our model allows for their absence as well (for instance the allocation process can assume unlimited resources). However, the resource categories can not be substituted with another in any ways or as a result of any rules.

Initially, we represent the total planning time *T* as *d* unit length time intervals, for the sake of simplicity in such a way that the durations of each activity are multiples of this time unit. We call *d* the elementary allocation time unit. The resource allocation problem is executed in such a way that in each allocation time unit we determine what tasks (activities) are active. Any solution to the allocation problem then should conform to the following criteria:

- More than one task can be active in any given time unit.

- The cumulative resource needs of the active tasks can not exceed the resource constraints for that allocation time unit.

- The tasks can be interrupted and there may be a delay in their execution.
- A given task is active exactly $\frac{t_i}{d}$ times.
- During scheduling we respect the predecessor network, in other words if activity a_{ij} precedes a_{kl} then the last unit of a_{ij} will happen earlier than the first unit of a_{kl}.
- All activities finish before $S+T$.

The goal is to find the solution with the shortest effective time (the end of the last activity) given the resources available at each time period.

5 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 we activate some unfinished activities using various allocation politics in such a way that all previously discussed conditions still hold. We distinguish between two families of allocation politics. We called the first family filtering, and the second ranking politics.

A filtering politics can activate (favour: place before others) or suppress a unit of unfinished activity based on some logic. A suppressed element can not be scheduled at the given time. However, a favoured element will get a higher priority and will be placed higher up in the list of unit activities that can be scheduled.

A ranking politics assigns a unique score between 1 to *n* to the *n* available activities. The most important activity will get the highest score, and the least important one will get the lowest score. The available activities then are activated based on their priority score.

There can be multiple versions of filtering and ranking politics, and they can be applied in combination in a given simulation context. In case of multiple filtering politics we obtain the list of favoured and suppressed elements by combining those resulting from applying the individual politics. If a given element appears in both the favoured and suppressed list (as a result of opposing politics), we remove it from the favoured ones and place it in the suppressed list. Finally, we obtain three disjoint sets: that of the favoured, the suppressed and the normal elements. Their semantics are identical to those we described in the case of filtering politics.

In the case of multiple ranking politics we have a slightly more difficult situation. One possibility to proceed would be to assign weights to the various politics, calculate the scores of all elements per politics, and then sum these using the weights. These cumulative scores could then be used to rank the available activities in a decreasing order. The main difficulty with this approach is that it can produce interference between policies: a given activity may get an average score despite the fact that it ranked the best by one politics and the worst by a conflicting other, provided that the two politics are weighted equally. This is not necessarily effective since despite being ranked on the top by one politics it is not activated due to averaging. An alternative approach would be to use one politics at any given time unit. In this case, we would choose a politics function at random, although we may elect to assign different probabilities for each politics selection. This approach is free of interference between politics given that only one is applied at any moment, and we can control the politics selection process by setting the appropriate probabilities for selecting a given politics function. This procedure, however, contrary to the previous approach, is not deterministic.

The simulation thus proceeds as follows. First, before each allocation time unit we determine the set of activities that can be scheduled. After this, using the filtering politics, we compute the favoured and suppressed elements (activities), and apply the ranking politics to those that are not suppressed. Finally, we iterate through the favoured and then the normal elements based on the ranking we obtained. For each element we attempt to allocate all the resources that are needed, if we do not have sufficient quantities of a resource category the given activity can not be scheduled. Provided that all resource needs can be fulfilled, we decrease the available resources accordingly and activate the given task. Then we proceed to assign the remaining resources to the elements with lower ranks.

The simulation can end in two ways: either all elements are scheduled within the total planning time, or we exceeded the total planning time. In the former case, we can create a project plan (for instance a Gantt diagram) from the final solution, in the latter case we conclude that the resource allocation problem can not be satisfied, and probably we need to loosen the conditions (need more resources, longer total planning time, simpler DSM, or perhaps applying different politics functions); this is always a management decision.

5.1 Number of interruptions filtering politics

This filtering politics favours those activities that had been interrupted relatively more often than other activities. A break counter is assigned to each activity and the activities are ordered by the cardinality of interruption. This way we can avoid the fragmentation of a complex planning process.

5.2 Completeness filtering politics

Generally many scheduling techniques are trying to assign resources to short activities in order to finish easy task as quickly as possible. This way the relative waiting time (effective working time / total time between start and finis) can be kept at a good value. Hence this politics favours those activities that are almost (at least 90%, say) finished because they need relatively short time to be completed. This politics doesn't differ between tasks beyond 90% readiness or tasks beneath this rate it only deals with two seceding categories: nearly ready and ongoing tasks.

5.3 Critical path ranking politics

Examining the critical path of a process is very common method in project planning. Every process has a critical path: this is the longest task sequence in the whole process graph. The total time of the design process cannot be shorter than the cumulative time of the critical path. This algorithm ranks the available elements based on their maximal buffer time (MBT): the maximum amount of delay that does not extend the length of the critical path (longest predecessor chain) [4]. It is obvious that MBT is 0 for tasks on the critical path and the higher value this property of tasks has the less it influences the total time of design. There are some quick algorithms for calculating the MBTs [4]. During the scheduling the critical path of the remaining process may change so it is worthy of note that MBTs should be recalculated after every allocation time in the simulation.

5.4 Dominant resource needs ranking politics

Since it is possibly that a particular resource is only available for a short amount of time it is important to exploit the period best. To embed this kind of wise into our simulation model the dominant resource needs ranking politics was developed. This politics ranks the available activities based on their resource needs, where the activity requiring the most resources is placed in the front. In this manner resource demanding tasks are having the benefit of getting the required resources immediately it is possible according to the resource environment.

6 Case study

We chose a classic gear drive development process for our examination. A process consists of 24 tasks with some iterative sub-processes (fig. 3). With the genetic algorithm-based optimization the optimal linear order of the design tasks can be achieved (fig. 4, 5). The resulting order can be used for automatic creation of a project plan fitting the defined resource environment (fig. 6) provided by the heuristic algorithm described before (fig. 7).

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

Fig. 3: The DSM of an unstructured ordered gear drive 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				1,9	1 900	2
																				▨	1	1		5	2 000	1
																					▨		1	10	2 000	1
																						▨	1	8	1 000	1
																							▨	1	0	1

Fig. 4: The process graph of the optimal development process.

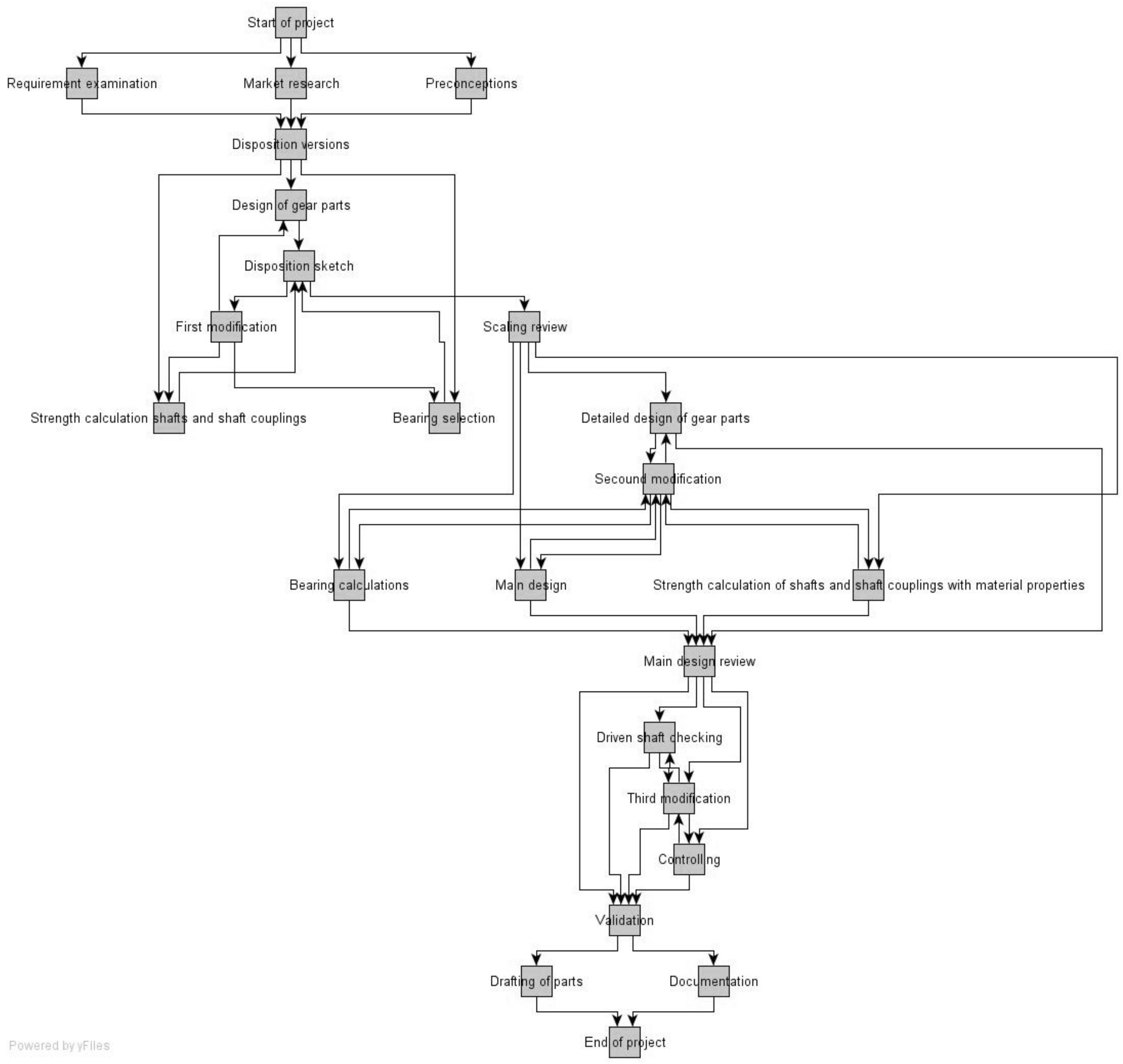

Fig. 5: The DSM of the optimal development process.

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 we prescribed a FEM specialist for the start of the project, but from this resource category there was none available in the first three days (fig. 6). The second artificial resource management 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 (*Market research* and *Bearing selection (1)*).

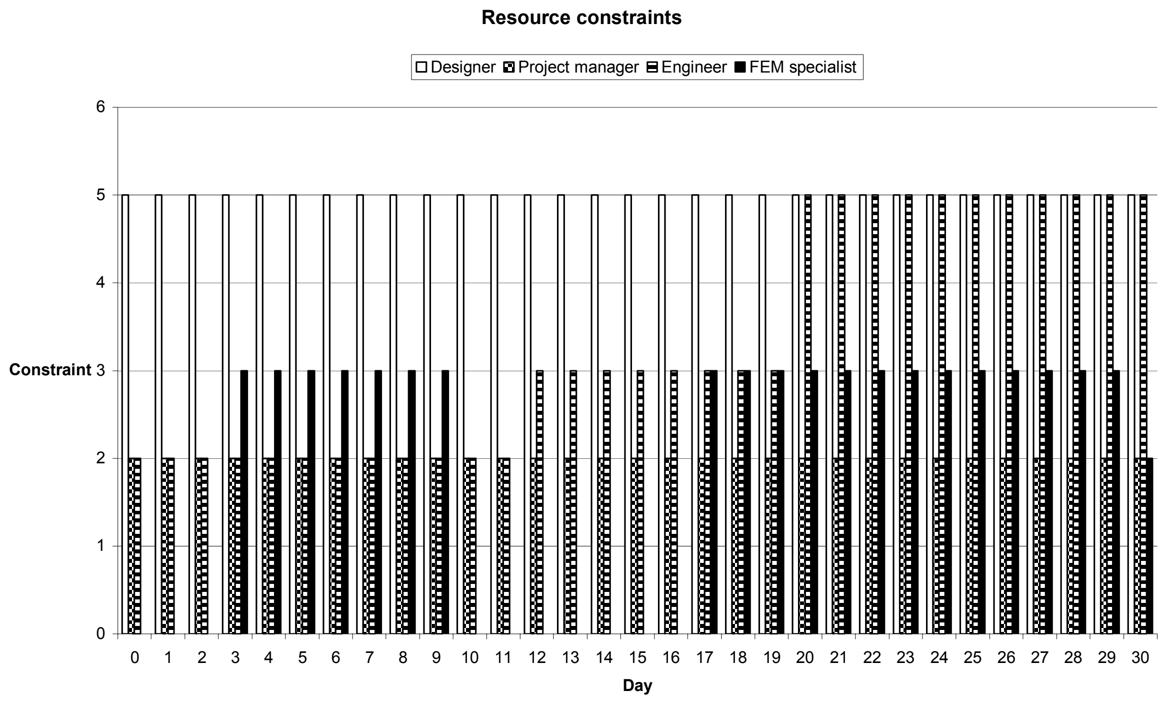

Fig. 6: A possible resource environment diagram of a design company.

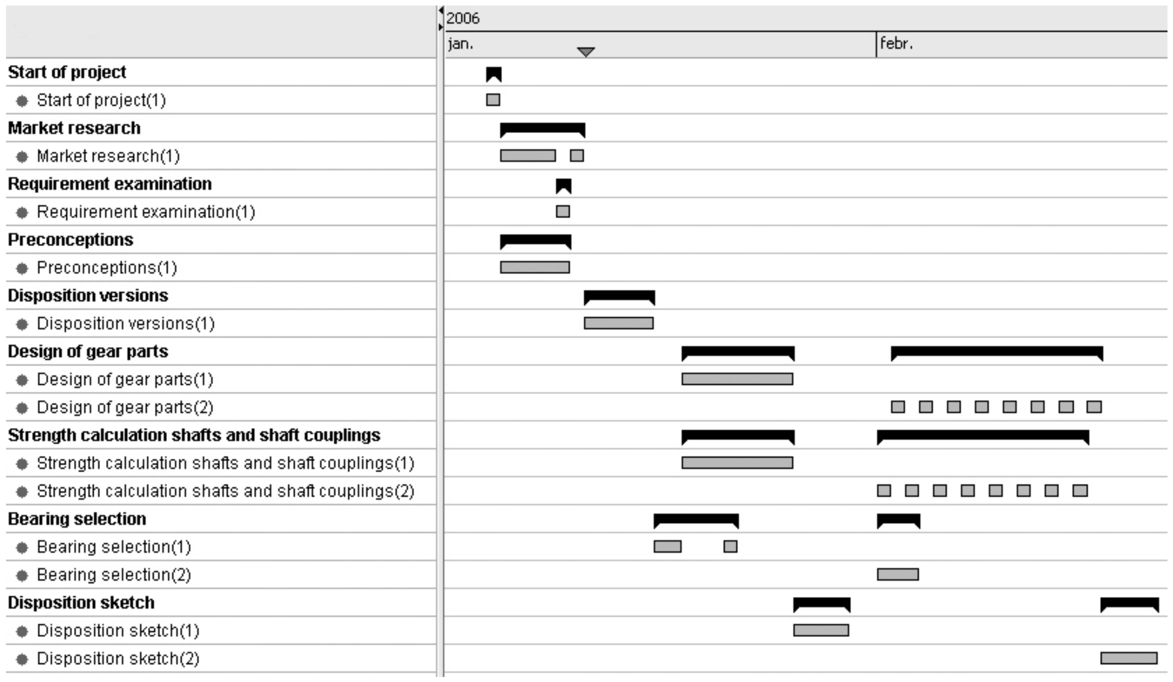

Fig. 7: The final project plan for the gear drive development process.

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 (*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 politics* 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 this 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 to any other (*Design of gear parts (2)* and *Strength calculation shaft and shaft couplings (2)*).

7 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 politics that simulate more realistic scenarios. Another fruitful endeavour is reducing the DSM complexity based on cluster analysis.

8 Preferences

[1] T. Rick, M. Horváth, T. Bercsey, „Product Development Processes – An Optimization of DSM Using Learning Rates in Design Loops", Proceeding of International Conference on Engineering Design ICED 05, pp.516-517

[2] T. Rick, I. Groma, T. Bercsey „Ressourcengerechte Produktentwicklungs-Prozessmodellierung und Optimierung mit Genetischen Algorithmen", 16. Symposium „Design for X" Neukirchen, 2005., 59-66 old., ISBN: 3-9808539-3-4.

[3] M. Milatovic,. A. B. Badiru, "Applied mathematics modeling of intelligent mapping and scheduling of interdependent and multi-functional project resources", Applied Mathematics And Computation Elsevier, 2004, Vol.149 (3), pp.703-721.

[4] O. Reichert, Netzplantechnik, 1994, ISBN: 3-528-05426-3.

[5] A. Basso, L. A. Pecatti, „Optimal resource allocation with minimum activation levels and fixed costs, European Journal of Operational Research, Vol. 131, No. 3, 2001, pp 536-549.

[6] B. Yang , J. Geunes, W. J. O'Brien, „A heuristic approach for minimizing weighted tardiness and overtime costs in single resource scheduling", Computers & Operations Research, Vol. 31, No. 8, 2004, pp 1273-1301

[7] S. L. Nonas, K. A. Olsen, "Optimal and heuristic solutions for a scheduling problem arising in a foundry", Computers & Operations Research, Vol. 32, No. 9, 2005, pp 2351-2382

[8] G. R. Donald, Managing the Design Factory, Carl Hanser Verlag, München, Wien, 1998.

[9] K. Ehrlenspiel, Integrierte Produktentwicklung: Methoden für Prozessorganisation, Produktentwicklung und Konstruktion, Carls Hanser Verlag, München, Wien, 1995.

[10] J. L. Rogers, "DeMAID/GA - An Enhanced Design Manager's Aid for Intelligent Decomposition", Proc. 6th AIAA/USAF/ NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Seattle, WA , September 1996, No. 96-4157.

[11] S. Hartmann, R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem" European Journal of Operational Research, Vol. 127, No. 2, 2000, pp 394-407.

[12] E. G. Gol'stejn, S. Dempe, "A minimax resource allocation problem with variable resources" European Journal of Operational Research, Vol. 136, No. 1, 2002, pp 46-56.

[13] J. L. Rummel, Z. Walter, R. Dewan, A. Seidmann, “Activity consolidation to improve responsiveness”, European Journal of Operational Research, 2005, Vol. 161, pp. 683–703

[14] C. Schwindt, Resource Allocation in Project Management, Springer, Berlin, Heidelberg, New York, 2005, ISBN 3-540-25410-2

Prof. Dr.-Ing. Tibor Bercsey
Dipl.-Ing. Tamás Rick
Dipl.-Math. István Groma
Dipl.-Inf. Ádám Gránicz
Lehrstuhl für Maschinenkonstruktionslehre
Technische und Wirtschaftwissenschaftliche Universität Budapest
Bertalan L. Str. 3, H-1111Budapest
Tel: +36-1-463-1372
Fax: +36-1-463-3505
Email: bercsey.tibor@gszi.bme.hu
rick.tamas@gszi.bme.hu
groma.istvan@gszi.bme.hu
adam_granicz@epam.com
URL: http://www.gszi.bme.hu