INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKOLM, AUGUST 19-21, 2003

A FRACTAL ORGANISATION APPROACH FOR CONCURRENT DESIGN

Egon OSTROSI, Jörg TAUTRIM, Michel FERNEY, Olivier GARRO

Abstract

In concurrent engineering attempts are made to achieve the concurrency through process design decomposition. This organization is under the influence of environment dynamics. This paper presents an approach to concurrent design considering the interrelationships in process design in very narrow confines. The approach performs in three phases. The first phase, called *Design Process Modelling*, consists of the process design models definition and implementation. The second phase, called *Design Process Structuring*, consist in searching for autonomous design units. The third phase called *Implementation Models*, consist in identifying the different models for design process implementation. An example illustrates the proposed approach and simulations with the software *ADEA*.

Keywords: concurrent engineering, design teams, design information management, simultaneous engineering

1 Introduction

Concurrent engineering is a systematic approach for considering all products' life cycle management including the integration of planning, design, production and related phases [3]. During recent years a lot of research has been carry out on the concurrent engineering. The research has been directed to the development, improvement and deployment of the methods and tools for concurrent product design and development, as well as to the organizational approaches and tools for concurrent design product organization.

The philosophy behind the methods for concurrent product design is that the various constraints and requirements, related to the product life cycle, must be integrated in the early design stages [5]. Hence, downstream design changes are minimized.

The philosophy behind the organizational approaches is essentially that in design process there will be quite natural partitioning of the design tasks [1] [4] [6]. Here, the activities of design and development of the product have to overlap. All that is required for this purpose are the design task dependencies of a design project. From the analysis of the task dependencies Steward [6] proposed the design structure matrix (DSM) that indicates which design tasks depend on which design tasks. In order to improve the engineering design management the tasks within the DSM are rearranged such that flow information will solely in the forward direction. From the rearranged matrix the following tasks can be distinguished: *Coupled tasks*, *Sequential tasks* and *Parallel tasks*.

Kusiak [1] outlines a method for task – design parameter grouping. The philosophy behind this method is that in any design process there will be a quite natural grouping of tasks and design parameters into task – design parameter groups. Grouping of tasks and design parameters allow one to determine a potential group of tasks that might be performed simultaneously. The reduction of the time product development, the simplification of the scheduling and management of the design project are some advantages of the design process decomposition [1].

However, the concurrent engineering organization based on the design process decomposition principle depend on the initial conditions of relationships between the different tasks. This organization is under the influence of environment dynamics. As a result, the concurrent engineering organization must be in dynamics. The decisive weakness of the global decomposition is its static nature. To overcome this problem, we consider the interrelationships in the concurrent organization can only be controlled in very narrow confines. According to Warnecke [7], it is an important characteristic of fractal structures: the self – organization.

Warnecke defines the fractal as an independently acting corporate entity whose goals and performance can be precisely described. There are the following interdependent characteristics for fractals:

- *Self – similarity*. It is related either to the structural characteristics of the organizational design or the manner to performing a service. Each fractal must be regarded in the same light as the whole company. The self – similarity permits deviation.
- *Self – organization*. It requires freedom to use the methods appropriate to perform a task. The aim of the self-organization is to achieve the global objectives locally.
- *Dynamics*. Fractals should be designed in such way that relationships within the fractal are stronger than those to the outside. The mechanism of dynamic structuring must be based on the interrelationships within and between fractals.

In this paper we propose a fractal-based approach for the concurrent design. The approach is based on the concept of autonomous design unit, which must have the characteristics of the fractal. The autonomous design units are generated by a heuristic implemented in the software ***ADEA*** (**A**utonomous **DE**sign Units Formation **A**ided System). In order to illustrate the approach, an application is chosen from the automotive industry.

2 Fractal Model for Concurrent Design

The objective of concurrent design is to reduce the lead-time of the *design process* by overlapping *activities* as well as reducing the length of the time in each activity. Hence, the design activities have to overlap. The lead-time reduction by transformation the *sequential design activities* into *parallel design activities* is shown in the figure 1.

A design process is a combination of design activities mobilizing multiples know-how, proceeding in time, and being finalized by an objective. In the same way, we can define a design activity as a combination of design tasks. Thus processes and activities have a *fractal structure*.

For a *design activity*, the objective of the concurrent design is to reduce its lead-time by overlapping *tasks* as well as reducing the length of the time in each task. Hence the *design tasks* have to overlap. The lead-time reduction by transformation the *sequential design tasks* into *parallel design tasks* is shown in the figure 1.

Depending on the level of the fractal, a sequential design process can be transformed into concurrent design process, and the *parallel design process* can be transformed into *dynamic design process* (Figure 3)

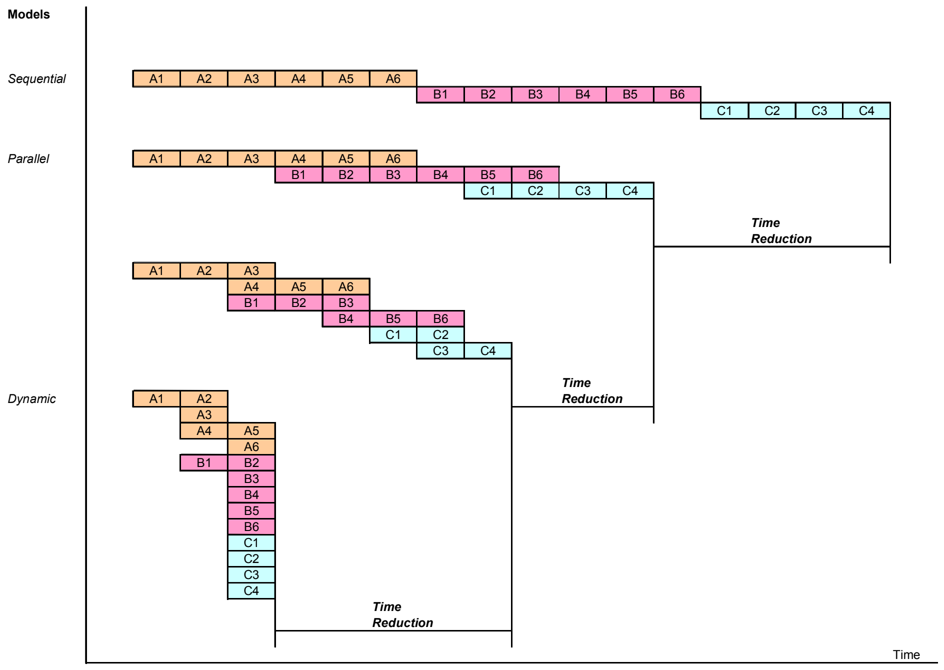


Figure 1. From sequential model toward dynamic model

Autonomous Design Unit formation is based on the hypothesis that a primitive element of the fractal structure is the *primitive design task*. A primitive design task has the following characteristics [2]:

1. ***Relationship:*** *The relationship between the components of the primitive design task, inside the task, is more important than the relationship with the design tasks outside of this task.*
2. ***Goal:*** *The primitive design task performs a service, formulates and follows a goal.*
3. ***Method:*** *The primitive design task needs to be free its appropriate methods.*

The concept of the features is used for the primitive design tasks formulation and expression. Features are viewed as generic or specific forms to which engineers associate certain attributes and engineering knowledge used in different phases of parts design and manufacturing. This characterization of features is quite interesting because it suggests their utilization as vector integration between the product design data and process design data.

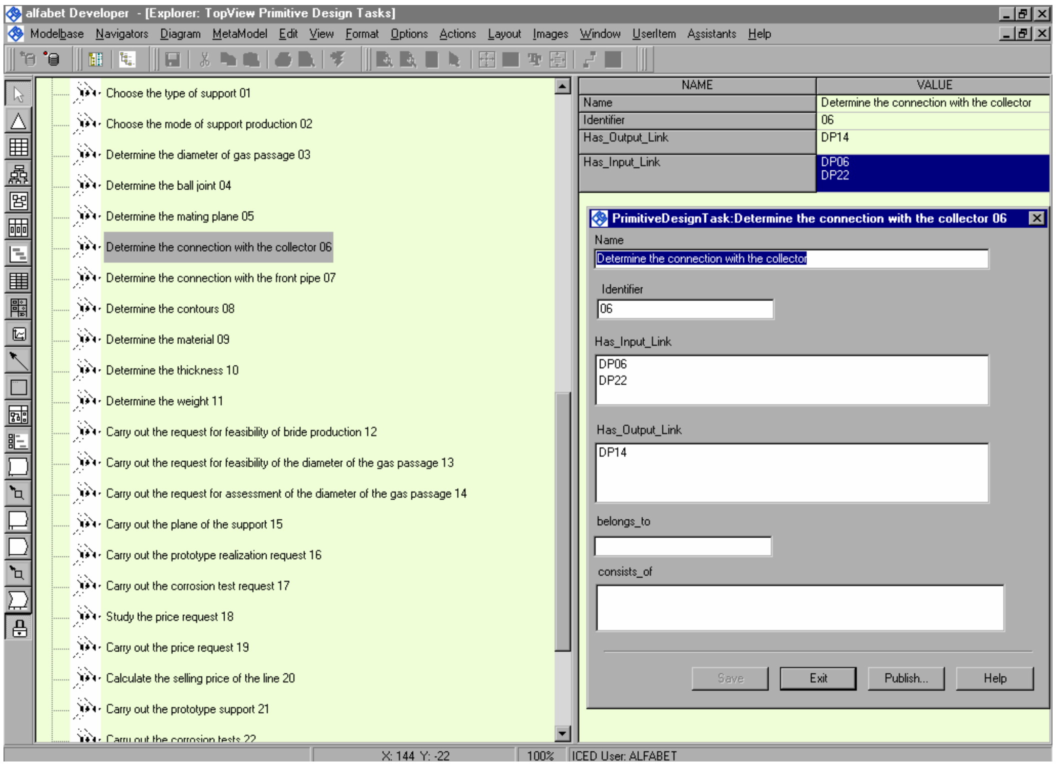
Our approach to fractal organisation includes the following phases:

- The first phase, called *Design Process Modelling*, consists of the process design meta-models definition as well as the building design process models.
- The second phase, called *Design Process Structuring*, consist in searching for autonomous design units.
- The third phase called *Implementation Models*, consist in identifying the different models for design process implementation.

2.1 Design Process Modelling

Process design modelling is developed using the *alphabet technology platform* []. Process design modelling is carry out in two stages. In the first stage, we define the process design meta-model. In the second stage, based on the process design meta-model, we built the process design model.

Stage 1: Design process meta-model definition. In this stage, the primitive design tasks and design parameters classes, their properties and their relationships are defined. Figure 2 shows the primitive design task class definition.


alfabet Developer - [Explorer: TopView Primitive Design Tasks]
Modelbase Navigators Diagram MetaModel Edit View Format Options Actions Layout Images Window UserItem Assistants Help
Save
Exit
Publish...
Help
X: 144 Y: -22
100%
ICED User: ALFABET

Choose the type of support 01
Choose the mode of support production 02
Determine the diameter of gas passage 03
Determine the ball joint 04
Determine the mating plane 05
Determine the connection with the collector 06
Determine the connection with the front pipe 07
Determine the contours 08
Determine the material 09
Determine the thickness 10
Determine the weight 11
Carry out the request for feasibility of bride production 12
Carry out the request for feasibility of the diameter of the gas passage 13
Carry out the request for assessment of the diameter of the gas passage 14
Carry out the plane of the support 15
Carry out the prototype realization request 16
Carry out the corrosion test request 17
Study the price request 18
Carry out the price request 19
Calculate the selling price of the line 20
Carry out the prototype support 21
Carry out the corrosion tests 22

NAME	VALUE
Name	Determine the connection with the collector
Identifier	06
Has_Output_Link	DP14
Has_Input_Link	DP06 DP22

PrimitiveDesignTask:Determine the connection with the collector 06

Name	Determine the connection with the collector
Identifier	06
Has_Input_Link	DP06 DP22
Has_Output_Link	DP14
belongs_to	
consists_of	

Figure 2. Primitive design tasks and their definition

Stage 2: Building design process models. Using the process design meta-model, we can incrementally represent the design process. For a part from automotive industry, using the concept of features, the designers of the part have defined a set of primitive design tasks. These primitive design tasks are drawn incrementally on the graph and their properties have been generated and stored according to the ADEA meta-model (figure 2).

The relationships between the primitive design tasks are represented incrementally by graph of primitive design tasks (figure 3). A node represents a primitive design task. There is a relationship between two design tasks if the output parameter of the first primitive design task represent the input parameter for the second design task. The affinity matrix primitive design task – design parameter (figure 4) represents the data to be subject analysing and structuring by ADEA in the second stage.

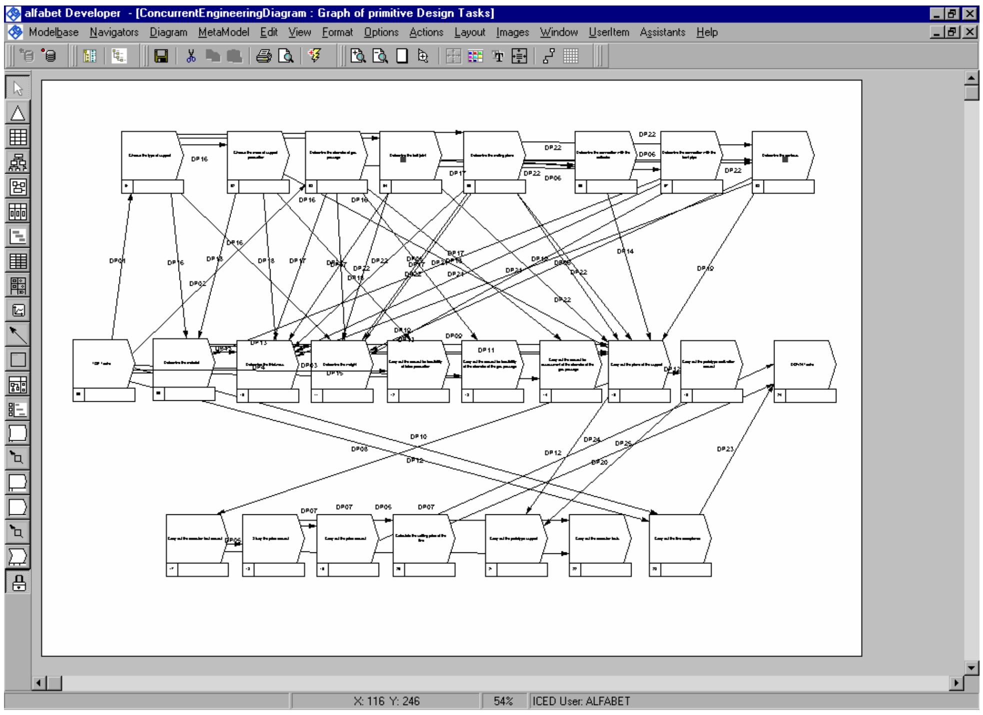

alfabet Developer - [ConcurrentEngineeringDiagram : Graph of primitive Design Tasks]
Modelbase Navigators Diagram MetaModel Edit View Format Options Actions Layout Images Window UserItem Assistants Help
X: 116 Y: 246
54%
ICED User: ALFABET

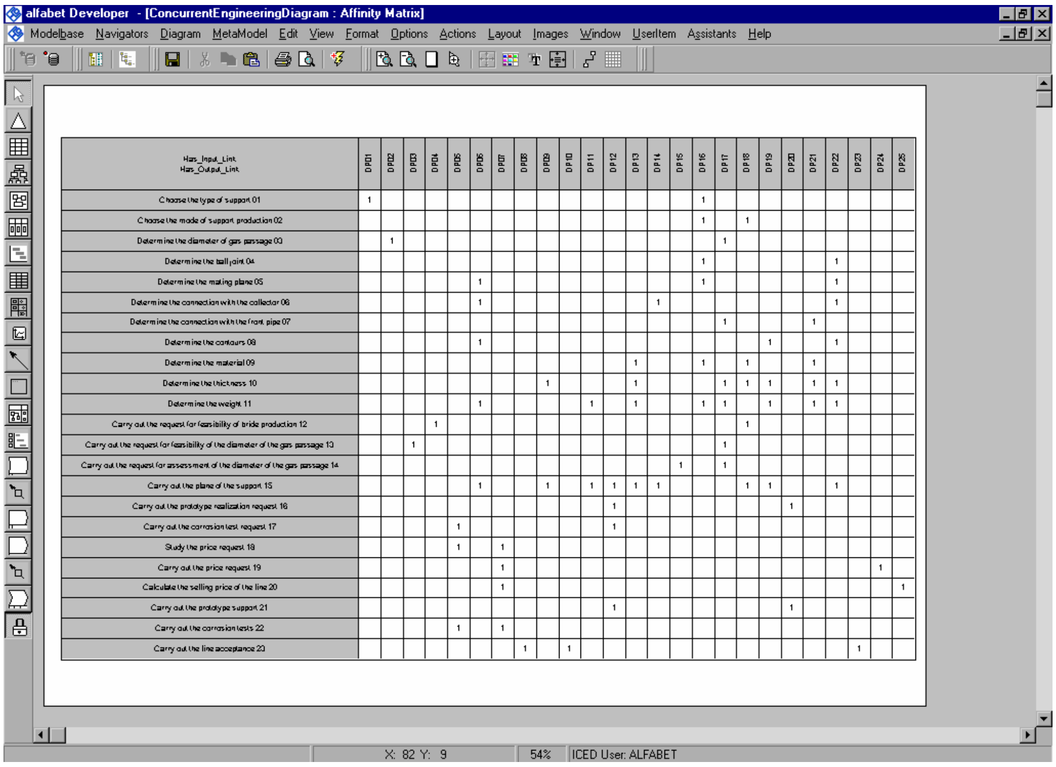
Figure 3. Graph of Primitive Design Tasks

2.2 Design Process Structuring

The objective of concurrent work is the creation of the *autonomous design units*, such as to minimize the exchanges between the units. For the autonomous design units, we propose a heuristic, which is inspired from the fractal structures generation. The heuristic has the following phases:

- Input data definition (Input)
- Grouping Task (Processor)
- Units reduction (Output)
- Feedback

Input data definition (Input). The input data is the matrix of the primitive design tasks. The relationship primitive design task – design parameter is represented in the form of a matrix $[A]^k = a_{ij}$, where each line i, $i=1,2\cdots n$, corresponds to a primitive design task and each row j, $j=1,2\cdots m$ to a design parameter; $k=1,2\cdots$ represent the number of iterations, for example $[A]^0 = a_{ij}$ is the initial matrix (figure 4). The inputs a_{ij}, for $i=1,2\cdots n$, and $j=1,2\cdots m$, can be defined by $\{0,1\}$. If the primitive design task i requires the design parameter j, then $a_{ij}=1$, otherwise $a_{ij}=0$.


alfabet Developer - [ConcurrentEngineeringDiagram : Affinity Matrix]
Modelbase Navigators Diagram MetaModel Edit View Format Options Actions Layout Images Window UserItem Assistants Help
X: 82 Y: 9
54%
ICED User: ALFABET

Has_Input_Link Has_Output_Link	DP01	DP02	DP03	DP04	DP05	DP06	DP07	DP08	DP09	DP10	DP11	DP12	DP13	DP14	DP15	DP16	DP17	DP18	DP19	DP20	DP21	DP22	DP23	DP24	DP25
Choose the type of support 01	1															1									
Choose the mode of support production 02																1		1							
Determine the diameter of gas passage 03		1															1								
Determine the ball joint 04																1						1			
Determine the mating plane 05						1										1						1			
Determine the connection with the collector 06						1								1								1			
Determine the connection with the front pipe 07																	1				1				
Determine the contours 08						1													1			1			
Determine the material 09													1			1		1			1				
Determine the thickness 10									1				1				1	1	1		1	1			
Determine the weight 11						1					1		1			1	1		1		1	1			
Carry out the request for feasibility of bride production 12				1														1							
Carry out the request for feasibility of the diameter of the gas passage 13			1														1								
Carry out the request for assessment of the diameter of the gas passage 14															1		1								
Carry out the plane of the support 15						1			1		1	1	1	1				1	1			1			
Carry out the prototype realization request 16												1								1					
Carry out the corrosion test request 17					1							1													
Study the price request 18					1		1																		
Carry out the price request 19							1																	1	
Calculate the selling price of the line 20							1																		1
Carry out the prototype support 21												1								1					
Carry out the corrosion tests 22					1		1																		
Carry out the line acceptance 23								1		1													1		

Figure 4. Affinity matrix primitive task – design parameter

Task grouping (Processor). Grouping tasks is done in the following steps:

Step 1 (Seed Choosing): Choosing a seed design parameter creates a new unit. The parameter, which is required by the highest number of tasks, can be considered as a seed design parameter. However, the rules for choosing a seed design parameter can be contextual.

Step 2 (Tasks Assignment): All the tasks depending on the seed design parameter are admitted in the unit.

Step 3 (Parameters Admission): The design parameters required by the primitive design tasks in the unit are considered. The design parameters can be admitted or rejected according to the coefficient of attraction:

$$\sigma_j = \frac{\sum_{i \in Unit} a_{ij}}{\sum_{i=1}^{n} a_{ij}} \qquad (1)$$

where:

- $\sum_{i \in Unit} a_{ij}$ is the number of primitive design tasks *in the unit* depending on the design parameter;

- $\sum_{i=1}^{n} a_{ij}$ is the total number of primitive design tasks depending on the design parameter.

The coefficient of attraction can vary from 0 to 1.

Step 4 (Unit Expanding): All the design parameters in the unit are considered as the seed parameters. Repeating the steps 1-3 until no more parameters can be considered leads to the expansion of the unit.

Step 5 (New Units): If in step 1 a seed is found then steps 2-4 are repeated. Otherwise the task grouping is finished.

Unit Reduction (Output). In each unit, the primitive design tasks are reduced into a macro-primitive design task and the design parameters are reduced into a macro-design parameter. A new matrix is formed $[A]^k=a_{ij}$.

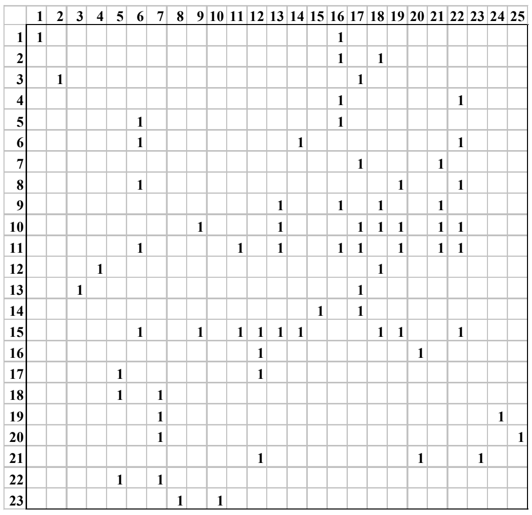
Feedback. If a new matrix $[A]^k=a_{ij}$ is found then this matrix will be considered as the input data. Otherwise the task grouping is finished.

The models that result from the simulations with ADEA can be divided into the following:

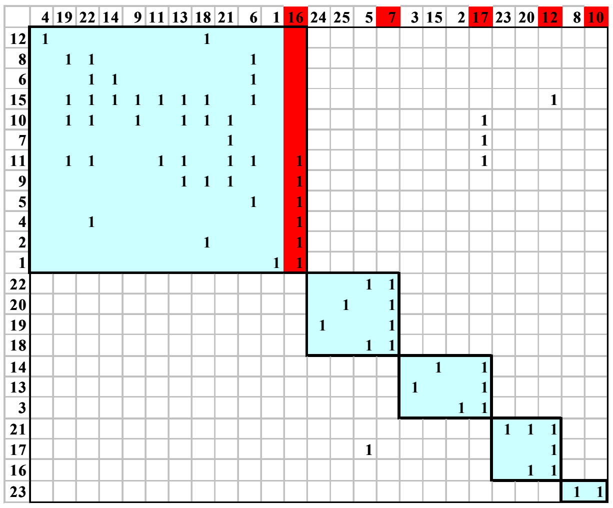
- Seed-Based Unit Model
- Unit – Unit Based Model

Seed-Based Unit Model. ADEA generates this model without considering the feedback in this model an important design parameter is considered as *seed design parameter*. Between the remaining design parameters, the design parameter having the most exchanges with the seed design parameter is chosen. The unit of the two design parameters is formed. Next, a third design parameter is released, from the set of the remaining design parameters. This parameter will be attracted either by the seed design parameter, or the unit, if the exchanges in terms of the tasks are considered important. This procedure can be repeated many times until, for example, a unit of the desired number of parameters is generated. The other units can be formed following the same procedure.

A model similar to the ***Seed Based Unit Model*** is the ***Multi-seed Based Unit Model***. In this model, more than one design parameter can be considered simultaneously as the seeds. When, two design parameters exchange a common primitive design task, where one of them is the seed, then a unit can be formed. If some concurrent units have relationships with another design parameter, then this design parameter will aggregate with the unit, with which it has more exchanges. In this way, larger and larger units are formed.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1															1									
2																1		1							
3		1															1								
4																1						1			
5						1										1									
6						1								1								1			
7																	1				1				
8						1													1			1			
9													1			1		1			1				
10									1				1				1	1	1		1	1			
11						1					1		1			1	1		1		1	1			
12				1														1							
13			1														1								
14															1		1								
15						1			1		1	1	1	1				1	1			1			
16												1								1					
17					1							1													
18					1		1																		
19							1																	1	
20							1																		1
21												1								1			1		
22					1		1																		
23								1		1															



	4	19	22	14	9	11	13	18	21	6	1	16	24	25	5	7	3	15	2	17	23	20	12	8	10
12	1							1																	
8		1	1							1															
6			1	1						1															
15		1	1	1	1	1	1	1		1													1		
10		1	1		1		1	1	1											1					
7									1											1					
11		1	1			1	1		1	1		1								1					
9							1	1	1			1													
5										1		1													
4			1									1													
2								1				1													
1											1	1													
22															1	1									
20														1		1									
19													1			1									
18															1	1									
14																		1		1					
13																	1			1					
3																			1	1					
21																					1	1	1		
17															1								1		
16																						1	1		
23																								1	1

Figure 5. Design Process Matrix after Grouping

The characteristic of this Seed-Based Unit Model and Multi-seed Based Unit Model is the concurrency between the units. For Seed-Based Unit Model illustration, let us consider the process design represented by the matrix of primitive design tasks (figure 4). Applying the **seed-based unit heuristic** (without the feedback) with a coefficient of admission parameter $\sigma_j \geq 0.4$, results in the matrix represented in figure 5.

Five autonomous design units are found. The seeds are the following design parameters: 16 for the first unit, 7 for the second unit, 17 for the third unit, 12 for the fourth unit, and 10 for the fifth unit. Each unit performs a group of primitive design tasks, which deal with a group of design parameters. For example, the first unit performs the group of primitive design tasks {12, 8, 6, 15, 10, 7, 11, 9, 5, 4, 2, 1} dealing with the group of design parameters {4, 19, 22, 14, 9, 11, 13, 18, 21, 6, 1, 16}.

The design parameters 5, 17, 12 must be shared between the units. For minimizing the dependency between units, the shared design parameters must be predefined preliminary. In this case predefinition is done based on the expert know-how. The matrix after the predefinition of shared variables is shown in figure 6

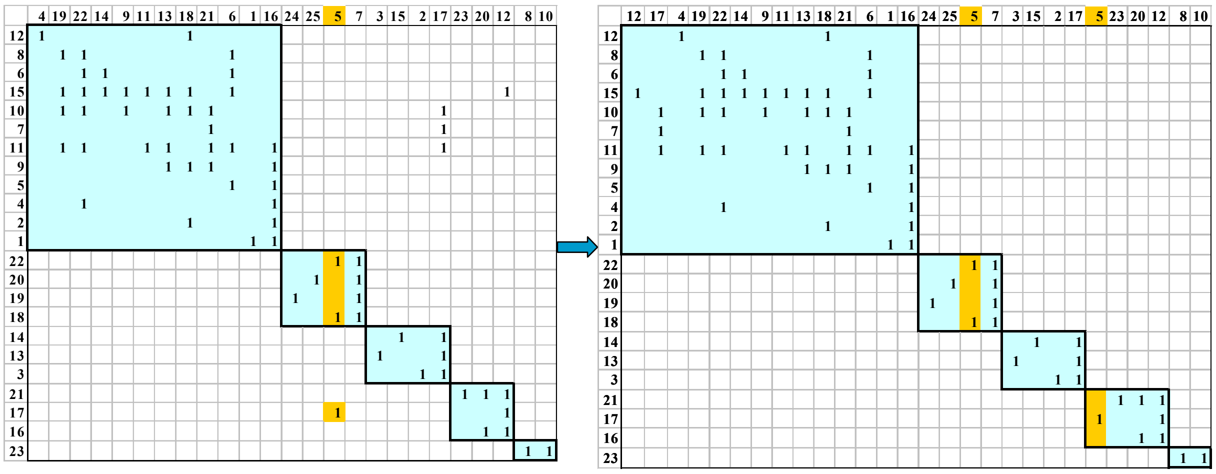
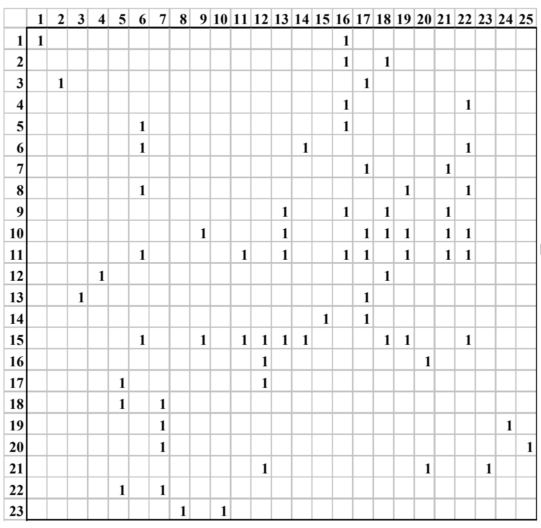


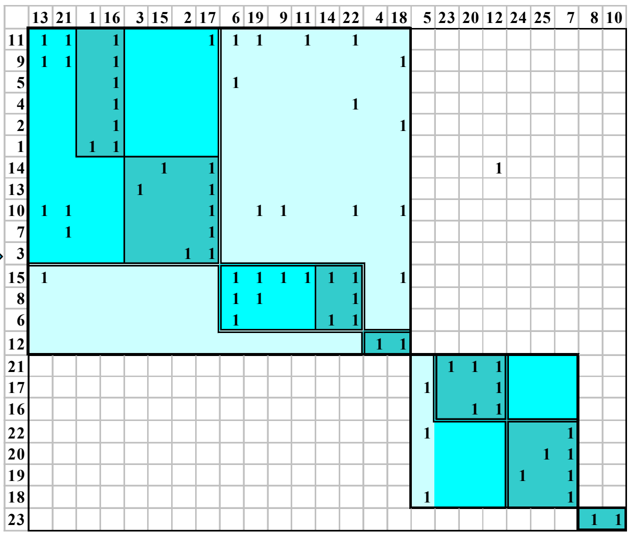
Figure 6. Design parameters sharing

Unit-Unit Aggregation Model. This model is generated by ADEA considering the proposed heuristic with the feedback step. In this model the formed units are reduced to a macro-parameter and macro-task. So a new seed design parameter can be chosen. The procedure will be the same as in the seed-based unit model. In this way, the new-formed units are composed of units, and so on. All these units have a fractal structure. In this model, the concurrent engineering can be done in level of units. The primitive design tasks in the unit can be integrated. For Unit-Unit Aggregation Model illustration, let us consider the same design process as in the Seed-Based Unit Model. Applying the **Unit-Unit Aggregation heuristic** (with the feedback) after the third iteration, the grouping results in the matrix in figure 7.

The results show that the design process can be done in three autonomous design units, which can be seen in the diagonal blocks. These units have a fractal structure. They are composed of autonomous design units, which are fractal units too. For example, if we zoom on the first autonomous design unit, we can see that it is composed of three units. The autonomous design units are self-similar. So, the simultaneous work can be done between the autonomous design units in each level of grouping. For example, assuming the design parameter 12 can do the simultaneous work between the first autonomous design unit and the second design unit. Similarly, assuming the design parameter 5 can do the simultaneous work between the two autonomous design units inside the second autonomous design unit (figure 7).



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1															1									
2																1		1							
3		1															1								
4																1						1			
5						1										1									
6						1								1								1			
7																	1				1				
8						1													1			1			
9													1			1		1			1				
10									1				1				1	1	1		1	1			
11						1					1		1			1	1		1		1	1			
12				1														1							
13			1														1								
14															1		1								
15						1			1		1	1	1	1				1	1			1			
16												1								1					
17					1							1													
18					1		1																		
19							1																	1	
20							1																		1
21												1								1			1		
22					1		1																		
23								1		1															



	13	21	1	16	3	15	2	17	6	19	9	11	14	22	4	18	5	23	20	12	24	25	7	8	10
11	1	1		1				1	1	1		1		1											
9	1	1		1												1									
5				1					1																
4				1										1											
2				1												1									
1			1	1																					
14						1		1												1					
13					1			1																	
10	1	1						1		1	1			1		1									
7		1						1																	
3							1	1																	
15	1								1	1	1	1	1	1		1									
8									1	1				1											
6									1				1	1											
12															1	1									
21																		1	1	1					
17																	1			1					
16																			1	1					
22																	1						1		
20																						1	1		
19																					1		1		
18																	1						1		
23																								1	1

Figure 7. Fractal grouping

Comparing the Seed-based Unit Model to the Unit-Unit Aggregation Model, we notice that the latter structures better the design process. Actually, we can have the global views as well as the local views. These permit to better control the design process, the relationships between the primitive design tasks and the design parameters.

2.3 Implementation Models

To implement the Seed-based Unit Model and Unit-Unit Aggregation Model, the following implementation models are distinguished:

- Central Unit Model;
- Parallel Model;
- Dynamic Model.

Central Unit based Model. Between different units, a particular one can be considered as central. Then the design methodology impose that there is a higher level of design analysis at the *central unit*. The requirements of the other units are integrated in the tasks within the central unit. The CAD models and features based CAD models are used here as vector of information transmission. They represent the *central information* used by the other units. The design process still take place after the design tasks in the central unit has been completed. Design change requires the modification of the central information driven by the central unit. Seed-based Unit Model can be implemented based on these assumptions.

Parallel Model. The process for parallel definition is characterised by overlapping design parameters. Because of the identification of the overlapping design parameter, which represents the constraint design parameter, the units can be released to facilitate greater concurrency. Here the information exchange exhibits greater intensity at the overlapping design parameter. However, the nature of the change becomes more ambiguous within the parallel definition, with higher exchange between design units. Seed-based Unit Model and Unit-Unit Aggregation Model can be implemented based on these assumptions

Dynamic Model. A further development of the concurrent definition system demands to introduce a much more intensive level of communication during the design process. This allowed a more time influence of the design tasks. The process becomes much more concurrent as all the design tasks can start rather at the same time. Information exchange is far more intensive. Unit-Unit Aggregation Model can be implemented based on these assumptions.

3 Conclusion

In this research, an alternative approach to concurrent design based on the concept of the fractal has been presented. It is assumed that a process can be represented by a fractal structure. The primitive design task is considered as the primitive element of the fractal structure. From simulations with the software ***ADEA*** (**A**utonomous **DE**sign Units Formation **A**ided System), the model, called Unit-Unit Aggregation Model, represents characteristics of fractal organization: *Self – similarity, Self – similarity and Dynamics.*
From this model, the following observations can be drawn:

- The quality of the units depends on the quality of the primitive task definition
- The concurrency could be seen as well locally as globally.
- The global decomposition can be seen as an alternative of the fractal structuring

Three models are distinguished for implementation. They are Central Unit Model, Parallel Model and Dynamic Model. They are discriminated on the basis of the intensity information exchange at the overlapping design parameter.

References

[1] Kusiak A., and Wang J.R, "Decomposition of the Design Process", Journal of Mechanical Design, Vol. 115, 1993, pp. 687-696.

[2] Mutel B. and Ostrosi E., "An approach for structuring of production systems into autonomous working unities", 2nd International Congress Industrial Engineering in a World without borders, Albi, France, 1997.

[3] Prasad B., Wang F., and Deng J., "A Concurrent Workflow Management Process for Integrated Product Development", J. Eng. Design, Vol. 9, No. 2, 1998, pp. 121-135.

[4] Decreuze C., Ferney M., Feschotte D., Jacobe P., Zerhouni S.N., "Method for design tasks partition", JESA, Vol. 30, No. 10, 1996, pp. 1315-1350.

[5] Brissaud D. and Garro O., "An Approach to Concurrent Engineering Using Distributed Design Methodology", CERA Journal, Vol. 4, No. 3, 1996, pp. 303-311.

[6] Steward D., "Planning and Managing of the design systems", Proceedings of Portland International Conference on Management of Engineering and Technology, Portland, OR, USA, October 1991, pp. 27-31.

[7] Warnecke H.J., "The Fractal company", Springer-Verlag, Berlin,1993.

[8] Alfabet Technology Platform, version 4.0, Berlin, Germany 2001

Corresponding author:
Egon OSTROSI
Université de Technologie de Belfort - Montbéliard
Laboratoire de Recherche Mecatronique3M
90010 Belfort Cedex - France
Tel: Int +33 (0)3 84 58 31 36
Fax: Int +33 (0)3 84 58 31 46
E-mail: Egon.Ostrosi@utbm.fr
URL: http//: www.utbm.fr