Conflict Management in Design Process: Focus on Changes Impact
Khadidja Grebici, Mohamed Zied Ouertani, Eric Blanco, Lilia Gzara Yesilbas, Dominique Rieu

To cite this version:

HAL Id: hal-00122469
https://hal.science/hal-00122469
Submitted on 2 Jan 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Conflict Management in Design Process: Focus on Changes Impact

K. GREBICI a,1, M-Z. OUERTANI b, E. BLANCO a,
L. GZARA-YESILBAS b, D. RIEU c

a GILCO-INPGrenoble, 46 Avenue Félix Viallet, 38031, Grenoble Cedex, France
b CRAN – UHP Nancy I, BP 239, Vandoeuvre-lès-Nancy, 54506, France
c LSR-SIGMA- IMAG, 2 Place Doyen Gosse-38031-Grenoble Cedex, France

Abstract: Design has increasingly become a collaborative endeavour carried out by multiple actors with diverse kinds of expertise. A critical element of collaborative design is to manage conflict, emerging due to the multi-actors interaction, and particularly the impacts once the conflict is resolved. Indeed, the conflict resolution comes up with a solution which often implies modifications on the product and the process organisation. This paper deals with this problematic of changes impact on conflict management process. It quantifies key issues with regards to concurrent engineering that enables us to better manage the design process. Strategies to overlap coupled activities are proposed based on the dependencies between the handled data during the design process.

Keywords: Conflict management, changes impact, data dependencies, overlapping strategies

Introduction

Due to multi-actors interaction in collaborative engineering design, conflicts can emerge from disagreements between designers about proposed designs. Therefore, a critical element of collaborative design would be conflict resolution. In a collaborative design context, conflicts occur when at least two incompatible design commitments are made, or when a design party has a negative critique of another design party’s actions [1]. The conflict management process could be perceived as the succession of five phases: Conflict detection, Conflict resolution team identification and formation, Negotiation management, Solution generation and Solution impact assessment.

Current conflict management approaches in collaborative design focus on Conflict detection [1] [2] [3], Negotiation management and Solution generation phases [4] [5] [6] [7].

In a previous work [8], we proposed a methodology, called DEPNET2, to tackle the Conflict resolution team identification and formation phase. This methodology addresses the problematic of identifying the actors to be involved in the negotiation process to resolve the conflict. Based on a process traceability system, the DEPNET methodology consists of identifying and qualifying the dependencies between the data

---

1 Corresponding Author: Khadidja Grebici, E-mail: grebici@gilco.inpg.fr
2 product Data dEPendencies NETwork identification and qualification

To appear in the proceedings of the 13th ISPE International Conference on Concurrent Engineering: Research and Applications, 2006
handled during the design process execution. This leads to the construction of a data dependencies network which allows identifying the actors to be involved in the conflict resolution process. Indeed, each data is carried out by an activity. This activity has a responsible to execute it. Consequently, once a data is identified, the actor responsible for its realisation is identified.

Concerning the Solution impact assessment phase once the conflict is resolved, it has not been tackled on the reviewed works. Indeed, the selected solution often implies the modification of one or more input data of the activity where the conflict has emerged, and thus, generating a cascade of modifications on the already produced data. Consequently, these data have to be redefined. This implies re-executing the various design activities responsible on the elaboration of these product data and also adjusting the process still in the pipeline. Accordingly, strategies are to be proposed to coordinate re-execution of the concerned activities. Hence, this paper purposes’ is to come up with strategies to coordinate the activities re-execution. In order to do so, we based ourselves on the data dependencies network, already built thanks to the DEPNET methodology, to identify the impacted data to be redefined, as well as the correspondent activities to be re-executed.

The remaining part of the paper is organised as follow. First, section 1 presents the data dependencies network constructs. Then, section 2 describes the mechanisms to identify the impacted data and their correspondent activities once a solution is selected. Section 3 presents a set of activities re-execution strategies. Finally, section 4 discusses and concludes the paper.

1. Data Dependencies Network Constructs

The data dependencies network is an oriented graph composed of nodes and arcs (cf. Example Figure 1):

- Nodes correspond to the product data handled during the design process and leading to the elaboration of the data source of conflict, i.e. the set of data on which this source of conflict depends. These product data can be of several types such as: structural, functional, behavioural, geometrical, etc. They correspond to the various descriptions of the product, elaborated by designers during the development process, in terms of geometrical entities, functions, bills of materials, CAD models, calculation notes, etc.

- Arcs correspond to the dependency relationships between the various nodes identified in the network (data). In a context of collaborative design, dependency between two data could be on forward or feedback direction. Forward dependent data are those that require input from other activities, but not themselves. Feedback dependent data are those that need inputs from other activities including themselves. The feedback links are to be considered since they are a source of rework and thus are resources consuming and time consuming. Thus, two data are said to be dependent in the case of forward dependency or interdependent in the case of feedback dependency.

The dependency relationships in this network are quantified with a dependency degree which is an aggregation of the three attributes: completeness (adapted from [9], variability (adapted from [10]) and sensitivity (adapted from [10]) (cf. Eq. (1)).

\[
\text{Dependency Degree} = \text{Completeness} \times \left(1 + \left(\text{Variability} \times \text{Sensitivity}\right)\right) \quad \text{Equation 1}
\]
These attributes are valued by actors when they capture their activities in the design process traceability system. In practice, these attributes are not always easy to define quantitatively. Therefore, using structured expert interviews, qualitative inputs are developed to provide insights on how to evaluate them. The *Completeness* (C) expresses the suitability of an input data to the creation of an output data (0 Weak, 1 Not Vital, 2 Vital and 3 Extremely Vital). The *Variability* (V) describes the likelihood that an output data provided by an activity would change after being initially released (0 Not Variable, 1 Low Variability, 2 Moderate Variability and 3 High Variability). The *Sensitivity* (S) describes how sensitive the completion of an output data is to changes or modifications of an input data. It expresses the degree to which work is changed as the result of absorbing transferred data (0 Not Sensitive, 1 Low Sensitivity, 2 Moderate Sensitivity and 3 High Sensitivity).

As completeness, variability and sensitivity are valued with numerical values (0, 1, 2 and 3), the resultant range value of the dependency degree is an integer between 0 and 30, whereas {0, 1, 2, 3, 4} denotes a weak dependency and a low risk of rework, {5, 6, 7, 8, 9, 10, 12, 14} describes a moderate dependency and a moderate risk of rework and {15, 20, 21, 30} denotes a high dependency and a high risk of rework.

Based on the data dependency network and thanks to a set of SQL queries on the database storing the process execution instances, it is possible to identify for each data of the network, the activity responsible on its elaboration as well as the actor performing this activity. In the next section, we detail how to evaluate the impact of a selected solution once the conflict is resolved.

### 2. Solution Impact Assessment

The technical solution selected through the negotiation and resolution phase corresponds to the change of one or more product data involved in the design process leading to the elaboration of the data source of conflict. The data to change is then a part of the data dependency network presented in Section 1. Hence, assessing the selected solution impact returns with propagating the impact of those data changes through the data dependency network and thus on the organisation of the responsible activities. For example, in the case of a conflict occurring on the piece of data D6 (cf. Figure 1), supposes that the solution retained after resolving that conflict consists of changing the piece of data D9 value. Consequently, according to the data dependency network, all data linked to D9 with a forward dependency relationship (arrow starting from D9) must be changed, i.e. data D1, D2, D3 and D4. Then, all data depending on D1, D2, D3 and D4 must be changed, i.e. D5 (depending on D4), D6 and D10 (depending on D2), etc. Therefore, activities responsible of the elaboration of D1, D2, D3, D4, D5, D6, D10, etc. have to be re-executed. In this example, the total number of solution (D9) depending nodes is low (10 nodes) and re-execution of the design process is simple. In case of complex products, whereas the corresponding data dependency network is huge, identifying the solution depending nodes and then the design activities to be re-executed may be ad-hoc and costly task. Not all the activities responsible of the identified depending data will be re-executed; such an operation is costly and time consuming. Thus, a concept of *critical Data Dependency Network* is introduced in order to reduce the number of solution depending nodes to consider for design process re-execution. In other words, this consists of eliminating data having low dependency...
degree and their entire successors among the impacted data\(^1\). Hence, based on the critical data dependency network, the identification of the activities to be re-executed is performed. Indeed, a set of SQL queries applied to the process execution database allow identifying the activities responsible on the elaboration of the critical data network nodes, the actors performing these activities as well as the input data and output data of each activity. Then, the actor responsible of the change management has at his disposal the set of activities to re-execute with the associated actors and Inputs/Outputs. Based on the input and output data, the order of executing activities can be determined. Indeed, when the output of an activity A corresponds to the input of another activity B, that means that activity A precedes activity B. In this paper, the focus is put on the resulting activities re-execution organisation. The aim of organising activities re-execution is to minimise the re-execution time by decreasing the probability and the magnitude of iterations in the newly executed process. This can be done by enhancing activities’ overlapping and concurrent execution. In Section 3, a set of coordination strategies are proposed based on dependency degree values.

**Figure 1. Example of data dependencies network**

3. Design Process Re-execution Strategies

Depending on the dependency condition of the data to be changed (variability, sensitivity and completeness), i.e. depending on the probability of iteration on both feed forward and feedback dependencies between associated activities (those producing and those using the data to be changed), different strategies for re-executing these activities are examined.

Inspired from the coordination strategies developed by [10] and [11], a set of coordination strategies, i.e. ways of ordering activities and diffusing data, are proposed. First, we discuss the case of dependency, i.e. data linked with only feed forward dependency. Then, the case of interdependency is treated, i.e. data linked with both feed forward and feedback dependencies.

---

\(^1\)the methodology proposed to identify the critical data dependency network is not presented in this paper; it will be developed in a future publication.
3.1 Dependency Case: Only Feed Forward Dependency

This case concerns two activities A and B, respectively producing data D1 and D2 where D1 and D2 are linked by feed forward dependency (cf. Figure 2).

In this section, different values of completeness (Weak, Not Vital, Vital and Extremely Vital) are considered and for each case, depending on variability and sensitivity values, strategies of coordination are suggested.

![Activity A](D1) ![Activity B](D2) Feed forward dependency

**Figure 2. Dependency case**

**Case 1: if the upstream data (D1) is weak (Completeness of D2 = 0), for every value of the variability V of the upstream data (D1), the execution of the downstream activity (B) does not require the considered data (D1). Thus, the adopted strategy would be the execution of the involved activities in early overlapping way. This means both activities A and B execution can begin at the same time without any risk of iteration.**

**Case 2: if upstream data (D1) is at least not vital (Completeness of D1 ≠ 0)**

**Case 2.1: if the upstream data (D1) is not variable (Variability of D1 = 0),** for every value of sensitivity S of downstream data (D2), the required upstream data (D1) is not variable (V=0). This means when the upstream data D1 is ready to be diffused, the downstream activity B starts earlier with finalised information D1. Thus, the corresponding re-execution strategy is a “pre-emptive overlapping” (see case 2.3 below).

**Case 2.2: if the upstream data (D2) is not sensitive (sensitivity of D2 = 0),** for every value of the variability V of the upstream data (D1), there is no impact in the downstream. Thus, the strategy of re-execution is based only on upstream completeness values and it represents a “thrown over the wall overlapping”, where activity A diffuses preliminary data (D1) corresponding to the required (by activity B) completeness value (C=1, C=2 or C=3).

**Case 2.3: if the upstream data (D1) is at least low variable, the downstream data (D2) is at least low sensitive (Variability of D1 ≠ 0 and Sensitivity D2 ≠ 0) and the upstream data D2 is Not Vital (C=1),** the uncertainty conditions depend only on the upstream data (D1) variability and the downstream data (D2) sensitivity. Table 1 summarizes the different coordination strategies depending on D1 variability and D2 sensitivity values (a preliminary data exchange is represented by a discontinued arrow, and a finalized data is represented by a full arrow). The various proposed strategies are defined in the following.

- **Distributive overlapping:** makes possible to both start downstream activity (B) with preliminary information and to pre-empt later changes in the exchanged upstream data (D1). The impact of overlapping is distributed between upstream and downstream activities, thus this strategy is called distributive overlapping.

- **Iterative overlapping:** the activities are overlapped by beginning the downstream activity (B) with preliminary information, and incorporating design changes in subsequent downstream iterations. The overlapping is said to be iterative.
- **Preemptive overlapping**: When variability of upstream data (D1) is low (V=1), information can be frozen earlier than its normal freeze time. This will allow high sensitive (S=3) downstream data (D2) to start earlier with finalised information. In such a case, the exchanged information is to be pre-empted by taking its final value at an earlier point in time. This is called pre-emptive overlapping and would help to reduce design time by starting the downstream activity earlier but with frozen upstream activity information.

- **Divisive overlapping**: the information (D1) is disaggregated into components to see if any of the components have low variability or if transferring any of the components in their preliminary form to the downstream activity is practical.

### Table 1. Dependent activities coordination strategies

<table>
<thead>
<tr>
<th>Low_V (1)</th>
<th>Medium_V (2)</th>
<th>High_V (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low_S (1)</td>
<td>Distributive Overlapping</td>
<td>Iterative Overlapping</td>
</tr>
<tr>
<td>Medium_S (2)</td>
<td>Distributive Overlapping</td>
<td>Activity and dependency link redefinition</td>
</tr>
<tr>
<td>High_S (3)</td>
<td>Preemptive Overlapping</td>
<td>Multifunctional Team (Concurrent Execution)</td>
</tr>
</tbody>
</table>

- **Activity and dependency link redefinition**: this strategy calls for disaggregating predecessor design information (D1) into two or more information fragments that can be released (as incomplete or partial design information) earlier than the planned one-shot release. Consequently, the follower (i.e. downstream) activity (B) is broken into n sub-activities, where each sub-activity utilizes one upstream information (D1) fragment. There is no general rule for the optimal number of upstream partial information pieces required or the number of downstream activity breakdowns. This depends on the specific design situation being analysed and requires a thorough understanding of the design process details.

- **Multifunctional team (concurrent execution)**: The basic goal of such a strategy is to guarantee that downstream concerns are considered upstream. This will result in: decreasing variability of a predecessor due to the fact that upstream activity engineer(s) are working closely with downstream activity engineer(s); and lower sensitivity value of the successor (D2) due to the instantaneous feedback accomplished by the multifunctional team.

**Case 2.4**: if the upstream data (D1) is at least low variable, the downstream data (D2) is at least low sensitive (Variability of D1 ≠ 0 and Sensitivity D2 ≠ 0) and the upstream data D2 is at least Vital (C ≤ 2). If upstream completeness is vital
(C=2) or extremely vital (C=3) then the interest is to avoid as much as possible upstream rework because this induces long iterations. Thus given low sensitivity (first row of Table 1), and medium or high variability, it is more interesting to realize coordination with less preliminary diffusion as possible; this means to prioritize distributive then iterative overlapping in order to reduce upstream iteration. However, if variability is low (first column in Table 1) and downstream sensitivity is medium or high, pre-emptive is more interesting then distributive overlapping in order to reduce iterations in both upstream and downstream activities.

3.2 Interdependency Case: Feed Forward and Feedback Dependencies

In most product development processes, interdependency between activities is essentially a one-way dependency between an upstream activity and a downstream activity with a feedback dependency from the downstream activity to the upstream activity (Figure 4). When the development process involves interdependent activities, it is divided into two stages: a planned stage and a random iteration stage. The first stage contains only the initial attempts of both activities. The second stage contains all the subsequent design iterations of both activities. There is no confusion on what activity to start first in this case.

The first stage consists of coordination strategy according to forward dependency degree between upstream and downstream (case § 3.1). The second stage represents coordination strategy according to feedback dependency degree between downstream and upstream.

As an illustration, we consider the interdependent activities in Figure 3 (Activity 9 and Activity 3 corresponding to data D9 and data D3 respectively from the Figure 1).

![Figure 3. Interdependency case](image)

In this example, forward dependency degrees are Medium (Low Completeness, High Sensitivity and Low Variability) and feedback dependency degree is medium (LC, LS and HV). Thus, the first step (coordination strategy between activity 3 and activity 9) corresponds to pre-emptive overlapping. Once activity 3 is ready to diffuse D3, the second step is launched and iterative overlapping is applied (Figure 4).

![Figure 4. Coordination strategies of interdependent activities](image)
4. Conclusion

In this paper, an approach to address the problematic of solution impact assessment is presented. This approach is based on a data dependencies network in order to identify the impacted data as well as the correspondent activities. Then, a set of strategies to coordinate the activities to be re-executed is proposed based on the three attributes Completeness, Variability and Sensitivity. These strategies are inspired by some research works addressing the overlapping problem by developing approaches to study Concurrent Engineering (CE) process [10] [11] [12] [13]. These approaches help when and how two development activities should be overlapped. The feasibility and acceptability of these coordination strategies have been validated on an industrial case. However, these approaches propose coordination strategies regarding only two attributes; variation and sensitivity and to coordinate only two activities. In this paper, we proposed to extend these approaches to the entire design process. Furthermore, the present approach relies on the definition of the data dependency network which is not the case with previous approaches.

Currently, the present approach has not been yet validated. A tool to support this methodology is under development to enable experimentation in an industrial context.

References