A Process Traceability Methodology to Support Conflict Management
Mohamed Zied Ouertani, Lilia Gzara Yesilbas, Gabriel Ris

To cite this version:
Mohamed Zied Ouertani, Lilia Gzara Yesilbas, Gabriel Ris. A Process Traceability Methodology to Support Conflict Management. IEEE. 10th International Conference on CSCW in Design, Southeast University, Nanjing, China, May 3-5, 2006., May 2006, Nanjing, China. pp.CDROM, 2006. <hal-00021935>

HAL Id: hal-00021935
https://hal.archives-ouvertes.fr/hal-00021935
Submitted on 29 Mar 2006

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.
A Process Traceability Methodology to Support Conflict Management

Mohamed-Zied Ouertani, Lilia Gzara-Yesilbas, Gabriel Ris

CRAN - Automatic Research Centre of Nancy, CNRS UMR 7039
Henri Poincaré University, Nancy I, Faculty of Sciences and Technologies, BP 239
54506 Vandoeuvre-lès-Nancy - FRANCE
Mohamed-Zied.Ouertani@cran.uhp-nancy.fr; Lilia.Gzara@cran.uhp-nancy.fr; Gabriel.Ris@aipl.uhp-nancy.fr

Abstract

The process of collaborative engineering design is relatively complex, and often results in various conflicts due to multi-actors interactions. Therefore, a critical element of collaborative design would be conflict situations resolution. In this paper, a process traceability based methodology is proposed to support conflict management. This methodology deals mainly with the conflict resolution team identification and the solution impact evaluation issues. A process metamodel is proposed in order to allow the design process traceability and the data dependencies network identification; which makes it possible to identify the conflict resolution actors as well as to evaluate the selected solution impact.

Keywords: collaborative design, conflict management, process traceability, data dependencies network, product model.

1. Introduction

Collaborative design is a collection of the co-operated efforts undertaken by a team of designers and other specialists. Each team member works on different parts of the design, from different perspectives and towards fulfilling different functional criteria. Product design is involved in complicated interaction among multi-disciplinary design teams in a distributed, heterogeneous and dynamic environment, including communication, cooperation, coordination and negotiation [10].

Conflicts can be revealed form disagreements between designers about proposed designs due to multi-actors interaction. In fact, each actor has his own point of view, concerns and objectives regarding the design project. Hence, a critical element of collaborative design is the conflict resolution.

Conflicts can be defined as “an expressed struggle between at least two interdependent parties who perceive incompatible goals, scarce rewards, and interference from the other party in achieving their goals” [4]. 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 [6].

Insight for conflict resolution in collaborative design is available from diverse sources, which include approaches to conflict resolution in social sciences, computer-supported methods for facility design and conflict resolution in collaborative product design. We quote in particular Klein’s research works [5-6] which propose a conflict resolution method based on taxonomy of conflict solving strategies mapped with taxonomy of conflicts. A multi-approach method for computer-supported resolution of conflict situation is introduced by Lara and Nof [8] providing fast identification of conflict situation, diagnostics of conflict parameters and mechanisms for conflict resolution. Zhuang [13] focuses on conflict detection by providing a method of using a web based distributed design system and intelligent agents to detect conflicts.

We highlight also the CONCENSUS platform proposed by Cooper and Taleb Bendiab [3] and the CO²MED prototype proposed by Rose et al. [9] aiming at supporting multi-party negotiation as well as the system-mediated resolution proposed by Kim [7]. Barker et al. [1] propose a tool support negotiation in concurrent design teams. Zhao and Deng [12] propose an MAS prototype to model interaction behavior including communication, negotiation, coordination and cooperation.

The focus of all works mentioned above is on suggesting conflict detection mechanisms, defining conflict resolution strategies and providing support to facilitate the negotiation between the different actors involved in the conflict. Indeed, all these works reveal that negotiation is a widely accepted approach for conflict resolution in collaborative design. However, none of them touches upon the problematic of identifying the actors that should be involved in the negotiation process leading to problem solving. This identification phase of the negotiation team constitutes a pre-requisite for conflict resolution. The conflict management process could be perceived as the succession of five phases:

1) Conflict detection: to consider means of detecting conflict occurrence depending on the method used to
represent design constraints, design goals, design intents and design dependencies;

2) **Conflict resolution team identification and formation:** to identify and form the team of actors (human or Software) required to participate in the resolution of the identified conflict. Similarly, as the negotiation progresses and the existing conflict is being resolved or further conflicts are created, the negotiation team needs to be dynamically reformed. This phase is considered as a prerequisite to conflict resolution;

3) **Negotiation management:** to conduct and control a collaborative/competitive negotiation session, which often needs the presence of a mediating impartial senior authority or ‘chair’;

4) **Solution generation:** to apply actors own domain knowledge to provide an ‘optimal’ solution to the considered conflict;

5) **Solution impact evaluation:** to propagate the selected solution onto the product and the design process.

Likewise, neither the negotiation team identification phase nor the selected solution impact evaluation phase have been tackled in the above reviewed works. Indeed, a selected solution may lead to modifications on the design process organisation, on a subset of the product to design or on the availability of the resources for the design.

This work constitutes a continuation of a research work conducted in the Automatic Research Centre of Nancy-CRAN dealing with a methodological support to negotiation [9]. The current methodology provides elements to the points (3) and (4) making the project manager able to form his/her team thanks to a competences matrix and to an availability matrix answering therefore point (2). These matrices constitute an immediate answer but are inappropriate for a big project based on distributed sub projects.

The objective of this paper is to bring in methodological elements that allow, first, the identification of the actors who are to be involved in the conflict resolution and, second, the assessment of the propagation impact of the selected solution on the product development whole process.

Indeed, the first section of this paper describes a motivation example illustrating conflicts in a collaborative design. The second section proposes the methodology illustrating the diverse proposed elements on the motivation example. Finally, the paper concludes with a discussion of directions for future work.

**2. Motivation example**

Our case study concerns the design process of a flexible production system (FPS) within AIPL¹ organisation (cf. Figure 1). This system is mainly composed of an item loading station, two workstations, an item unloading station and a moving system of items between the stations composed of a conveyor and a palette. We are particularly interested in the workstation design process.

![Figure 1. Flexible Production System (FPS)](image)

In a collaborative design context, this process is composed of four concurrent sub processes:
- workstation frame design sub process
- workstation energy block (pneumatic and electric energy) design sub process
- workstation operative part (items positioning for the product assembly) design sub process
- command part (automata) design sub process

A special focus is given to the design of the operative part composed of a “handler” mechanism and a “positionner” mechanism. The handler is an arm controlled by the automata, which allows moving parts from the workstation stores to the palette. The “positionner” is made of three stores from which the handler picks up the parts to assemble and of a “director” responsible for directing the items towards a given position in order to guarantee the quality of the product.

The operative part design sub process, itself is split into two parallel processes: the “handler” design process and the “positionner” design process. At the beginning of the operative part sub process, the concerned actors have to respect the following requirements:
- the palette shape and the positioning perimeter of the wholes where to place the items;
- the automata’s, operation part’s and energy block’s positions;
- the jacks available for the actors to design their respective systems: three big jacks, four medium jacks, three small jacks and three rotary jacks.

The “handler” designer defines the mechanism by using the four medium jacks available (H_Solution1). This solution only allows four possible positions split into two positions on the workstation and two positions on the palette. Hence, the “positionner” designer is able to define only two “alimentations” composed of a vertical store with a spring each. A “director” is affected to each one of them (P_Solution1). Figure 2 partially describes the succession of these activities in an UML activity diagram [11].

A first conflict is then detected following from an acceptability problem of the “positionner” sub process solution by the Energy_Block designer. He realises that it is impossible to fit the energy block subset into the

¹ Atelier Inter-établissements de Productique – Lorraine. 
workstation frame because the stores’ lower parts lie within the frame dedicated to the energy block subset. Moreover, this volume problem results in the impossibility of alimentations stores’ loading as well as the inaccessibility to the maintenance of the energy subset.

Additionally, a second conflict appears: the project manager rejects the solution of a “director” (i.e. motor) for each alimentation because of the high cost of the motors. He requires that this subset is composed of only one engine.

A new configuration of the “positionner” subset is then proposed (P_Solution2) taking into account all the problems encountered earlier: the set is now made up of one mobile “director” with medium jack and two jacks (H_Solution2) accounting for the inaccessibility to the maintenance of the energy block subset. Additionally, a second conflict appears: the project manager into an output piece of data “defined wholes locations on palette” and this, respecting the “available jacks” constraint. The input data of an activity correspond to:

- Data produced by a previous activity within the same process, linked to a sequence flow [2].
- Data produced by a concurrent process activity, linked to message flow [2].
- Constraints predefined when the design process is launched, such as standards, previous projects ...
- Constraints emerging during the design process deployment due to the actors expertise, such as trade rules.

Consequently, a piece of data source of conflict, resulting from a transformation activity, is dependent on a whole set of input data. For instance, the data “wholes location on the palette” constitute a conflict data for the activity “to manufacture the wholes” executed by the “mechanic machining”. This data is produced by the activity “to propose handler solution” offering the solution H_Solution2 (“kinematics and pneumatic schema” and “jack type and number”). This activity has as input data the “positionner” configuration (“two Alimentations with vertical store each” and “one mobile director with medium jack”).

Therefore, to resolve the conflict, it is necessary to go back to the input data (“two Alimentations with vertical store each” and “one mobile director with medium jack”) of the activity (“to propose handler solution”); whereby the activity producing these data, as well as the actors performing it, are identified. Hence, identifying the actors to be involved in the conflict resolution process comes to identify the activities that produce the data on which the piece of data source of

3. A process traceability methodology to support conflict management

3.1. Specifications

The design process is a succession of activities handling data related to the product to be designed. To accomplish an activity, the actor that has been assigned to it (thanks to his know how) relies on input data to produce output data; i.e. to create new data in the case of creation activity or to transform data in the case of transformation activity. Figure 2 shows that in order to design a handler, the designer transforms the input piece of data “wholes location perimeter” provided by the project manager into an output piece of data “defined wholes locations on palette” and this, respecting the “available jacks” constraint. The input data of an activity correspond to:

- Data produced by a previous activity within the same process, linked to a sequence flow [2].
- Data produced by a concurrent process activity, linked to message flow [2].
- Constraints predefined when the design process is launched, such as standards, previous projects ...
- Constraints emerging during the design process deployment due to the actors expertise, such as trade rules.

Consequently, a piece of data source of conflict, resulting from a transformation activity, is dependent on a whole set of input data. For instance, the data “wholes location on the palette” constitute a conflict data for the activity “to manufacture the wholes” executed by the “mechanic machining”. This data is produced by the activity “to propose handler solution” offering the solution H_Solution2 (“kinematics and pneumatic schema” and “jack type and number”). This activity has as input data the “positionner” configuration (“two Alimentations with vertical store each” and “one mobile director with medium jack”).

Therefore, to resolve the conflict, it is necessary to go back to the input data (“two Alimentations with vertical store each” and “one mobile director with medium jack”) of the activity (“to propose handler solution”); whereby the activity producing these data, as well as the actors performing it, are identified. Hence, identifying the actors to be involved in the conflict resolution process comes to identify the activities that produce the data on which the piece of data source of

![Figure 2. Partial UML activity diagram of the operative part design process](image-url)
conflict depends. In fact, an actor is assigned to each of the design process activities.

Accordingly, the methodology proposed to identify the conflict resolution actors relies on building up a network that illustrates the dependencies between the handled data during the design process. Indeed, when a conflict appears, it would be easier, thanks to this network, to identify the data on which the piece of data source of conflict depends. Once these data are extracted, it is then possible to find out the activities that produce them and consequently the actors responsible for their execution.

In the following section, the different dependency types existing between design data are discussed.

3.2. Dependencies typologies

In the scenario Section 2, the data exchanged (proposed solutions) are presented in a text format. In a product data management context using Product Lifecycle Management (PLM) tools, this textual formalisation can hardly be exploitable. Indeed, it is difficult to associate the data (activity input/output) to the product model managing the product configurations and the technical data.

Moreover, in order to achieve a better use of the elaborated dependency network it is appropriate to distinguish several dependency levels between the data. For this purpose, we propose two dependency typologies: a syntactic typology and a semantic typology.

3.2.1. Syntactic typology

For a better management of the product model generated through the design process, it is necessary to well structure the data concept in order to identify the elements on which the data dependency lies. In fact, from a product model point of view, a piece of data, stated as an input or an output of a design activity, corresponds to:

- either the creation of a new class in the model: for example, as a result of “to propose handler solution” activity a new class “sensor” has been defined to add a component to the “handler”. This component has to be linked to the “jack” to which it is associated, thus resulting in the creation of a link between the “jack” and “sensor” classes. Figure 3 illustrates the different syntactic levels of data discussed above. From generic point of view, the FPS package is composed of the workstation package, the conveyor package, etc. which, in turn, are composed of other sub-packages. The sub-package “handler” is composed of the “jack” class/attributes and the “sensor” class/attributes which are linked together. To define a particular FPS, these packages, sub-packages and classes/attributes are then instantiated and valued.

Accordingly, a piece of data could be of one of the four types previously described (an existing class, a new class, an existing attribute or a new attribute). Consequently, a dependency link between two pieces of data could be between two classes, between two attributes or between a class and an attribute (already existing or newly created). Moreover, this dependency may concerns the creation or the instantiation/valuation of these concepts. Figure 4 explicit these dependencies.
For example, the dependency link between the “handler kinematics” and the “whole on palette” data is a dependency between the existence of the class “jack” and the existence of the attribute “location” of the class “wholes on palette”.

Figure 4. Syntactic level dependencies

3.2.2 Semantic typology

For a better management of the conflict resolution process, it is interesting to distinguish the nature of the dependency link that exists between the data. This would lead to the elaboration of various network filters providing the project manager with different points of view of the handled conflict source.

In this network, various types of dependencies can be distinguished:
- Inter-domain: dependency between data produced by actors form different domain. For example, the data “jack” and “workstation frame” are inter-domain dependent since they are produced by actors from different domain.
- Intra-domain: dependency between data produced by two actors from the same domain. For example, the data “jack” and “wholes on palette” are intra-domain dependent since they are produced by the same actor, hence the same domain.
- Inter-process: dependency between data produced in two different processes. For example, the data “handler volume” and the data “energyblock volume” are inter-process dependent since they are produced within different process.
- Intra-process: dependency between data produced within the same process. For example, the data “jack” and the data “alimentation” are intra-process dependent since they are produced within the same process (operative part design process).
- Inter-Bill of Material: dependency between different Bills of Material data (Functional, Structural and Geometrical), of the same product (article) or of two different products (articles).
- Intra-Bill of Material: dependency between the same Bill of Material data, of the same product (article) or of two different products (articles).

It is then possible thanks to these two typologies to establish the dependencies network according to various levels expressing progressively the classes linked to the dependent data, the syntactic dependency between these classes and the semantic dependency between these data.

In the following section, we will present the process model that would allow the traceability of the design process execution in order to extract the data dependency network.

3.3. Process model to build up the dependencies network

Figure 5. A metamodel for the traceability of design process
To build up the dependencies network discussed in section 3.1 and 3.2, the methodology is based on a tool to trace the execution of the design process activities taking into account the input and output data of each activity (a kind of “a posteriori” workflow). This tool relies on the proposed process metamodel (cf. Figure 5). This metamodel is the formalisation of the adopted concepts; i.e. activity (transformation activity, creation activity), resource executing it, flow between the activities (sequence flow and message flow [2]) and data handled by them (constraints issued from customer specification, data exchanged through a sequence flow and data exchanged through a message flow).

The “Justification” concept is added to extract the “know-how” adopted by the actor when transforming the data and thus the dependency links between the data. In fact, while the activity is taking place, the actor should justify and argue the transformations made on the data in order to achieve his objective. This justification can be done through various modes: with simple and common terms known by all, using equations and mathematical formulas, by associating appendices (abacus, standard…), by attaching plans CAD, by working out mechanical calculations…

- Once the conflict is detected, the data dependencies network is built up based on the process metamodel instances, the starting point being the data source of conflict. In order to build up this network, requests on the obtained model instances are elaborated by identifying the Input/Output data of any activity and the corresponding syntactic and semantic dependencies between the handled data as discussed in sections 3.1 and 3.2.

4. Conclusion

Nowadays, the product design actors evolve in a context of strong interactions sharing their knowledge and their competences. Being the most forced collaboration situation, the conflict management is the succession of five phases: conflict detection, conflict resolution team identification and formation, negotiation management, solution generation and solution impact evaluation. A motivation example, the FPS sub-process design, has been presented to illustrate conflict situations. This example allows the identification of a typology of the dependencies existing between the handled data during the design process. A methodology has been introduced to support conflict management; in particular conflict resolution team identification and formation phase. This methodology is based on a process traceability tool. In fact, this tool permits the building up of the data dependencies network to identify the conflict source dependent data as well as the activity that produces these data.

The identification of the data dependencies network would allow, on another hand, the propagation of the selected solution onto the identified dependent data. We could then evaluate the impact of any modification that might be generated by the selected solution once the conflict has been resolved. This would prevent the apparition of further conflicts.

However, further thoughts remain to be carried out for: 1) providing the tool with mechanisms allowing the assignment of weights to the data; the weight corresponds to the data importance in the design process; 2) establishing rules to set up collaborative space (activity) before the detection of the conflict, based on a “weighted” dependencies network. We, thus, speak about “a priori” conflict management process and not only “a posteriori” conflict management process. This method has to be implemented in a software prototype in order to validate it and to measure its effectiveness.

5. References