Tracking Design Dependencies to Support Conflict Management

Mohamed Zied Ouertani, Lilia Gzara, Gabriel Ris

Tracking Design Dependencies to Support Conflict Management

Mohamed-Zied Ouertani, Lilia Gzara, Gabriel Ris

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

Abstract. Due to the multi-actors interaction during collaborative design conflicts can be revealed from disagreements between designers about proposed designs. A critical element of collaborative design would be conflict resolution. This paper proposes a methodology to support designers during the process of conflict management; focusing mainly on identifying the conflict resolution team and evaluating the selected solution impact issues. The methodology is based on product data dependencies network which is composed of the handled data and the dependency links between them. Qualification concepts are introduced to manage these dependencies. In order to identify this network, a traceability model to track the design process is proposed.

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

1 Introduction

Collaborative 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 [11]. Due to the multi-actors interaction conflicts can be revealed from disagreements between designers about proposed designs. In fact, each actor has his own point of view, concerns and objectives regarding the design project. According to Klein [6], 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. Hence, a critical element of collaborative design would be the conflict resolution.

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. The focus of these research works is on suggesting conflict detection mechanisms [16], defining conflict resolution strategies [6] and providing support to facilitate the negotiation between the different actors involved in the conflict [1] [10] [15]. 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. For example, the development of a business jet may involve more than 140000 data which need to be managed appropriately by the development team. In order to provide a solution to a conflict occurring on one of these data, design actors have to collaborate and negotiate forming this way the *negotiation team*. Thus, it will be difficult first to identify who are the actors involved in the design process; and second, who are the right actors (competencies, knowledge, domain, etc.) to resolve the conflict rapidly and in a proper way; i.e. to avoid the appearance of new conflicts.

The conflict management process could be perceived as the succession of five phases:

- 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;
- Forming the conflict resolution team: to identify and form the team of actors (human or Software) required to participate in the resolution of the identified conflict. This phase is considered as a prerequisite to conflict resolution;
- Negotiation management: to conduct and control a collaborative negotiation session to reach a consensus;
- Solution generation: to apply actors own domain knowledge to provide the 'optimal' solution to the considered conflict;
- Solution impact evaluation: to propagate the selected solution onto the product and the design process.

Likewise, the selected solution impact evaluation phase has not 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.

The objective of this paper is to bring in methodological elements that allow the identification of the actors who are to be involved in the conflict resolution as well as the assessment of the propagation impact of the selected solution on the product development process. The first section of this paper describes a motivation example illustrating conflicts in a collaborative design. The second section presents the proposed methodology to support conflict management during design progress. The main concept of this methodology is the capture of the data dependencies network during design progress. This includes the proposal for a UML (Unified Modelling Language) traceability model to capture the product data dependencies. Finally, the paper concludes with a discussion of directions for future work.

2 Motivation Example

The case study concerns the design process of a Flexible Assembly System (FAS) at the training centre AIP-PRIMECA[1] Lorraine (AIPL). This system is mainly

[1] http://www.aipl.uhp-nancy.fr

composed of an item loading station, two workstations, an item unloading station and a transportation system composed of a conveyor and a palette. We are interested in the workstation design process; which is composed of four concurrent phases:
- workstation frame design sub phase;
- workstation energy block (pneumatic and electric energy) design phase;
- Assembly Operation System (items positioning) design phase;
- Automata design phase.

A special focus is given to the design of the assembly operation system which is 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 assembly operation system design phase is split into two parallel planned activities: the handler design process and the positionner design process. At the beginning of the assembly operation system phase, the concerned actors have to respect the following requirements:
- the palette shape and the positioning perimeter of the wholes to place the items;
- the automata's, assembly operation system'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). The UML activity diagram Fig. 1 partially describes the succession of these activities.

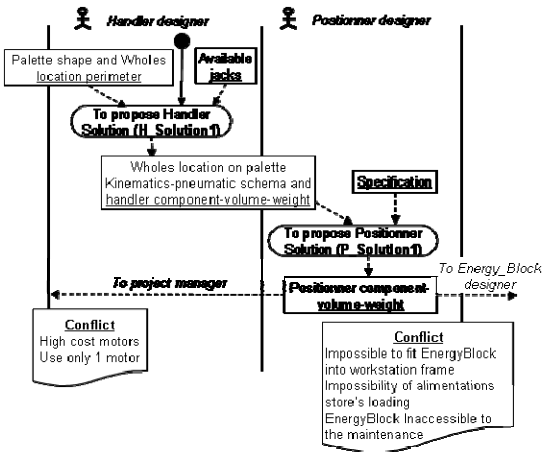

Fig. 1. Partial UML activity diagram of the operative part design process

A conflict is then detected following from an acceptability problem of the positionner planned activity solution by the Energy Block designer. He realises that it

is impossible to fit the energy block subset into the 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 to upload the alimentations stores' 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.

Consequently, new configurations of the positionner and the handler subsystems are to be proposed taking into account the problems encountered earlier.

The FAS design process scenario has been presented to illustrate the conflicts in a collaborative design context. In the following section, a methodology support the conflict management process is presented; the design process example is used to illustrate this methodology.

3 Design Traceability Methodology to Support Conflict Management

3.1 Specifications

The design process is a succession of activities handling product data to achieve the design objective. 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. Fig. 1 shows that in order to design a handler mechanism, the designer transforms the input data *wholes location perimeter* provided by the project manager into an output 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 design process, linked to a sequence flow [2].
- Data produced by a concurrent design 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, the 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* constitutes 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, it is necessary to resolve the conflict to go back to the input data (*positionner configuration*) of the activity (*to propose handler solution*); whereby the

activity producing these data and the responsible actors 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 data source of conflict depends; since an actor is assigned to each of the design process activities.

Accordingly, the methodology proposed to identify the conflict resolution actors is based on building up the dependencies network between the handled data during the design progress. Indeed, when a conflict appears, it would be easier using the network to identify the data on which the 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 data dependencies network is presented.

3.2 Data Dependencies Network

The data dependency network is an oriented graph composed of nodes, which correspond to the product data handled during the design process, and arcs, which correspond to the dependency relationships between these data. In the following, these network components are presented.

Network Nodes (Product Data). Design work returns with a succession of tasks to define a new product through the use and the generation of various product data. The handled data can be of several types: structural, functional and 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, simulations, etc.

Depending on the margin left to the designer to elaborate data, on the values of data properties and on the context in which it is committed, product data can evolve through different states. Grebici et al. identify four data states: draft, exhibit, enable and deliverable [5]. These states are presented in the following.

- The draft is a piece of data that one has to apply the modalities of creation and validation of hypothesis or solutions to a project or to a design problem. It is defined by a design actor individually. For instance, in the case study presented in Section 2, the handler designer produced a handler mechanism draft composed of kinematics and pneumatic schemes.
- The exhibit is a piece of data that one applies a persuasion modality in accordance with what is represented in either for convincing about the existence of a problem or for showing a solution and allowing a common construction and the point of view exchanged. For instance, the handler designer asks the machining expert for his opinion on the possibilities to manufacture holes (support of the pieces) of a given diameter D in a manner that they are aligned on the palette. The opinion of the machining expert allows then the handler designer to define the kinematics scheme.
- The enabled traces are data the designer accepts to diffuse to others, after his consent or his agreement with a collective prescription to which he takes part. It is non-officially validated objects but sufficiently convincing to be published. For example, the handler designer publishes his design parts when he is ensured that

the handler solution is validated. Indeed, the latter spreads the handler design parts in the official space to the positionner designer in order to allow him finishing his activity: to define positionner mechanism.

- The deliverable is data that transmit a strong regulation. They have been formally verified and validated (by hierarchy). They are those contractual supports to being communicated to the customer.

Network Arcs (Dependencies Links). Among the criteria that characterize links between elements (objects, tasks, design team, etc.), the dependency link criterion remains the more studied in research works and the less simple one to treat in the collaborative design modeling. Many definitions of dependency link are proposed in literature. According to Kusiak [8], a dependency between variables is the effect of change in a value of one variable on another variable, whereas for Jin [13], two components are said to have dependency relation if any of the two can not be completed without the other. These definitions reveal that two kinds of dependencies may exist between two product data: dependency at creation and dependency at modification. Two data are said "dependent at creation" if the creation of one of them depends on the creation of the second one – this corresponds to the dependency definition proposed in [13]. Two data are said "dependent at modification" if the change of one of them implies the modification of the second one – this corresponds to the dependency definition in [8]. The rest of this sub-section is devoted to characterise these dependencies. The main focus is on qualifying the dependency link through the concepts of dependency degree and dependency nature.

Dependency Degree. Depending on the dependency context (at creation or at modification), there exist different attributes that can be used to express the dependency degree between two elements in design.

Dependency at creation measure: Some research works have attempted to define attributes to express the link between data at creation, such as: the relevance, the usage and the completeness of the upstream data to the downstream data [4]. We are particularly interested in the completeness attributes which draw the actual data variation interval. The actor will express how should be the variation interval of the consumed data. Higher is the completeness attribute value, smaller would be the input data interval variation. Even if the completeness of data is defined by its user with regard to his needs to produce another data[2], it represents an absolute measure of the "at creation" dependency compared to the relevance and usage. In this paper, we are concerned with the estimation of the completeness attribute as the "at creation dependency" measure.

[2] A data could be considered as complete in order to be used for a given data and not for another one.

Table 1. constructed attribute for completeness

Attribute level	Description of the attribute level
0	*Weak*: the input data could be given below a certain maximum value
1	*Not Vital*: the input data should be given within a certain value range
2	*Vital*: the input data should be given with the smallest value range
3	*Extremely Vital:* the input data should be precisely given

The completeness of data provided by an activity is arbitrarily categorized in four levels according to a discrete and subjective measurement scale using structured expert interviews. Table 1 summarises the completeness attribute values.

Dependency at change measure*:* among the attributes proposed in science management and engineering design works to define the dependency , the following are the most relevant to measure the dependency "at change": *Level Number* [12], *Importance Ratings* [9] and *Probability of Repetition* [3]. We are particularly concerned with the estimation of the last measure of dependency since it constitutes the hardest to obtain input for simulating a development process that involves iteration. The *probability of Repetition* reflects the probability of one element (activity, data) causing rework in another. Krishnan et al. define the dependency measure as the multiplication of both attributes: *Variability* and *Sensitivity* [7].

Variability is the likelihood that the output data provided by one task would change after being initially released [14]. Since the variability is associated with the *stability* of a particular element, each output element has its own variability value. The variability concerns the results of an upstream activity output that constitutes input data for the downstream activity. It is difficult, if not impossible, to come up with a universal objective measurement scale for data variability to be used in all product development situations. Therefore, a discrete, subjective measurement scale for this measure is constructed using the techniques for constructing subjective attributes. The estimated variability of data provided by a task is arbitrarily categorized in four levels, each having a numerical value, as shown in Table 2.

Table 2. levels of data variability – adapted from [14]

Attribute level	Description of the attribute level
0	*Not variable*: the output data don't vary
1	*Low Variability:* the output data varies but few
2	*Moderate Variability*: the output data is instable
3	*High variability*: the output data is very instable

Sensitivity is the degree to which work is changed as the result of absorbing transferred data. In another words, this attribute expresses the sensitivity of output data (performed during a downstream activity) if the input data variation occurs (in the upstream activity producing this input data). Sensitivity depends on the level of dependency between two particular elements. Table 3 describes the three subjective levels of an element sensitivity developed using techniques for constructing subjective attributes.

Table 3. levels of data sensitivity – adapted from [14]

Attribute level	Description of the attribute level
0	*Not sensitive*: output data sensitivity is null to most input data changes
1	*Minor sensitivity*: output data sensitivity is low to most input data changes
2	*Moderate Sensitivity*: output data sensitivity is medium to most input data changes
3	*Major sensitivity:* output data sensitivity is high to most input data changes

In order to quantify the dependency link between two data, the dependency at creation and dependency at change measures are aggregated to one criterion to express the *dependency degree* between two data.

Therefore, the three attributes; completeness, variability and sensitivity are aggregated to measure the dependency degree (cf. Fig. 2.). As they are complementary attributes, a multiplicative utility function is utilised in the aggregation of the variability and sensitivity attributes (V*S). Furthermore, more the completeness is *high* (from *Not Vital* until *Extremely Vital*) more the required rework is long. Thus, for a given variability and sensitivity values, more the completeness is important, more the iterations are long and more the dependency degree is high. In the case when the variability value is "0" and the completeness value is different from "0", the dependency degree value must be different of "0", since that a not null completeness implies a dependency at creation. The dependency degree formula is presented in Fig. 2.

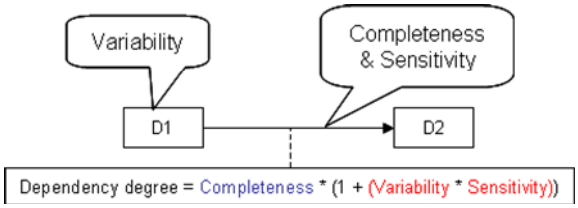

Fig. 2. Dependency degree between data D1 and D2

Accordingly, the resultant range value of the dependency degree is an integer between 0 and 30, whereas {0, 1, 2, 3 and 4} denotes a weak dependency and a low risk of rework, and {15, 20, 21 and 30} denotes a very high dependency and a high risk of rework. The values {5, 6, 7, and 8} and {9, 10, 12 and 14} describe respectively a moderate and a high dependency and risk of rework.

Dependencies Nature. 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, two various typologies can be distinguished:

- Syntactic typology: 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, either the addition of a new attribute to an existing, either the instantiation of an existing class or the valuation of an attribute in an existing instance. 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 concern the creation or the instantiation/valuation of these concepts. Fig. 3 summarises these dependencies.

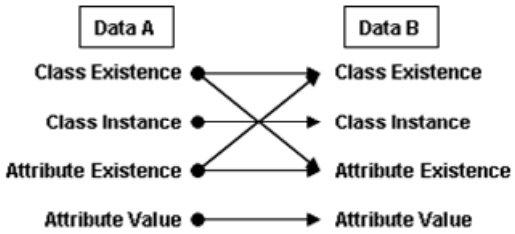

Fig. 3. Syntactic level dependencies

- Semantic typology. From a semantic point of view, various types of dependencies can be distinguished:
  - Inter-domain: dependency between data produced by actors from 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 design processes. For example, the data “handler volume” and the data “energyblock volume” are inter-process dependent since they are produced within different design processes.
  - Intra-process: dependency between data produced within the same design process. For example, the data “jack” and the data “alimentation” are intra-process dependent since they are produced within the same design process (operative part design process).
  - Inter-Bill of Material (BoM): dependency between different BoMs data (Functional, Structural, etc.), of the same product or of two different products.
  - Intra-Bill of Material: dependency between the same BoM data, of the same product or of two different products.

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 traceability model that would allow tracking the design progress in order to extract the data dependency network.

3.3 Traceability Model to Build up the Data Dependencies Network

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 model (cf. Fig. 4).

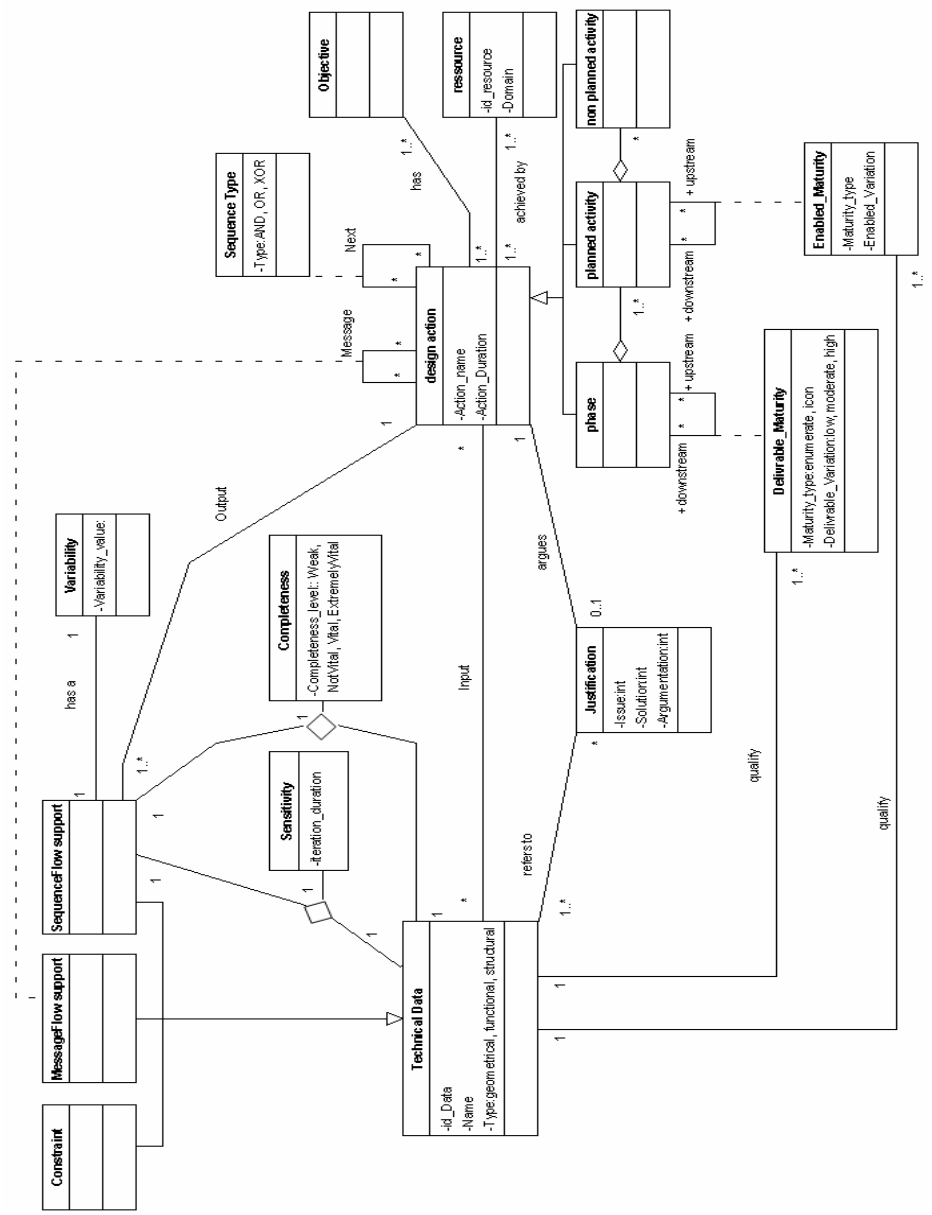

Fig. 4. The design process traceability model

This model is the formalisation of the adopted concepts; i.e. *activity*, *resource*, *flow* between the *activities* (sequence flow and message flow [2]) and *data* handled by them. The "Justification" concept is added to extract the "know-how" adopted by the actor when performing their activities. The key issues for the justification herein adopted are articulated as questions, with each *issue* followed by one *solution* among

several *alternatives* that responds to the issue. *Arguments* support the adopted solution and could be of Requirement-based, Rule-based, Case-based, etc.

Once the conflict is detected, the data dependencies network is built up based on the model 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 each activity and the corresponding syntactic and semantic dependencies between the handled data as discussed previously.

4 Summary and Future Work

The conflict management process is a succession of five phases: conflict detection, formation of the conflict resolution team, negotiation management, solution generation and solution impact evaluation. In this paper, a methodology has been introduced to support conflict management; in particular the phases of negotiation team formation and of impact propagation on product data and design process organisation. The main purpose of the proposed methodology is to explicitly capture the product data dependencies during the design progress. Concepts such as the dependency degree which is based on the completeness, variability and sensitivity attributes as well as the dependency nature (i.e. syntactic and semantic) are proposed to qualify product data dependencies. Then an UML traceability model is proposed to identify these dependencies; which are represented throughout a data dependencies network. The FAS design process has been used to illustrate the whole methodology.

However, further thoughts remain to be carried out for the design process re-organisation problematic. In fact, based on the data dependencies network, designers are able to identify negotiators and to propagate modifications on the already defined product data. Nevertheless, these modifications often require a re-execution of activities producing product data to be modified. In addition, the concerned actors are not often available since they can ensure different roles in several concurrent projects whose delivery dates are predefined. Consequently, the design process reorganisation proves to be a difficult task since it depends on availabilities of the actors able to execute these modifications.

References

1. Barker R., Holloway L.P., Meehan A.: Supporting Negotiation in Concurrent Design Teams. The Sixth International Conference on CSCW in Design. London, Ontario, Canada IEEE (2001).
2. BPMN, 2004, Business Process Management Notation Specification, Version 1.0. OMG/BPMI Report. http://www.bpmn.org/.
3. Browning T.R., Eppinger S.D.: Modelling Impacts of Process Architecture on Cost and Schedule Risk in Product Development. IEEE Transactions on Engineering Management, Vol. 49(4), (2002) 428-442.

4. Culley S.J., Davies S., Hicks B.J., McMahon C.A.: An assessment of quality measures for engineering information sources. 15th International Conference on Engineering Design. August 15-18. Melbourne, Australia (2005).
5. Grebici K., Blanco E., Rieu D.: Toward Non Mature Information Management in Collaborative Design Processes. International Conference on Engineering Design (ICED'05). August 15 – 18. Melbourne, Australia. (2005).
6. Klein M.: Supporting conflict resolution in cooperative design systems. IEEE Transactions on Systems, Man and Cybernetics Vol. 21(6), (1993) 1379-1390.
7. Krishnan V., Eppinger S.D., Whitney D.E.: A Model-Based Framework for Overlapping Product Development Activities. Management Science, Vol. 43(4), (1997) 437-451.
8. Kusiak A., Wang J.: Dependency analysis in constraint negotiation. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, (1995) 1301- 1313.
9. Pimmler T.U., Eppinger S.D.: Integration Analysis of Product Decompositions. The ASME Design Theory and Methodology Conference (DTM'94). (1994).
10. Rose B., Gzara L., Lombard M.: Towards a formalization of collaboration entities to manage conflicts appearing in cooperative product design, in: *Methods and Tools for Cooperative and Integrated Design*, S Tichkiewitch and D Brissaud, Editors. Kluwer Academic. Printed in The Netherlands. (2004).
11. Shen W., Norrie D.H., Barthès J.P.: Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing, Taylor and Francis, London, UK. (2001). (0-7484-0882-7).
12. Steward D.V.: System Analysis and Management: Structure, Strategy and Design, New York: Petrocelli Books, NY, USA. (1981).
13. Wang K.L., Jin Y.: Modelling dependencies in engineering design. The ASME Design Theory and Methodology Conference, September 12-15. Las Vegas, Nevada (1999).
14. Yassine A., Falkenburg D.R., Chelst K.: Engineering Design Management: An Information Structure Approach. International Journal of Production Research, Vol. 37(13), (1999) 2957- 2975.
15. Zhao G., Deng J.: Cooperative Product Design Process Modelling. The Sixth International Conference on CSCW in Design. London, Ontario, Canada IEEE (2001).
16. Zhuang R., Conflict Detection in Web Based Concurrent Engineering Design. (1999), Master Thesis, University of Florida: FLorida.