Managing Data Dependencies to Support Conflict Management

Mohamed Zied Ouertani, Lilia Gzara Yesilbas, Muriel Lombard, Gabriel Ris

Managing Data Dependencies to Support Conflict Management

M.Z. Ouertani, L. Gzara-Yesilbas, M. Lombard
CRAN - Automatic Research Centre of Nancy, CNRS UMR 7039
University Henri Poincaré, Nancy I, Faculty of Sciences, BP 239
54506 Vandoeuvre-lès-Nancy - FRANCE
Mohamed-zied.ouertani@cran.uhp-nancy.fr

Abstract
Fundamental in engineering design is the notion that collaboration is useful to reduce total project development time and increase design quality. Collaborative design is a collection of the co-operated efforts undertaken by a team of designers. Due to multi-actors interaction, conflicts can emerge from disagreements between designers about proposed designs. Therefore, a critical element of collaborative design would be conflict resolution. In this paper, a process meta-model based methodology is introduced to support conflict management. This methodology provides identification of the conflict resolution team and evaluation of the impact of a selected solution. The proposed process model allows the design process traceability and the data dependences network management to achieve conflict management.

Keywords:
Conflict Management, Process Traceability, Data Dependencies Network

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 [1].

Due to multi-actors interaction, conflicts can emerge from disagreements between designers about proposed designs. In fact, each actor has his own point of view, concerns and objectives regarding the design project. Thus, a critical element of collaborative design would be the conflict management.

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” [2]. 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 [3].

Insight for conflict resolution in collaborative design is available from diverse contexts, which include social sciences approaches, computer-supported methods for facility design and conflict resolution in collaborative product design. In particular, Klein’s research works [3] [4] propose a conflict resolution method based on taxonomy of conflict solving strategies mapped with taxonomy of conflicts. A methodology for analysing collaborative design process based on a social-technical design framework is proposed by Lu *et al.* [5]. It also provides mechanisms to detect and manage conflicts. A multi-approach method for computer-supported resolution of conflict situation is introduced by Lara and Nof [6] providing fast identification of conflict situation, diagnostics of conflict parameters and mechanisms for conflict resolution in facility design.

To support multi-party negotiation, several research works such as the CONCENSUS platform [7], the system-mediated resolution [8] and the CO²MED prototype [9] have been conducted. In addition, Barker *et al.* propose a tool to support negotiation in concurrent design teams [10], while Zhao and Deng propose an MAS prototype to model interaction behaviour including communication, negotiation, coordination and cooperation [11].

The focus of all works mentioned above is on suggesting conflict detection mechanisms, defining conflict resolution strategies and providing support to facilitate 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. In the mean time, a significant progress has been made in negotiation theory and its application in engineering design [12].

However, none of the considered works has touched upon the problematic of identifying the actors that should be involved in the negotiation process that leads to problem solving. This phase of *negotiation team identification* constitutes a pre-requisite for conflict resolution. 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 come up with methodological elements that allow, first, identification of the actors to be involved in the conflict resolution process and, second, assessment of the selected solution impact on the whole product development process.

The remaining part of the paper is organised as follows. In section 2, the problematic of negotiation team identification and selected solution impact is examined. In section 3, the concept of data dependency in engineering design is studied and dependencies typology will be proposed. In section 4, a process based methodology to extract data dependencies is proposed in order to answer the tackled problematic. Section 5 summarises the key elements of the paper and identifies some future researches.

2 PROBLEMATIC

The conflict management process could be perceived as the succession of five phases (Figure.1):

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 and 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 assessment: to propagate the selected solution onto the product and the design process.

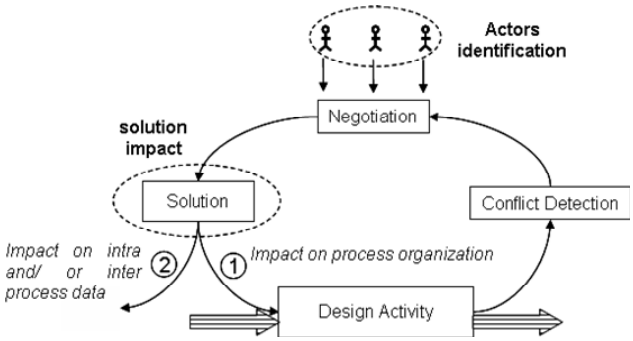

Figure 1: Conflict management process

The phases of negotiation management and solution generation were tackled in a previous research work at CRAN Laboratory proposing a methodological support to negotiation: the CO2MED protocol [9]. This paper aims at completing this methodology by covering the phases of negotiation team identification and solution impact assessment. With regards to this last phase, two types of impacts are distinguished: impact on the handled technical data and impact on the design process organisation. In the following sections, both concepts of negotiation team and impact of a selected solution are studied.

2.1 Negotiation team in design process

Conflict resolution cannot be achieved by one single actor; it requires the gathering of different expertise areas. So to avoid much iteration in the conflict resolution process, it is highly recommended to do it in a collaborative way that seeks the interference of many actors to reach quickly a consensus. In order to provide a solution to the detected conflict, design actors have to collaborate and negotiate forming this way the *negotiation team*. The negotiation phase is composed of the succession of popularisation and mediation activities [9] of the problem to be solved in order to reach a consensus. Different roles are identified and assigned to negotiation team members to better coordinate their activities. These roles are:

Initiator: the actor detecting the conflict.

Supervisor: the responsible for releasing the solution resulting from the conflict resolution process. This role can be assumed by the manager of the project where the conflict was detected, or a managerial person, if the data source of conflict is critical.

Negotiators: actors directly involved in the realisation of the conflict source data (the process leading to the data). A conflict data is the result of one or more previous activities in the design process. The actor responsible for its creation, as well as the actors already intervening in the process and producing data influencing the value of data, should be involved to know the reasons for creating their data (with the given values) and to invite them to their modification.

Hence, identifying negotiators supposes knowing the dependency relationships between the conflict source data and the data previously produced.

2.2 Impact of selected solution in design process

Impact on product data

One of the key factors influencing the conflict management process is the propagation of the modifications on the whole product data set. Indeed, the negotiation phase leads to a solution which 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. In fact, the design process as a succession of activities which transform input data into output data, defines a precedence link between these data (i.e. an output of an activity becomes the input of successive activities).

The handled data can be of three types: structural, functional and geometrical. They correspond to the various descriptions of the product elaborated by the designer, at the early stage of the design process, from the customer specifications and requirements (description with geometrical entities, functional, structural, technological and behavioural descriptions). A direct link exists between these various types of data. Indeed, it is possible to define the product composition and its corresponding structure through the functional specifications. Thus, a modification on functional data may have an impact on structural data.

Accordingly, it is necessary to take into account all the data dependency links during the propagation of the modifications resulting from the conflict resolution. Hence, a data dependency management framework is required to ensure modifications propagation.

Therefore, it is important to identify the actors responsible for the data exposed to be impacted in order to inform them about the modifications to be made so that they can act accordingly – these actors are often assigned as *subscribers*.

Impact on design process

In a context of engineering design, in particular collaborative design, activities should be planned in order to support and coordinate designers work according to the activities' precedence constraints and the material and human resources availability. Yet it is difficult to completely define design process a priori. This could be explained by the fact that the design process prescription often neglects specific factors and constraints with which designers are confronted during their daily work (economic constraints, tight timelines, team work difficulties, etc.) [13]. Nevertheless, general guidelines could be provided and the necessary resources allocated. Indeed, the process organisation should be tailored in order to manage design as a project and to promote actors collaboration.

For the elementary activities (i.e. task or operation), the organisation is identified a posteriori. The structure and the most appropriate plan of this organisation are determined through the activities dependencies. It is the need of each activity to input data that generates these dependency links. Indeed, an activity consumes and transforms input data into output data to be used by the successive activities.

Once a conflict appears, the resolution process generates a solution which often brings about data modifications. For a process still in the pipeline, these modifications require an adjustment to the preliminary project organisation. However, the processes that have already been launched and executed are re-examined and their re-execution is necessary. But the organisation of the activities to be re-executed is difficult to manage because of the availability of the allocated resources and the increasingly tight delivery deadlines.

Therefore, one of the key elements to be taken into account in the methodology is the process organisation impact caused by the modifications generated by the chosen solution following from the conflict resolution process. Impact propagation is highly dependent on handled data during design process.

2.3 Approach

From the previous analysis (§2.1. and §2.2.), it arises that identifying negotiation team as well as propagating selected solution impact means identifying the dependency relationships between technical data handled during the design process.

Indeed, in collaborative design, the dependencies of engineering problems determine how and with whom designers should coordinate. Consequently, solving these two phases of the conflict management process would mean:

- Identifying the dependencies network of the data handled during the design process execution in order to define the negotiation team.

- Qualifying the data dependency links in order to propagate the impact of the chosen solution.

3 DATA DEPENDECY

Many definitions of data dependency are proposed in literature. We particularly retained the definition proposed by Kusiak and Wang [14], for whom a dependency between variables is the effect of change in a value of one variable on another variable and the definition of Jin and Wang [15], for whom two components are said to have

dependency relation if any of the two can not be completed without the other.

The concept of data dependency is developed in this section through, first, a case study where examples of data dependency are given to illustrate our proposal (§3.1) and, second, the proposition of a typology for data dependencies in product design context (§3.2).

3.1 Data dependencies examples

This section presents a case study illustrating data dependencies examples. The case study concerns the design process of a Flexible Production System (FPS) within AIPL[1] organisation (Figure 2). The system is mainly composed of an item loading station, two workstations, an item unloading station and a transportation system of items between stations. The latter is composed of a conveyor and a palette. We are particularly interested in the workstation design process which is composed of four concurrent sub processes:

- Workstation frame design sub process;

- Workstation energy block (pneumatic and electric energy) design sub process;

- Operative part (items positioning for the product assembly) design sub process;

- Command part (automata) design sub process.

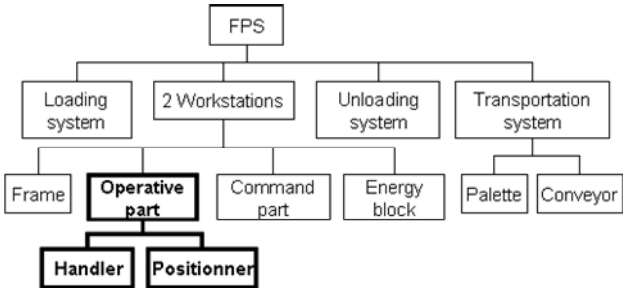

Figure 2: The FPS structural decomposition

A special focus is given to the design of the operative part which is composed of a *handler* mechanism and a *positionner* mechanism. The *handler* is an arm controlled by the automata, that allows moving items from the workstation stores to the palette. The *positionner* is made of three stores from which the *handler* picks up the items to assemble and of a *director* to guarantee the quality of the product.

The operative part design sub process is split into two parallel processes: the *handler* design process and the *positionner* design process. At the beginning of the operative part design 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 on the workstation frame;

- the jacks available for the actors (*handler* and *positionner* designers) to design their respective systems: three big jacks, four medium jacks, three small jacks and three rotary jacks.

According to these specifications, the *handler* designer defines the mechanism by using the four medium jacks available. This solution only allows reaching four possible positions; two positions on the workstation and two positions on the palette. Thus, the *handler* designer defined the structure, kinematics and volume of the sub-system to be designed (*handler* mechanism). Consequently, the *positionner* designer is able to define only two *stores* composed of a vertical *stockpile* with a *spring* each. A *director* is affected to each one of the *stores*. Likewise, the *positionner* designer has defined the structure and the volume of the sub system to be designed.

These characteristic data of the operative part sub systems are then used by the *energy block* and the *workstation frame* designers to be able to fulfil their respective design activities. Indeed, in order for the *energy block* designer to define the characteristics of his sub system, it is necessary to know the *handler* kinematics and the number of *jacks* used. As for the *workstation frame* designer, he will need the *handler* volume data and the *handler* kinematics data to be able to specify the frame structure and the type of material to be used. Figure. 3 recapitulateS these dependencies – an arrow defines the direction of a dependency.

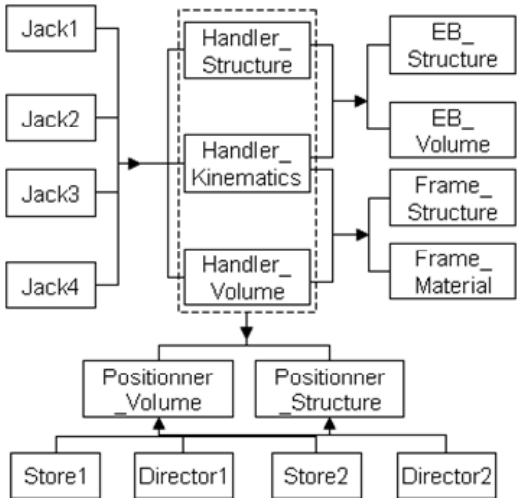

Figure 3: Precedence dependency links between handled data

In the following section, a dependency typology is presented. The qualification of these dependencies would allow the identification and the management of the data dependency links in order to achieve better conflict resolution.

[1] Atelier Inter-établissements de Productique – Lorraine. http://www.aipl.uhp-nancy.fr

3.2 Typology of data dependencies

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:

- **Creation of a new class in the model**: for example, when defining the *handler* part decomposition, a new component is defined (such as the component *jack*); this corresponds to the creation of a new class (*jack*) on the product model.

- **Addition of a new attribute to an existing class**: for example, when defining the kinematics of the *handler*, a new characteristic of the component *store* is defined (the stores' position); this corresponds to the creation of a new attribute *position* in the class *store*.

- **Instantiation of an existing class**: for example, once the class *jack* is added, the first time it is used in the workstation, the following actors needing jacks in their design tasks would instantiate the existing class *jack* as many as they add jacks in the product.

- **Valuation of an attribute in an existing instance**: for example, the *frame dimension* of the workstation is defined only after the completion of the *operative part*, *command part* and *energy block* deign processes.

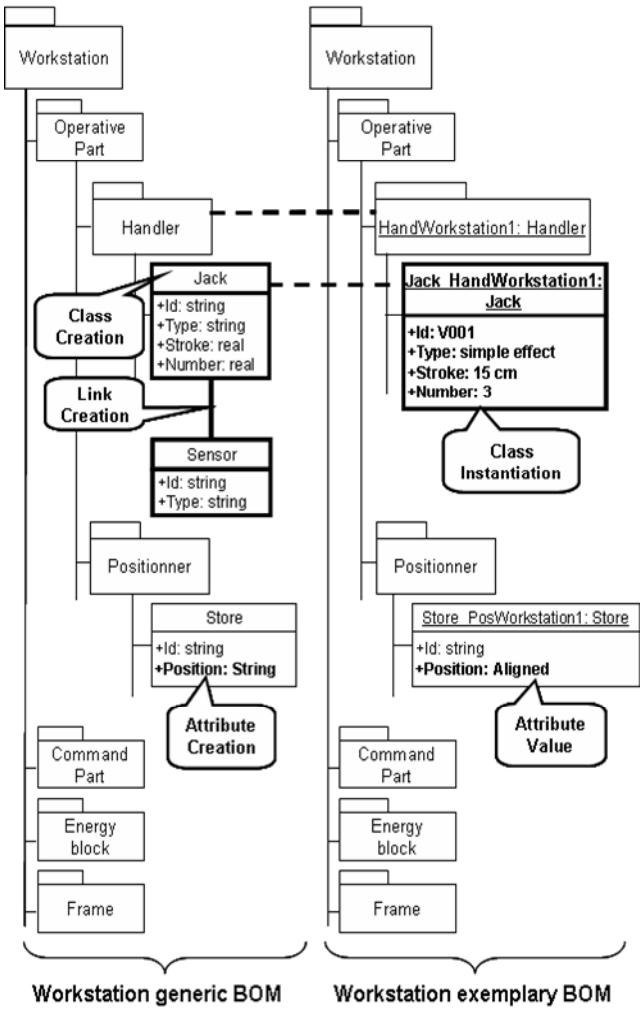

Figure 4: Syntactic data level

We should note that the creation of a class is often accompanied by the creation of links between this new class and the product model existing classes. For example, when defining the component *sensor* of the *handler* the new class *sensor* is added to the product model. 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 4 illustrates the different syntactic levels of data discussed above. The workstation package is composed of the *operative part* package, the *command part* 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 the workstation part, these packages, sub-packages and classes/attributes are then instantiated and valuated.

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 concern the creation or the instantiation/valuation of these concepts. Figure 5 explicit the possible dependencies between the four types, shown with an arrow.

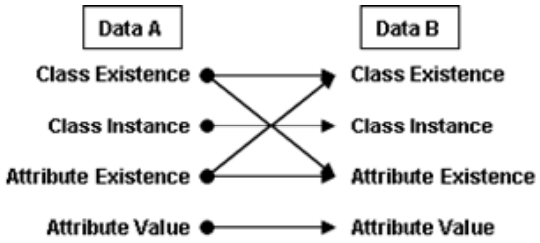

Figure 5: Syntactic level dependencies

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 from a resource perspective, a process perspective and a product perspective.

Resource perspective dependencies

Design process actors, coming from different domains and adopting various knowledge, are provided with various expertises (mechanic, electric, aerodynamic, etc.). To reach their objectives, actors collaborate through data exchange. Depending on actor's domain, two dependencies types are distinguished:

- Inter-domain: dependency between data produced by actors form different domains. For example, the data *jack* and *workstation frame* are inter-domain dependent since they are produced by actors the *pneumatic domain* for the

handler designer and the *mechanical domain* for the *workstation frame designer*.

- Intra-domain: dependency between data produced by two actors from the same domain. For example, the data *jack* of the *handler* mechanism and the data *store* of the *positionner* mechanism are intra-domain dependent since they are produced by actors from the same domain; the *pneumatic* domain.

Process perspective dependencies

In a collaborative design context, activities are executed simultaneously in concurrent projects. Data are exchanged between these activities, either through sequence links between activities of the same process (i.e. sequence flow [16]), or between concurrent processes through data shared spaces or messages such as e-mails, etc. (i.e. message flow [16]). According to the exchanged data support, two types of dependencies are distinguished:

- Inter-process: dependency between data produced in two different processes. For example, the data *handler* and the data *palette* are inter process dependent since they result from two concurrent sub-processes: the *handler* and the *palette* design sub-processes.

- Intra-process: dependency between data produced within the same process. For example, the data *jack* and the data *sensor* are intra process dependent since they result from the same sub-processes; i.e. the *handler* design sub-process.

Product perspective dependencies

To achieve a given objective, design actors act (transform or create) during design activities execution on product data input. These data are representations of product which can be of different natures. The output data can be in their turn of different nature than those from the input data. According to the nature of an activity Input/Output data, two types of dependencies are then distinguished:

- Inter-Bill of Material (BoM): dependency between elements from different BoMs (Functional, Structural and Geometrical) of the same product or two different products. For example, the linked data *jack kinematics* and *store position* are inter-BoM dependent: functional and geometrical BoMs.

- Intra-Bill of Material: dependency between the same BoM, of the same product or of two different products.

3.3 Data dependency in literature

In this section, a brief state of art on research works dealing with data dependency in design process is given. These research works have tackled the problem along two major research directions:

Dependency identification: since engineering design is dependency-oriented and often involves multiple perspectives, it is assumed that the initial values of some design variables are determined by the corresponding perspectives and propagated to the remaining variables through a set of constraints. Then, when a conflict occurs for a variable, a suitable value is selected for that variable and it is propagated backward to the variables that precede the conflicting variable. To do that, and due to the existence of cycles and multipaths in a network of constraints, Kannapan and Marshak proposed ad-hoc heuristics to support backward chaining path [17]. To avoid the existence of cycles and multipaths in a network of constraints, Jin and Wang proposed a framework of engineering dependency based on a generic design process model and a set of composition and combination dependencies patterns to avoid reciprocal and cyclic dependencies in design and thus optimizing the collaborative design process [15]. On a different side, Park and Cutkosky [18] tackle the problem of dependencies but from an activity point of view. He proposes a framework to model activities' dependencies in a collaborative design process. As data dependency is strongly lying on activity dependencies, Park and Cutkosky approach can be useful to identify data dependencies.

Dependency qualification: even though research works on propagation, such as Kannapan and Marshak proposition, allow the identification of the variables to modify, they do not allow the determination of the nature of modifications to be run. Then, some researchers have focused on dependency qualification. Kusiak and Wang developed an approach to analyse dependencies among design variables and constraints to assist designers in negotiation of violated constraints [14]. This approach is based on a dependency network used to represent qualitative (i.e. to determine the direction of change of a variable affected by another variable) and quantitative (i.e. the rate of change of a variable affected by another variable) dependencies among design variables, constraints and goals. This dependency qualification allows the modification impact of a design variable on another one. We finally quote the work of Jin and Wang which categorized dependencies as natural dependency, product dependency, task dependency and designer dependency [15].

Even if these approaches have proposed elements to manage design variables dependencies; some problems have not been tackled. Regarding dependencies identification, current works define the dependencies a priori; i.e. at early stage before the launching of the design process. However, during activities execution, new data dependencies may appear since know and know-how deployed by the actors may differ from one actor to another and may change according the design context. In addition, new know and know-how may emerge during product design. Indeed, these proposals are based on process models known in advance. The dependencies are already prescribed and no research work tackled the dynamic identification of dependencies during process execution.

In terms of dependencies qualification, those suggested in the literature do not take into account the various dependency types that a designer can encounter during the design process execution (cf. §3.2). Indeed, the

proposed method by Kusiak and Wang [14], by qualifying the dependencies of qualitative and quantitative, does not take into account the dependencies between two elements of different nature such as described by Jin and Wang [15]. However, by proposing a qualification of the dependencies according to the nature of the studied entities (process or data), the proposed approach does not take into account the modifications impact of an entity on another.

4 IDENTIFICATION APPROACH BASED ON PROCESS TRACEABILITY

The objective of the proposed approach is to come up with methodological elements that allow the identification of data dependencies and then their qualification. In order to do so, the first step is to trace the real design process by storing it in data base tables. Then, SQL queries are applied on the obtained data to extract data dependencies network (cf. Figure 6).

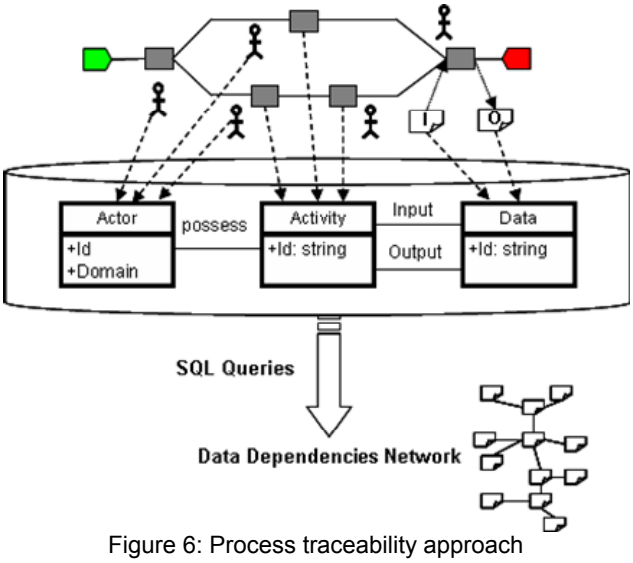

Figure 6: Process traceability approach

Traceability in product development is defined by Hamilton and Beeby as the ability to *discover the design history of every feature of a product*" [19]. Traceability has been considered as an important quality attribute [20] and can be defined as a quality factor of designing [21]. Tracing the design process is a property of a product development environment with a goal to ensure that product development context (data, information and engineering knowledge that evolves throughout the product development) is clearly linked to its sources in design representation created as the result of the product life-cycle.

Traceability dimensions can be described by answering the basic questions adopted from Zachman [22]:

What are the traceable items – design objects, requirements, constraints, design decisions, rationale behind these decisions etc. – that are managed during the design process.

Who are the actors playing different roles in the creation, maintenance and use of those items.

Where are those items being handled (i.e. created or transformed). Which activities did consume/handle the items.

How are these activities being performed in order to handle the items; the design rationale behind the decision taken during the various activities in product development.

Why are items being created or transformed; i.e. the objective of the activities where the items were handled.

When are items being created or transformed. Some of this property would be date/time the item was created/transformed.

However, traceability needs well-defined formal support for describing process and design. In this respect, a meta-model taking into account the traceability dimensions discussed above is proposed. This meta-model is presented in the following section.

4.1 Meta-Model presentation

On the basis of the traceability dimensions presented previously and in order to extract the data dependencies network which will allow first, the identification of the negotiation team members and second, the propagation of eventual changes on the whole product, the main items of the meta-model for achieving traceability in product development are determined. Figure 7 describes all constructs needed to generate traceability model as well as their semantics. It is build following the dimensions necessary for achieving traceability in design process.

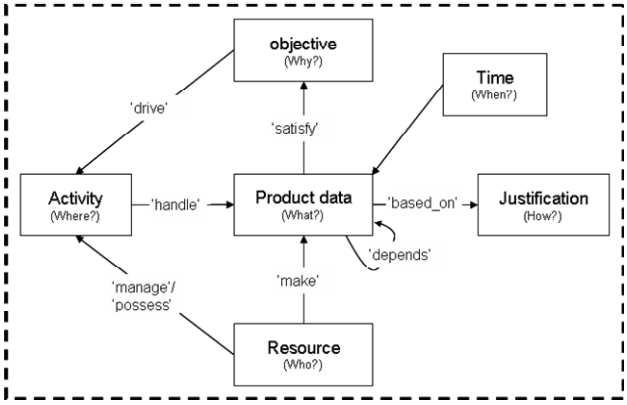

Figure 7: Process model constructs for achieving traceability

Product data are the inputs and outputs of product development process. Two data classification types are identified: first, according to the handled data type (Functional, Geometrical and Structural data) and second, according to the handled data exchange support (data produced by a previous activity within the same process, linked to a sequence flow [16], data produced by a concurrent process activity, linked to message flow [16] and constraints).

Activity is an action carried out during the design process (i.e. data creation or transformation), by one or more resources having various roles, to satisfy a given objective. An activity can be an operation (if it is elementary) or process (if it is made up of other activities).

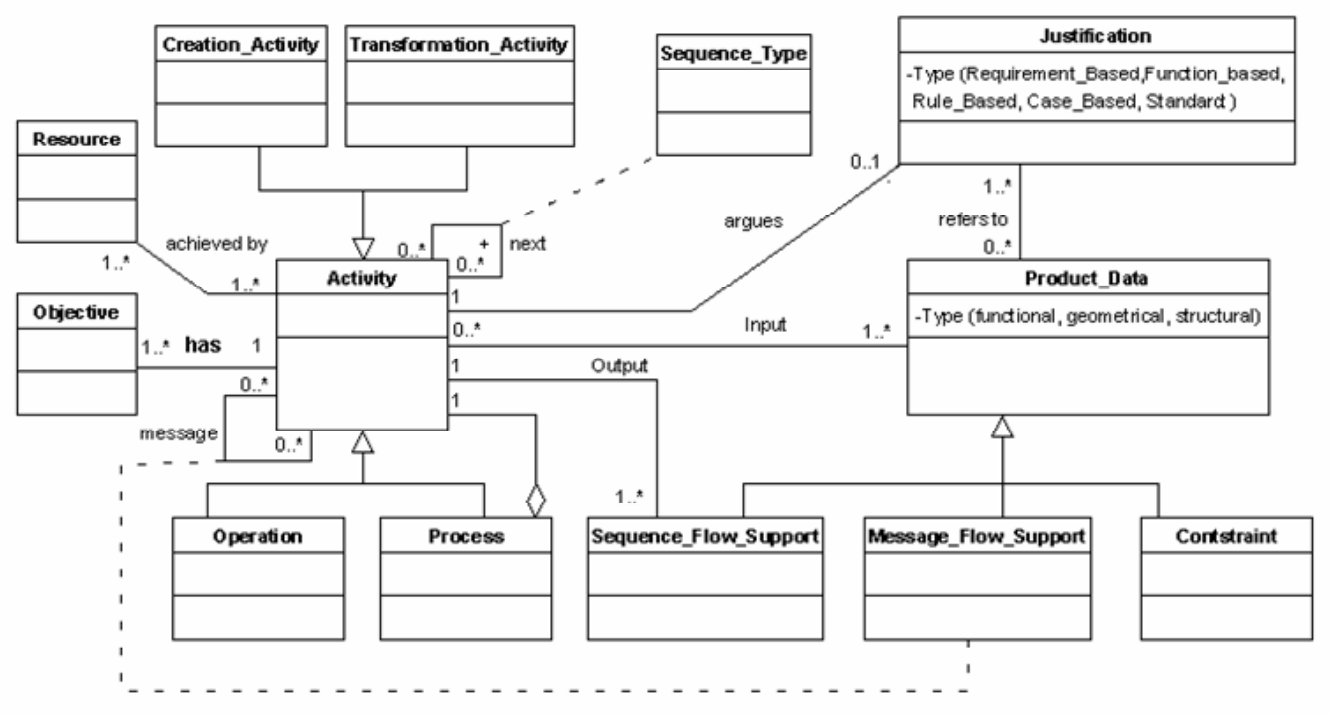

Figure 8: The UML process meta-model for achieving traceability

Resources are the human and material resources necessary to carry out an activity. It could be an actor or group of actors or material resources. They act in different roles in creating or transforming product data. Roles are characterised by resource knowledge, competencies, rights and obligations. The role notion has to be considered in the meta-model. Indeed, the same resource can get involved in the process according to various roles depending on the activity to be executed.

Justification represents the design rationale behind the creation and the transformation of the handled product data. 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 etc. The "Justification" allows extracting the know-how adopted by the actor when handling 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.

Objective represents the need of the handled data during the design process.

In a context of Information System development, the previously introduced concepts (traceability items) are structured using the UML language [23]. Figure 8 represents a model formalizing the concepts that have to be managed in order to achieve the design process traceability.

4.2 Meta-model instantiation

The instantiation of the meta-model (Figure 8) will allow the traceability of the design process onto a database. Indeed, the actor should inform the meta-model fields when starting his design activity; first by specifying which project the design activity belongs to, then, by specifying the activity to execute (name, Id...), its nature (transformation or creation), the input and the output data, etc. For example, the *handler designer*, when starting the design of the *handler mechanism*, specifies to which project this activity belongs: the *operative part design sub-process*. Then, he specifies the activity itself: *to propose handler solution*. The activity consumes input data: *palette shape*, *specifications* and *handler volume*, to achieve an objective which will be represented as output data: *handler structure*, *handler kinematics* and *handler volume*. Following from this activity, the *handler designer* would create the data *handler structure* and *handler kinematics* and transform the data *handler volume* into *handler defined volume*. The UML Object Diagram Figure 9 illustrates this instantiation.

This step must be applied by all the involved actors during each one of their activities. Thus, a trace of the design process execution will be saved onto a database following from the meta-model instantiation.

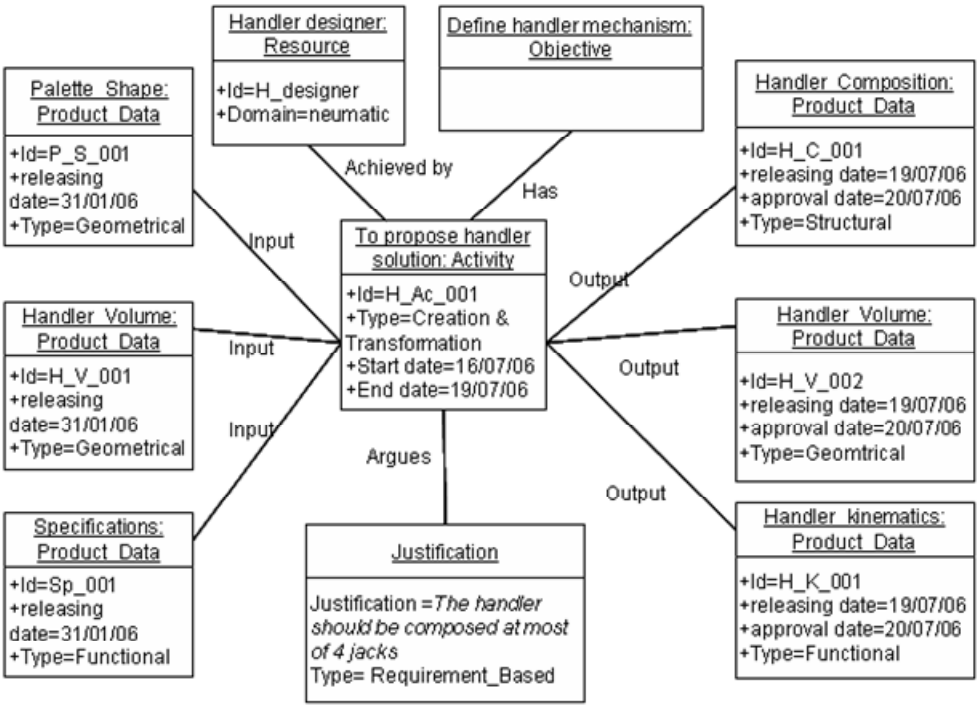

Figure 9: UML object diagram of the handler designer activity

Once a conflict appears, a set of SQL queries is applied in order to collect the dependencies of the conflict source data. As a data is produced by an activity which consumed (transformed, created) input data, a first query – *Activity Query* – allows the identification of the conflict source data (C for instance) producing activity.

Activity Query

Select Activity Id From Activity

Where OutputData Id = X ;

A second query – *Data Query* – allows identifying the consumed input data in order to produce the conflict source data.

Data Query

Select InputData Id From Activity_Data

Where Activity Id = (Select Activity Id From Activity Where OutputData Id = C ;) ;

As specified in section § 2.1., a conflict source data is defined according to the input data (I) of the activity producing it (a). Thus, any change of this data questions the input data (I) values. In the same way, these input data (I) are the outputs of a certain previous activity (b). Then, modifying the values of the data (I) questions the values of the input data (B) of the activity (b).

While applying this reasoning to the whole backward activities, the modification of the conflict source data will thus imply the modification of a set of already handled data. It is then necessary to apply, as many times as it is necessary, the queries *Activity Query* and *Data Query* to the data and activities preceding the conflict appearance. A dependency network is thus identified.

This dependency network allows identifying, first, the dependent data, second, the activities producing these data and third the actors executing these activities. Then negotiation team is formed and subscribers are identified (cf. §2.2.1.) to propagate modifications.

In the following paragraph, the approach is illustrated on the case study. A conflict is detected following from an acceptability problem of the *positionner mechanism solution* by the *energy block designer*. He recognises that it is not possible 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 of the stores' loading as well as the inaccessibility to the maintenance of the *energy block* subset.

The *energy block designer* identifies the data dependent on the conflict source data through the data dependency network. These data are: *positionner solution*, *handler solution*, *workstation frame* and *energy block frame*. Then, the actors responsible of these data are invited to negotiate, via the CO²MED negotiation protocol [9], to resolve the conflict. The latter are *positionner*, *handler*, *energy block* and *workstation* designers (cf. Figure 10).

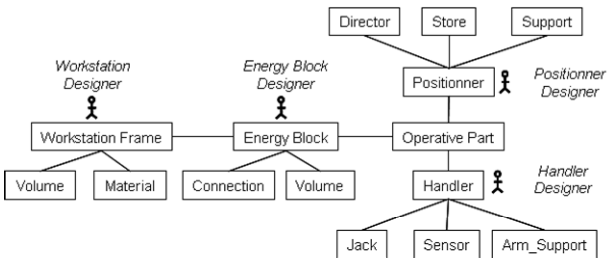

Figure 10: FPS partial data dependencies network

As discussed in section § 2.2.1., the selected solution may impact on a set of data; whose producing activities have consumed the data to be modified. For this purpose, concerned actors (subscribers – cf. § 2.2.1.) are supported by the data dependencies network and the dependencies typology discussed in section § 3.2.. Consequently, the actor is able to assess the modification impact. Indeed, the dependencies syntactic typology allows the definition of the modification that should be done (creation of a new data, modification of a data attribute or addition of a new attribute), and the dependencies semantic typology allows to specify the necessary competences to modify data, to locate the activity to be re-executed (producing activity of the data to be modified), and to define the modification type (geometrical, structural or functional product data modification).

For example, the *handler designer* should modify the data *handler solution*. In order to do so, he modifies the subset structure by adding a new component, a *jack*. The addition of this new component questions the already defined components. Consequently, it will be necessary to add another component, the *sensor*, and modify the characteristics of the other predefined *jack*, such as decreasing their stroke.

5 SUMMARY AND CONCLUDING REMARKS

A methodology has been introduced to support conflict management; in particular negotiation team formation and impact propagation on product data phases. In fact, being the most forced collaboration situation, the conflict management is perceived as the succession of five phases: conflict detection, conflict resolution team identification and formation, negotiation management, solution generation and solution impact evaluation. The proposed methodology is based on a process traceability method. The latter allows building up the data dependencies network to identify the conflict source dependent data as well as the activity that produces these data. This paper presents also a discussion of the typology of data dependencies during the design process. Semantic and syntactic topology is proposed and illustrated with examples.

However, further thoughts remain to be carried out for the process re-organisation problematic. In fact, the proposed methodology presents a support for conflict management. Based on data dependencies network, designers are able to identify negotiators and to propagate modifications on previously defined product data. However, these modifications often require a re-execution of activities producing data to be modified. Actors can ensure different roles in several concurrent projects whose delivery dates are predefined. Consequently, the processes reorganisation proves to be a difficult task since it depends on availabilities of the actors able to execute these modifications.

In order to do so, it is necessary, first to identify the projects to reorganise, since a data can be used in several concurrent processes; and next, to define the needs to these processes and the objectives to achieve with them. The third phase would be to model and to analyse these processes. A description of the different aspects of the processes (i.e. technological, resources, etc.) is then to be given. The execution and the coordination of the activities, the exchanged data and the allocated resources are to be analysed. Finally, based on the result of the modelling and the analysis phases, the process can be properly redesigned.

Consequently, it is necessary to take into account this problematic in the proposed methodology to propose a framework allowing the management of the impacts on the product data as well as on the engineering process. Especially that the data dependencies network proposes responses elements to processes reorganisation problematic (those that have already been executed). In fact, it is possible to identify trough the network, first, the resources (roles and competences) necessary to the activities re-execution; and second, the concurrent processes to reorganise.

While as for the planned processes but not yet executed, metrics are to be proposed in order to guarantee an optimised adjustment for the impact assessment. These metrics constitute a dashboard to the project manager and concerns essentially the number of impacted activities, the impacted projects, the resources to be assigned (competences, availabilities, etc.).

Further issues could be: (1) providing the methodology 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 before the detection of the conflict, based on a “weighted” dependencies network. We, thus, speak about “a priori” conflict management and not only “a posteriori” one.

6 REFERENCES

[1] Shen, W, Norrie, D-H., Barthès, J-P, 2001, Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing, Taylor and Francis. London, UK.

[2] Hocker, J-L., and Wilmot, W-W., 1985, Interpersonal Conflict, 2nd Edition, William C. Brown, Dubuque, IA.

[3] Klein, M., 1995, Conflict management as part of an integrated exception handling approach, Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, vol. 9, 259-267.

[4] Klein, M., 1994, Integrated support for cooperative design coordination: Managing processes, conflicts and memories, In: Information and collaboration

models of integration (S.Y.Nof ed.), Kluwer Academic Publishers. Printed in the Netherlands, 435-459.

[5] Lu, S. C-Y., Cai, J., Burkett, W., Udwadia, F., 2000, A Methodology for Collaborative Design Process and Conflict Analysis, CIRP Annals. 49(1), 69-73.

[6] Lara M-A., and Nof, S-Y., 2003, Computer-supported conflict resolution for collaborative facility designers, International Journal of Production Research, vol. 41, no. 2, 207-234.

[7] Cooper, S., and Taleb-Bendiab, A., 1998, CONCENSUS: multi-party negotiation support for conflict resolution in concurrent engineering design, Journal of Intelligent Manufacturing, vol. 9, 155–159.

[8] Kim, M-O., 2002, Coping with conflict in concurrent design environment, ACM SIGGROUP Bulletin, vol 23, Issue 1, 20 – 23.

[9] Rose, B., Gzara, L., and Lombard, M., 2004, Towards a formalization of collaboration entities to manage conflicts appearing in cooperative product design, Methods and Tools for Cooperative and Integrated Design, Kluwer Academic Publishers, 12 p.

[10] Barker, R., Holloway L-P., and Meehan, A., 2001, Supporting Negotiation in Concurrent Design Teams, Proceedings of the Sixth International Conference on CSCW in Design, 243 – 248.

[11] Zhao, G., Deng, J., 2001, Cooperative Product Design Process Modelling, Proceedings of the Sixth International Conference on CSCW in Design, 236-242.

[12] Lu, S. C-Y. 2004, Engineering as Collaborative Negotiation: a new paradigm for collaborative engineering research, CIRP Working Group on Engineering as Collaborative Negotiation, http://wisdom.usc.edu/ecn/

[13] Ehrlenspiel, K., 1999, Practicians — How they are designing? … and Why ?, Proceedings ICED 99, Munich, Vol. 2, 721-726.

[14] Kusiak, A., and Wang, J., 1995, Dependency Analysis in Constraint Negotiation, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, No. 9, 1301-1313.

[15] Jin, Y., and Wang, K., 2000 Modelling and managing dependencies in engineering design, in CIRP Annals, Vol.49 (2).

[16] BPMN, 2004, OMG/BPMI Business Process Management Notation Specification, Version 1.0. http://www.bpmn.org/

[17] Park, H., Cutkosky, M-R, 1999, Framework for Modeling Dependencies in Collaborative Engineering Processes, Research in Engineering Design 11. 2, 84-102.

[18] Kannapan, S-M., and Marshek, K-M., 1992, Engineering Design Methodologies: A New Perspective, Intellignet Design and Manufacturing, Kusiak, A., Ed., John and Wiley & Sons, Inc., New York, 3-38.

[19] Hamilton, V-L., and Beeby, M-L., 1991, Issues of Traceability in Integrating Tools, Proc. IEE Colloquium Tools and Techniques for Maintaining Traceability During Design.

[20] Gotel, O-C-Z., and Finkelstein, A-C-W., 1994, An analysis of requirements traceability problem, Proceedings of the IEEE International Conference on Requirements Engineering, Colorado Springs.

[21] Štorga, M., 2004, Traceability in product development, Proceedings of the design 2004 / Marjanović, Dorian (ed.), Glasgow: the design Society, FSB.

[22] Zachman, J-A., 1987, A Framework for Information Systems Architecture, IBM Systems Journal, Vol. 26, No. 3, 276-292.

[23] UML, 2004, OMG Unified Modelling Language Superstructure Specification Version 2.0, http://www.omg.org/docs/formal/05-07-04.pdf.