DEPNET: A Methodology for Identifying and Qualifying Dependencies Between Engineering Data

Mohamed Zied Ouertani, Khadidja Grebici, Lilia Gzara, Eric Blanco, Dominique Rieu

DEPNET: A Methodology for Identifying and Qualifying Dependencies Between Engineering Data

M. Z. Ouertani [1, #], K. Grebici [2,3], L. Gzara [1], E. Blanco [2], D. Rieu [3]

[1] CRAN-Nancy University, BP239, Vandœuvre-lès-Nancy 54506, France;
[2] GILCO-INPGrenoble, 46 Avenue Félix Viallet, 38031, Grenoble, France;
[3] LSR-SIGMA- IMAG, 2 Place Doyen Gosse-38031-Grenoble , France.

Abstract

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, the DEPNET methodology is introduced to support conflict management. This methodology is based on a Unified Modelling Language (UML) traceability model to extract the data dependencies network. This will allow identifying the conflict resolution team as well as evaluating the impact of a selected solution. A case study within an industrial partner is described to illustrate this methodology.

Keywords

Conflict Management, Data Dependencies Network, Case Study.

Introduction

Due to multi-actors interaction in collaborative product design, conflicts can emerge from disagreements between designers about proposed designs [5]. Thus, a critical element of collaborative design would be the conflict management; which could be perceived as the succession of mainly five

phases [9]: (1) Conflict detection, (2) Identification of the conflict resolution team, (3) Negotiation management, (4) Solution generation, and (5) Solution impact assessment.

In this paper, the main objective is to come up with methodological elements to allow the team identification and the assessment of the solution impact on product and process organisation.

The identification of negotiation team constitutes a pre-requisite for conflict resolution. Indeed, conflict resolution cannot be achieved by one single actor; it requires the gathering of different expertise areas to avoid unnecessary iteration. In order to provide a solution to the detected conflict, design actors have to collaborate and negotiate forming this way the *negotiation team*. It is then important to identify the right actors to resolve a conflict, else new conflicts could occur.

Once a solution is generated, the latter leads often to modifications on a subset of the already defined product parts as well as on the process organisation. Indeed, the negotiation phase leads to a solution which often implies changing 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. These modifications require a re-execution of a set of design activities necessary to achieve changes, and also an adjustment to the preliminary project organisation.

Therefore, negotiators identification and impact assessment are highly dependent on handled data during the design progress. This supposes knowing the dependency links between the conflict source data and the data previously produced. Consequently, solving these two phases mean:

- Identifying the dependencies network of the data handled during the design progress in order to define the negotiation team.
- Qualifying the data dependency links in order to assess the impact of the selected solution.

The remaining part of this paper is organised as follows. In section 2, the data dependencies network components are described. This network is composed of nodes and arcs representing respectively the handled data during the design process and the dependency links existing between them as well. Section 3 presents the DEPNET (product **D**ata d**EP**endencies **NET**work identification and qualification) methodology to identify this network. This methodology is based on a UML traceability model to track the design progress; allowing this way to extract the data dependencies network. Section 4 describes a case study to illustrate our approach. Finally, section 5 concludes with some perspectives.

Data Dependencies Network

Network Nodes: Handled 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 drawing, 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. [3] identify four data states: draft, exhibit, enable and deliverable, according to the workspace where they are handled: private, proximity, project and public workspaces. These concepts are summarised in the following:

- 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 a design problem. They are defined by a design actor individually.
- 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.
- Enabled traces are data the designer accepts to diffuse to others, after his agreement with a collective prescription to which he takes part. It is non-officially validated data but sufficiently convincing to be published.
- Deliverable are 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: Dependency Links

According to Kusiak [7], a dependency between variables is the effect of change in a value of one variable on another variable. Whereas, for Wang [10], 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*. Both of these dependencies kinds will allow us to qualify a dependency link.

Dependency at creation

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 in [10]. Some research works have attempted to define attributes to express this link, such as: the *relevance*, the *usage* and the *completeness* [2]. We are particularly interested in *the completeness* attributes which draw the actual data variation interval. The design actor should 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. The completeness attribute is then considered as the "at creation dependency" measure. The completeness attribute values are:

- 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

Dependency at change:

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 [7]. Attributes such as *Level Number*, *Importance Ratings*, and *Probability of Repetition* [1] were proposed to define this dependency link. We focused on the probability of repetition attribute since it constitutes the hardest to obtain input for simulating an iterative development process. The *Probability of Repetition* reflects the probability of one element causing rework in another. Krishnan et al. [6] defines the dependency at change measure as the multiplication of both attributes: *Variability* and *Sensitivity*.

Variability is the likelihood that the output data provided by one task would change after being initially released. The variability measurement scale is:

- 0 Null: the output data don't vary
- 1 Low: the output data varies but few
- 2 Moderate: the output data is instable
- 3 High: the output data is very instable

Sensitivity is the degree to which work is changed as the result of absorbing transferred product data. The sensitivity measurement scale is:

- 0 Null: output sensitivity is null to most input changes
- 1 Minor: output sensitivity is low to most input changes
- 2 Moderate: output sensitivity is medium to most input changes
- 3 Major: output sensitivity is high to most input changes.

Dependency degree

In order to qualify the dependency link between two data, the dependency at creation and the dependency at change measures are aggregated to one criterion to express the *dependency degree* between two data. Therefore, the attributes completeness, variability and sensitivity are aggregated to measure the dependency degree (cf. Eq.1). As they are complementary attribute, a multiplicative utility function is utilised in the aggregation of the variability and sensitivity attributes (**V*S**). 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.

$$\textit{Dependency Degree} = \textit{Completeness} * (1 + (\textit{Variability} * \textit{Sensitivity})) \quad (1)$$

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.

DEPNET to Build up the Data Dependencies Network

The objective of the DEPNET methodology 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 design process progress by storing it in a database system. Then, a set of queries are applied on the obtained data to extract the network.

Traceability Model

Traceability in product development is defined by Hamilton and Beeby [4] as the ability to discover the design history of every feature of a product. Traceability dimensions can be described by answering the basic questions adopted from the Zachman framework [11]:

What are the traceable items: refers to the design objects, requirements, design decisions, etc. which will allow building up the data dependencies network. As design process deals mainly with consuming, exchanging, communicating and producing product data, the traceable items to represent by the data dependencies network are the handled product data.

Where the traceable items are: refers to the design actions handling the product data during the design process. Two management levels of the design process exist, the prescribed one and the emerging one. At the prescribed level, the process is composed of phases which are composed of planned activities. The emergent level corresponds to the non planned activities occurring during the design progress.

Who are the resources playing different roles in the creation, modification and exchange of the product data.

Why – How product data are created, modified and/or evolved the way it is; this corresponds to the design rationale behind the design actions.

When are the product data being created, modified and/or evolved; this corresponds to the starting and finishing date of the design actions (phase, planned activities and/or non planned activities).

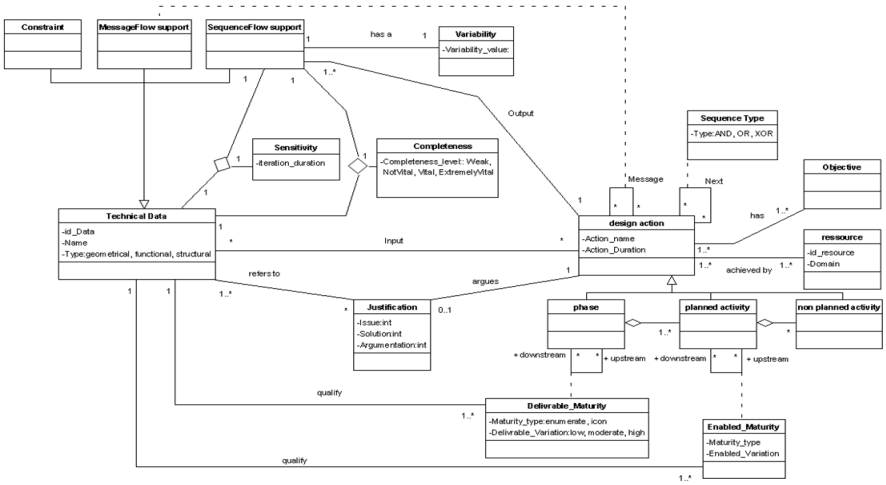

Fig. 1. UML traceability model to track data dependencies during design progress

To allow tracking the design progress in a database system, the various constructs discussed above are formalized in the UML model Fig. 1.

From Traceability Model to Data Dependencies Network

Based on the model presented in Fig. 1, a traceability tool was defined. This tool is a kind of an *a posteriori* workflow tool which allows declaring the design ongoing. This tool allows each actor involved in the design process to declare, when achieving his design action: the phase, planned activity or non planned activity he is executing; the objective if his design action; the input and the output data used and generated with the

associated sensitivity, variability and completeness measures, as well as the maturity attributes (for enabled or deliverable maturities); and, the justification of the choices made during the design action.

These declarations must be done by all the involved actors during each one of their design actions. Once the design process ongoing is declared, the captured constructs are stored in a database whose tables correspond to the various classes of the traceability model described above. Indeed, each process element declared by actors corresponds to an instantiation of one of the database tables. Then, a set of queries can be applied to the stored data in order to collect the dependencies of the conflict source data. These queries are applied as many times as it is necessary in order to identify the whole conflict dependant data. Once the dependency network is extracted, it is possible to identify the negotiation team and the design activities to be re-executed. The data on which depends the piece of data source of conflict are identified through the network backward coverage; the piece of data source of conflict is the starting point. Then, a set of queries are applied in order to identify the actors responsible of theses data realisation. The formers (i.e. these actors) will then be part of the negotiation team that will resolve the conflict. We should note that the negotiation team could be dynamic during the conflict resolution process. Indeed, as the conflict resolution process goes on, new problems/conflicts may be detected. It is then necessary to invite the most qualified actors to solve these. As a result, the dependencies network could be used several times in the conflict resolution and the negotiation team composition could vary over time. Once a solution to the conflict is selected, the impacted data are identified through the data dependency network (i.e. a forward cover of the network starting from the data to which leads the selected solution).

Case Study: Turbocharger Design Process

The case study described in this section concerns the design process of a turbocharger within our industrial partner. The mechanical concept of a turbocharger is based on three main parts. A *Turbine Wheel* which is driven by the exhaust gas from a pump to spin the second main part, an *Impeller* – i.e. a Compressor Wheel – whose function is to force more air into the pump's intake, or air supply. The third basic part is a *Center Hub Rotating Assembly* (CHRA) which contains bearing, oil circuit, cooling, and a shaft that directly connects the turbine and impeller. At the beginning of the turbocharger design process, the concerned actors have at their disposal a set of specification as well as of requirements to respect.

According to these specifications, the impeller designer – i.e. compressor wheel process design responsible – starts his planned activity "to define the impeller part". The latter has to define the impeller attributes composed of wheel cast-material, expected compressor inlet/outlet temperature etc. Once these attributes completed, the impeller designer define the exducer and inducer diameters of the compressor wheel as well as the 3D CAD drawing. The last task of the impeller designer is to define the impeller housing and how should this part be connected to the engine. In order to do it, the impeller designer calculates the impeller attributes Trim and A/R[1]. These attributes make the designer able to finish the impeller 3D CAD drawing.

Based on customer data, turbocharger specifications and impeller defined attributes, the turbine designer – i.e. turbine wheel process design responsible – concurrently starts his planned activity "to define the turbine wheel". The turbine designer defines first a set of turbine attributes to reach the turbocharger performance. These attributes are composed of the wheel, nozzle ring and insert ring material, the inlet/outlet turbine pressure. Once these attributes defined, the turbine designer starts defining the wheel dimensions as well as the 3D CAD drawing of the turbine wheel. The wheel dimensions consist of calculating the exducer and inducer diameters. Then, the designer finishes his part design by defining the turbine housing by calculating the turbine attributes Trim and A/R.

Concurrently to the impeller and turbine parts definition activities, the CHRA designers specify their parts; based on the impeller and turbine defined parts and the turbocharger specifications. The CHRA designers have to define the bearing system (frame size, diameter, etc.), oil circuit (filtration, seals, etc.), shaft, etc. In order to do so, the designers have to exchange preliminary information making possible the process progress. Not only must the components within the turbocharger itself be precisely coordinated, but the turbocharger and the engine it services must also be exactly matched. If they're not, engine inefficiency and even damage can be the results. Thus, it's important that the concerned actors collaborate closely by coordinating their activities as well as the data exchange. Indeed, the different parts are highly dependent and modifying one of them impacts the others. Figure 2 recapitulates the precedence dependencies between handled data – an arrow defines the direction of a dependency.

Suppose that the turbine designer detect a conflict when defining the turbine 3D CAD drawing. In order to resolve this conflict, the negotiation team is formed. The negotiation members are those they participated in the design process leading to the turbine 3D CAD drawing. Hence, the team

[1] A/R is the inlet cross-sectional area divided by the radius from the turbo centerline to the centroid.

members are: the turbine designer, the impeller designer, the innovation team, the customer, the project manager etc. Once this team resolves the conflict, the modification impact is propagated according to the data dependencies network. Starting from the data to be modified the impact is propagated on the whole data dependencies network.

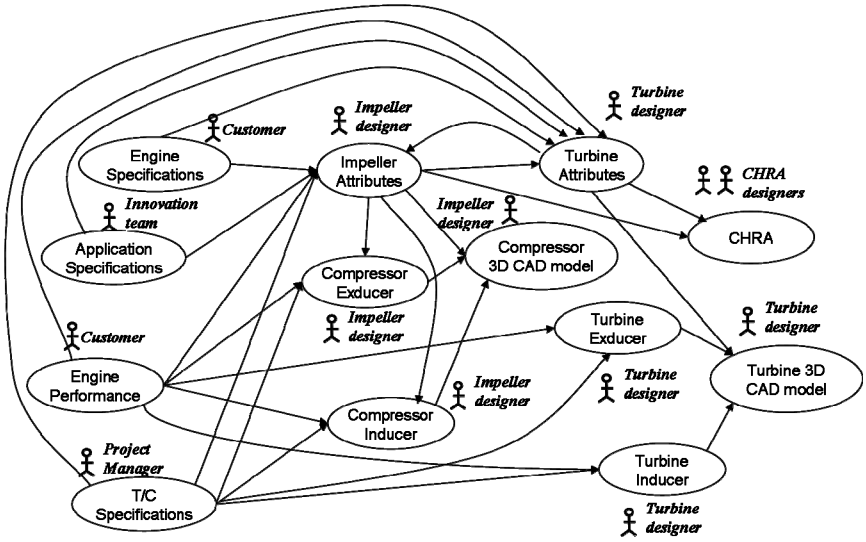

Fig. 2. Partial view of the data dependencies links during the turbocharger design.

Conclusions

In this paper, the DEPNET methodology has been introduced to support conflict management; in particular negotiation team formation and impact propagation on product data and process organisation. The proposed methodology is based on a process traceability method support to building up the data dependencies network composed of nodes i.e. product data and arcs i.e. dependency links. A tool is developed in order to implement this methodology and an illustration is done with a simplified case study [8].

However, further thoughts remain to be carried out for the process re-organisation problematic. In fact, the DEPNET 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. These modifications often require a re-execution of activities producing data to be modified. 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 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. Strategies to manage the overlapping of coupled product development activities are to be proposed to answer questions such as: when should downstream activity act on upstream data? How should the activities be overlapped when downstream activity cannot work on preliminary data?

References

1. Browning TR, Eppinger SD (2002) Modelling Impacts of Process Architecture on Cost and Schedule Risk in Product Development. IEEE Transactions on Engineering Management, Vol. 49(4), 428-442.
2. Culley SJ, Davies S, Hicks BJ, McMahon CA (2005) An assessment of quality measures for engineering information sources. 15th ICED Melbourne,Australia.
3. Grebici K, Blanco E, Rieu D (2005) Toward Non Mature Information Management in Collaborative Design Processes. 15th ICED. Melbourne, Australia.
4. Hamilton VL, Beeby ML (1991) Issues of Traceability in Integrating Tools. IEE Colloquium Tools and Techniques for Maintaining Traceability During Design. London, UK.
5. Klein M (1995) Conflict Management as Part of an Integrated Exception Handling Approach. AI in Engineering Design Analysis and Manufacturing, Vol. 9, 259-267.
6. Krishnan V, Eppinger SD, Whitney DE (1997) A Model-Based Framework for Overlapping Product Development Activities. Management Science, Vol. 43(4), 437-451.
7. Kusiak A, Wang J (1995) Dependency analysis in constraint negotiation. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, 1301- 1313.
8. Ouertani MZ, Gzara L (Submitted) Tracking Product Information Dependencies in Collaborative Design for Conflict and Change Management, Computer Aided Design Journal.
9. Ouertani MZ, Gzara-Yesilbas L, Lombard M, Ris G (2006) Managing Data Dependencies to Support Conflict Management. Proceedings of 16th CIRP International Design Seminar, Kananaskis, Alberta, Canada, pp 342-352.
10. Wang KL, Jin Y (2000) Managing dependencies for collaborative design. The ASME DETC/CIE conference. Baltimore, MA.
11. Zachman JA (1987) A Framework for Information Systems Architecture. IBM Systems Journal, Vol. 26(3), 276-292.