The role of associations in CAD and PLM for handling change propagation during product development

T.G.TREMBLAY, L.RIVEST[1], O.MSAAF, R.MARANZANA
Dept of Automated Production Engineering, Ecole de technologie superieure, Montreal

Abstract. Current Computer Aided Design (CAD) systems can capture some of the design intent by creating associations between objects. This increases the productivity during product development, and helps maintain the coherence of the product definition when handling engineering changes. CAD systems establish some associations as well as their use at a rather low level of abstraction, e.g. a parallelism constraint. Product Lifecycle Management (PLM) systems, on the other hand, use associations at a higher abstraction level, generally between files. The associations handled by these systems do differ both in terms of abstraction level and formalism of the knowledge they encapsulate. Moreover, limited connections exist between the associations manipulated in both systems, so that handling change propagation in concurrent engineering remains an issue. In this paper, we propose a taxonomy and a model of the different types of associations required to support the set of tasks in the area of product development. The terms on which the taxonomy relies are: association, relation, link, and constraint. The proposed model, named RLC, uses the concepts of aggregation and decomposition to relate Relations, Links, and Constraints.

Keywords. Associations, Relations, Links, Constraints, CAD, PLM, RLC model, Change propagation, Change Management, Data consistency, Decomposition, Aggregation

Introduction

In today's competitive market, delays and production costs of new products are very important considerations. Companies overcome these factors by depending on approaches such as concurrent engineering. On the technical level, this requires management of a digital model of an evolving product, as well as the distribution of information in multiple files (CAD, Computer-Aided Manufacturing, CAM, tooling, etc.). To conserve coherence within this digital model, the implementation and effective management of a network of associations within each file, as well as between the files which make up the product model, is essential.

One should differentiate evolutions of the product definition during design from evolutions of released data. In the first case, a multidisciplinary design team works simultaneously on various aspects of the product, the definition of which is in progress. These evolutions can be called Corrections (Maurino [1]). In the second case, the

[1] Corresponding author : Louis Rivest, professor, ÉTS, 1100 rue Notre-Dame ouest, Montréal, Québec, Canada, H3C 1K3; louis.rivest@etsmtl.ca

evolution uses predefined processes and forms (Engineering Change Request, Engineering Change Order, etc.), and is called Modification (Maurino [1]).

The study of administrative mechanisms and processes concerning the use, and, eventually, the computerization of these processes, is related to Change Management [2, 3]. The study of the models, tools and methods through which the effect of an evolution can be evaluated, and eventually documented for each effected object, falls under the domain of Change Propagation. This article proposes a classification of associations in order to organize change propagation within CAD and PLM systems.

1. Current Systems

Engineering projects rely on a set of specialized software. CAD systems usually handle the product's design and development (authoring) aspects. PLM systems (Product Lifecycle Management) handle the management of the information generated by the development process.

1.1. CAD Systems

CAD systems manage the lowest-level associations, i.e. between geometrical entities or elementary numerical parameters. Extensive mechanical design projects, a trademark of the automotive and aerospace industries, generate hundreds of thousands of CAD files and, potentially, millions of associations or dependencies. A rigorous knowledge of the dependencies between the geometrical elements should allow for efficient change management. However, this is not the case. CAD systems overestimate the modified elements when a change occurs. Hence, automatic update functions of the associations between CAD files cannot be used in an industrial context because they imply the creation of new versions of too many files, often needlessly.

1.2. PLM Systems

PLM systems manage associations between files, based on information external to the CAD files that is mainly administrative in nature, such as the date of the last modification, the modification author, the file's maturity status (WIP, Approval pending, Released, etc.). Actually, PLM systems manage files as black boxes. In this context, they cannot possibly circumvent a modification impact to only the affected entities, and hence to only the affected files affected by the change. A modification in a file leads the PLM system, by "domino effect", to deduce that each file may be affected by this change. For example, a modification of a feature in a component changes the component, which in turn changes the component that uses it, and so on.

The associations managed by the PLM system are fewer than those managed by the CAD system. However, due to their very weak semantic content, their exploitation potential is greatly limited in regards to change propagation.

1.3. Interoperability Between PLM and CAD Systems

The CAD system is the content editor whereas the PLM system is the content manager. Even when the CAD and PLM systems are from the same software editor, their interoperability is extremely weak. These computer applications do not provide an efficient solution to change propagation, be they corrections or modifications.

1.4. Objective of This Paper

Efficient change propagation is a fundamental business issue that relates to delays, quality and competitiveness. Understanding the roles of associations between geometrical elements, features, parts, products and processes is a key to solving this issue, and to reaping the benefits of modern CAD and PLM systems. The objective of this paper, which presents some of the work conducted by our research team [4], is to propose a common vocabulary, so as to characterize and analyze the role of associations in product development. Taxonomy of associations is proposed based on the knowledge they handle. From there, we propose the RLC model that coordinates the associations manipulated in CAD and in PLM systems, which would enable, in the long term, to better integrate CAD and PLM as well as to efficiently manage change propagation.

2. Review of Literature

Numerous papers deal with the role of relations during product development. These papers use quite a heterogeneous vocabulary, and have not yet reached a consensus.

In the context of an assembly, many authors [5, 6, 7] use the term *mating relation* to describe a dependency between two components within an assembly. The terms *against* and *fit constraint* are used to define the dependency type. An *against constraint* establishes a contact condition between two planar faces [8], while a *fit constraint* establishes a contact condition between a shaft and a hole [8]. Shih and Anderson [9] define the term *geometrical constraint* as a limitation of the value of an element's attribute in regards to itself or to any other element of the design, such as the position (coincidence, concentricity), the orientation (parallel, angle), etc.

Eustache [10] introduces four types of *constraints*: the dependency constraint, the relational constraint (<,=,>,≠,≤,≥), the boolean constraint (logical operations : AND, OR, etc.) and the rule constraint. He considers the dependency constraint to be a dependency association between two elements. The relational and boolean constraints are related to mathematical functions and logical operators. Finally, the rule constraint is a trade-rule applied to an entity. This constraint is similar to the derivation link proposed by Giguère et al. [11, 12]. They both try to automate trade-oriented tasks. To reach this objective, Giguère et al. define the technological link as a link expressing a dependency between two information elements. He uses a derivation link, which encapsulates some knowledge, to automatically obtain a pseudo-imprint of a feature from the imprint of another feature, in order to automate a design task.

Similarly, Michaud establishes a persistent technological link between the model of an aerospace skin panel and its associated tooling used for the chemical milling. He demonstrates that a proper preparation of the models with the help of a trade-oriented methodology, typified entities and technological links (expertise carriers), allows for considerable time gain during change propagation, starting from the part and all the way to its associated tooling [13].

According to Shah and Mantyla [14], multiple constraints (form, dimension, orientation and position) are required to describe the relation between two features or parts in the same assembly. For some authors [5, 14] a *link* aggregates many contact constraints in one entity, so as to increase the level of model abstraction. Mukherjee and Liu [15] discuss a *quasi-link*, the main objective of which is to describe geometrical relations (concentricity, perpendicularity, etc…) between two features. Their link definition is similar to the *constraint* provided by Shih and Anderson [9].

Zimmermann et al. [16] describe the logical link between two engineering objects (EO). They also describe the knowledge that allows the automatic instantiation of EO and EOR (engineering object relation) from generic EO and EOR. Laako and Mantyla [17] use the term *relation instance* to describe the link between an abstract object and each of its instances. Finally, Yassine et al. [18], in their discussion of the links between heterogeneous elements, use the term *relation*. They consider that a relation indicates a dependency between two elements without necessarily describing the dependency type.

Analyzing the literature, we observe that there is a need to establish a common terminology in order to organize various concepts in a general model. This is what we propose in the next section.

3. Characterizing and Organizing Associations for CAD and PLM

In this article, we propose to characterize and organize associations and the objects to which they are applied. Essentially, an association between objects indicates the existence of a dependency between them. An object, again, as defined by Maurino [1] is labeled a technical object (TO) and impartially indicates every useful identifiable element of a product, such as a part, an assembly, a function, etc. In this paper, a geometrical entity, a feature, a document, are also seen as technical objects, and a TO can represent a set of TOs, in a recursive manner.

3.1. Types of Associations

The PLM and CAD systems manage associations of very different natures. We have defined three types of associations that aim for an increased interoperability between PLM and CAD. We use the generic term Association to refer to any type of dependency regardless of its specificities.

Relation

A relation establishes an abstract dependency between TOs. A relation is not formalized enough for it to be used by a software tool to propagate a change. Therefore, its interpretation and use requires human intervention. The ‘representation link’ discussed by Maurino [1], associates a TO (such as a part node in a PLM structure) to a document (such as a CAD file) that describes it. A relation is generally managed by the PLM system. For example, a relation could contain the positioning of a part in relation to another part, expressed by a positioning matrix resulting from the design intent at some point in time. If one part changes, the relation itself does not suffice to propagate automatically the change and maintain design intent and model consistency.

Link

A link defines a formal dependency between TOs. It describes a specific know-how associated to a given task using a procedural language or an approach similar to artificial intelligence. A task is considered here as a small portion of a work to be performed (defining a feature in a part can be considered as a task in a design work). It can be implemented by software means and may require occasional human intervention. The derivation link described by Giguère et al. [11, 12], that creates a joggle at the intersection between a stringer and a frame is an example of a link that captures a specific know-how based on methodologies unique to the aerospace field. A link can be built inside a CAD system (using Knowledgeware or User Defined Features within CATIA V5) or outside by using the A.P.I. (Application Programming Interface). Thus, a link could, for example, correspond to a pin joint association between features (hole, shaft, etc.) of mechanical parts.

Constraint

A constraint represents a specific formal and indecomposable dependency between TOs. Constraints are handled by the application (in our case, a CAD system) that also manages the dependent TOs. The application makes them available, generally, via the A.P.I., a macro-language, or the user interface. They are, in a way, terminal level associations. A parallelism constraint between two lines is a typical example of this type of association. Consequently, the constraints are indecomposable dependencies used to create links. Thus, the pin joint (link) mentioned above is decomposed into coaxiality and planar contacts constraints between the geometrical elements of the two parts.

3.2. Some Association Properties

A set of properties has been defined to characterize associations. While some are conventional, others are more specific to the field of application at hand.

Cardinality

We adopted the accepted definition of the cardinality of an association, i.e., the number of objects used in the dependency. To be complete and flexible, the cardinality of the association m X n is supported by our model. The particular cases of injection (m X 1), surjection (1 X n), and bijection (1 X 1) are considered. This attribute is linked to the concept of granularity presented hereafter, as well as to the previously mentioned recursive decomposition of TOs. Thus, generally, an association relates two sets of TOs (m X n).

Direction

Direction describes the orientation of the dependency or the orientation of the information flow between associated TOs. The RLC model distinguishes four cases:

1. **Directional association.** An association is directional when the dependency allows for the respective identification of the set of target and reference (or destination and source) TOs, or, generally speaking, when the roles of two sets of associated TOs are different, and cannot be exchanged. For example, in the case of two lines associated by a parallelism constraint, if one line is identified

as a datum, it becomes a reference, and only the orientation of that specific line can be modified and propagated to the other line.

2. **Bidirectional association.** An association is bidirectional when the flow of information can circulate both ways through the association. This is the case when the target and reference TOs cannot be differentiated, i.e., when they have identical roles, and when modification of the state of one will impact the other. An example could be similar to the above, i.e., two lines associated by a parallelism constraint, except that no datum is identified. An orientation change of one of the lines impacts the second one. An association is also said to be bidirectional when the reference and the target objects play distinct roles that can be exchanged in the association, which implies that the semantics of the association can be inversed. For example, a diameter A associated to a diameter B, related by a ratio of 2 (B is twice the size of A). The inverse is also true: A is half the size of B. Conceptually, a bidirectional association is equivalent to a pair of inversed directional associations.
3. **Association without direction.** An association is without direction when it does not convey a flow of information between the associated TOs. This implies that the state of an object cannot be determined by the state of the associated object. For example, a sub-assembly A is associated to a sub-assembly B to compose a product. The association between both sub-assemblies is without direction. These associations are generally relations.
4. **Association with an undetermined direction.** The direction of an association is unknown as long as it has not been determined as belonging to one of the previous cases. This situation arises when an association used during the product development process has been identified but not characterized.

Temporality

Regarding time, an association is either persistent, transient or semi-persistent. According to Michaud [13], the massive use of persistent associations in V5 is a fundamental differences between CATIA V4 and CATIA V5.

1. **Persistent association.** Only the destruction or modification of the association brings an end to the dependency expressed between TOs. Persistent associations facilitate change propagation. In Michaud's work [13], persistent links, established between TOs of the part and of its tooling, support the automatic change propagation by capturing the design intent.
2. **Transient association.** A transient association only exists during the creation of the target TOs. Only the result is saved, not the acquiring mechanism. Such an association has, a posteriori, no usefulness when a change occurs.
3. **Semi-persistent association.** A semi-persistent association has limited effectivity in order to fulfill specific needs. The control of this effectivity applied to a composition relation (in a product structure) allows, for example, for a component to be informed that it will be used in the assemblies of a given series.

Aggregation and Decomposition

Aggregation is the grouping of a set of associations or TOs in one entity in order to facilitate its manipulation at a higher level of abstraction. Decomposition is the contrary operation. Aggregation and decomposition are recursive operations.

In the context of product development, we consider that relations are aggregated links, whereas links are aggregated constraints. Similarly, files (parts) are aggregated features, whereas features are aggregated elements. In practice, relations associate files, links associate features, and constraints associate geometrical elements.

The concepts of aggregation and decomposition, illustrated in figure 1, help bring a continuity (a transition) between the numerous constraints established between low-level TOs, generally managed by the CAD system, and the fewer relations between high-level TOs (files), managed by the PLM system. Indeed, in the case of current, large-scale aerospace and automotive projects, a current PLM system could not possibly handle the entire CAD system's constraints if they were to be exposed. Moreover, the mere knowledge of relations between files gives only minimal assistance in change management and is insufficient to consider any kind of automatic processing.

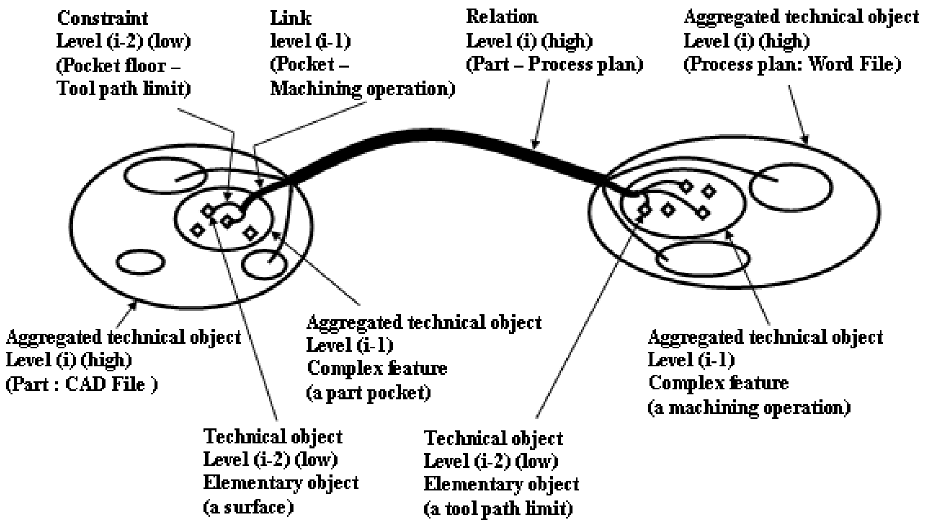

Figure 1. Example of aggregation/decomposition of associations and of technical objects

The RLC model gives designers a means to express design intents by handling links that encapsulate a trade-oriented know-how between TOs having a certain level of abstraction, that would afterwards be decomposed by the CAD system into constraints between elementary TOs. Current CAD systems allow designers, generally, to handle low-level constraints that only partially capture the design intent. Figure 1 illustrates an aggregation and decomposition example.

4. Conclusion

Current CAD systems primarily manage constraints established between elementary technical objects, to which links between more complex objects can be added. These objects are aggregations of geometrical elements, when UDF, Knowledge or VB are used. Current PLM systems used in industry primarily manage relations between files.

These various types of associations are characterized by the know-how they formalize or define: a constraint carries a highly formalized, fragmentary know-how, whereas a relation conveys an abstract dependency. A link defines a formal dependency to which is associated a particular know-how that is specific to a given task. These three types of associations are characterized by the properties of cardinality, direction and temporality. Aggregation and decomposition make it possible to navigate between relations, links and constraints. In this context, the RLC model (Relation-Link-Constraint) positions the link at a satisfactory level for human interaction with the CAD tool, and establishes the link as an adequate way to achieve CAD-PLM integration, and allows for the eventual efficient change propagation during product development.

References

[1] Maurino, M. (1993). *La gestion des données technique : technologie du concurrent engineering*. Paris: Masson.

[2] Bouikni, N., Desrochers, A., & Rivest, L. (2006), *A Product Feature Evolution Validation Model for Engineering Change Management*, JCISE, V(6), N(2).

[3] Chen, S.-J., & Lin, L. (2002). *A Project Task Coordination Model for Team Organization in Concurrent Engineering,* Concurrent Engineering: Research and Application, 10(3), pp. 187-202

[4] Tremblay, T.G. (2006). *La propagation du changement dans un contexte PLM : vers la maîtrise du rôle des associations dans le développement de produits et des processus associés*, Mémoire de maîtrise, École de technologie supérieure, Montréal, Canada.

[5] Lee, K., & Gossard, D. C. (1985). *Hierarchical Data Structure For Representing Assemblies: Part 1.* Computer Aided Design, 17(1), 15-19.

[6] Salomons, O. W., van Slooten, F., de Koning, G. W. F., van Houten, F. J. A. M., & Kals, H. J. J. (1994). *Conceptual graphs in CAD*. CIRP Annals, 43(1), 125-128.

[7] Tan, J., & Zhang, Y. (2000). *Self-organizing assembly modeling based on relational constraints.* Chinese J. of Mechanical Engineering (English Edition), 13(2), 145-152.

[8] Lee, K., & Andrews, G. (1985). *Inference of the Positions of components in an assembly: Part 2.* Computer Aided Design, 17(1), 20-24.

[9] Shih, C.-H., & Anderson, B. (1997). *Design/constraint model to capture design intent*. Proceedings of the 1997 4th Symposium on Solid Modeling and Applications, May 14-16 1997, Atlanta, GA, USA.

[10] Eustache, J. (2002). *Contribution à une meilleure gestion de la cohérence des données du produit au cours de son cycle de vie*. Thèse, IFTS, Université de Reims-Champagne-Ardenne, France.

[11] Giguère, F., Rivest, L., Desrochers, A., & Maranzana, R. (2001). *Les caractéristiques contextuelles : une solution pour accroître la productivité en CAO*. Revue Internationale de CFAO et d'Informatique Graphique*, 17(1-2), 105-117.*

[12] F.Giguere, L.Rivest and A.Desrochers. (2002). *Improving design productivity and product data consistency*, Feature Based Product Life-Cycle Modelling, René Soenen and Gustav J. Olling, ed., Hardbound, Kluwer Academic Publisher, pp.77-91.

[13] Michaud, M. (2004). *Méthodologie de modélisation unifiée pièce-outillage en CAO aéronautique : application aux tôles et gabarits de découpe*. Mémoire de maîtrise, École de technologie supérieure, Montréal, Canada.

[14] Shah, J. J., & Mäntylä, M. (1995). *Parametric and feature-based CAD/CAM: concepts, techniques, and applications*. New York, N.Y., J. Wiley and Sons.

[15] Mukherjee, A., & Liu, C. R. (1995). *Representation of function-form relationship for the conceptual design of stamped metal parts.* Research in Engineering Design - Theory, Applications, and Concurrent Engineering, 7(4), 253-269.

[16] Zimmermann, J. U., Haasis, S., & Van Houten, F. J. A. M. (2002). *ULEO - Universal linking of engineering objects*. CIRP Annals - Manufacturing Technology, 51(1), 99-102.

[17] Laakko, T., & Mantyla, M. (1996). *Incremental constraint modelling in a feature modelling system*. Computer Graphics Forum Proceedings of the 1996 17th Annual Conference and Exhibition of the European Association for Computer Graphics, EUROGRAPHICS'96, Aug. 26-30 1996, 15(3), 367-376.

[18] Yassine, A., Whitney, D., Daleiden, S., & Lavine, J. (2003). *Connectivity maps: modeling and analysing relationships in product development processes*. J. of Engineering Design, 14(3), 377-394.