INTERNATIONAL DESIGN CONFERENCE - DESIGN 2004
Dubrovnik, May 18 - 21, 2004.

DESIGN 2004

"SYMBOLIC DESIGN": A UML BASED ENVIRONMENT FOR INTEGRATED PRODUCT DEVELOPMENT

A. Liverani, G. Amati and M. Pellicciari

Keywords: CAD, design, UML

1. Introduction

While in the very near past the primary product development tools developers' focus was on solving product complexity, now the focus is on identifying tools and methods capable to manage and optimize the resources involved, planning, monitoring and simulating different scenarios, with the main target to integrate all the results and activities. Nowadays, state-of-the-art PLM (Product Lifecycle Management) software packages are approaching the problem expanding PDM (Product Data Management) environments into multi-purpose and multi-resources databases. At a deeper analysis, such approach seems to guarantee only a conversion from the paper documents into fully digital data of easier management. But, on the other side, in this way it is not possible to identify a real step ahead into the design and product development resources integration and control. In fact databases have fixed structures and needs to be filled in order to provide a real benefit to the project. The product development needs a much more flexible tool than multi-resource databases.
Moreover the design activity is mainly subdivided in two different stages: conceptual design and product development [Pahl 1988]. These two steps may be basically identified by sentences like: (a) from idea (or need) to a feasible solution; (b) from a feasible solution up to a commercial product (through a concurrent and recursive refinement and optimization). Several authors prefer to define the second stage as PPD (Process and Product Development), including also the resource planning and production setup. In fact CAD/CAx systems are everyday expanding their focusing area, including FE (Finite Elements) analysis software, PDM and several other simulation packages. A more global vision is given by PLM: CAD is part of a complex and heterogeneous environment, where Resource Planning modules are also cohabiting with powerful production flow graph makers and databases. Additionalluy several research activities have highlighted the relevance and benefits in a project of a very early functional decomposition [Pavkovic 2000], targeted to setup an adequate modularity and design strategy. An helpful approach is given by IDEF schema special designed for Concurrent Engineering purpose. A commonly used diagram symbolism, which has been introduced and nowadays widely applied in most enterprises, is the QFD (Quality Functional Deployment) visualization. This is a quality-oriented diagramming system that, like all other described above, rarely exchanges data with product geometric description and parameter-driven dimensions.

1.1 The "Symbolic Design" paradigma

The very first project step is somewhat we can call Conceptual Design: the work developed in this stage is deeply different from the second part and includes the analysis of several conceptual solutions and basic calculations, strategically more important than the later accurate geometry improvement and optimization. Forces, masses, velocities, energy are parameters involved in this stage. All the calculations are performed on coarse models since it is not necessary detailed geometry, but general dimensions about space occupancy, axis, commercial component documentation, macroscopic costs … time, human resources, detailed costs and geometric description are included in the PPD stage. While tools like PLM may be greatly helpful in that area and visual charting packages, like Visio, make the Systemic Design easier, a big lack of software tools can be found in the Conceptual Design. The proposal described in this paper is aimed to overcome this deficiency with an original conceived software interface, giving to the designer a smart tool for a project setup: this is the Symbolic Design (SD). The key-points that the SD approach took as main driving directions from the beginning have been the followings:

1. managing *less geometry* and more engineering knowledge;
2. using *labels and standard strings* instead of full 3D standard parts;
3. using a *parametric approach* not only in the geometry development, but also in the conceptual stage;
4. using easy and well-known software tools, like *spreadsheets* or similar.

During the very first project steps the designers need as quick as possible to evaluate several different ideas, arising from commercial inputs or research activities. A detailed geometry is considered a waste of time and many designers prefer to use self-made spreadsheets, technical manuals and commercial datasheets to build a variable driven (parametric) model. The flexibility of a such approach is total: a complex machinery, for example, is considered in its functional parts, where motors, joints, bars, etc… can be mainly collected by catalogues and simply placed in the spreadsheet with their commercial codes. Only special components are truly detailed and drafted for the production. Variables, like power, masses, velocities, general dimensions and also costs, extracted from external component datasheets, are really involved in the preliminary stage.
A full project implementation in a CAD system is expensive in terms of time and human resources and this activity begins essential only when the designer team feels the design solution is feasible. That means that a big amount of design solutions aren't deeply evaluated due to the lack of resources to spend in project development. Furthermore, it is a common thought that CAD-based parameterization is crucial when detailed geometry is modified and optimized, but absolutely useless when the project changes functionally or conceptually (for example, due to a change in a physical functioning principle). Following this direction, a function based and less geometry oriented software tool for preliminary design strongly empowers the engineering knowledge storage and the design evolution display. Sometimes the pure 3D models or 2D drafts are not showing immediately the design choices taken during the conceptual and development stages. A simple and intuitive graphical interface may be greatly helpful to overcome this drawback, by showing functional data and string based (label) important information. The benefits also in the management of conceptually different project approach are evident due to the dramatically reduction of resources dedicated the a detailed solution, but a enforced attention dedicated to functions.
On the other hand several software tools appeared on the market to aid the design team in the project management and resource planning. Several other good commercial packages help the designers in the diagramming all project phases, but are considerably too far from physical parameters to be integrated in a CAD environment. Finally a system with this kind of features allows to overcome the poor interaction between industrial company departments, increasing standardization of the entire product lifecycle steps and decreasing production costs. The main effort is then building a shared co-design interface among project teams, which enable a real-time interaction overcoming the poor communication that is a bottleneck in an industrial manufacturing process context.

2. Functional UML: UML for functional diagramming and entities constraining

One of the most important advances in the diagramming theory for Object-Oriented Programming (OOP) software development is commonly considered the UML (Unified Modelling Language) [Booch 1999]. It consists in a standardized diagramming tool for complex software code design. The development of UML tools and interfaces led to a software environment capable to automatically generate source code infrastructures. An aspect that couldn't be set aside in the development of an interface for the SD was the need of a formal standardization for diagramming. UML appeared to be the most complete and modern and has been adopted for SF-CAD. But the considerably big gap in the applications between software layout and Symbolic Design diagramming has been overcome by defining the F-UML (Functional UML). In other terms the UML has been transposed into an engineering field, totally maintaining its semantic features. Additionally a 3D interface has been implemented in order to add also functional relationships like tri-dimensional axis aligned or plane face mating. In the rest of this chapter an outline and compact description of F-UML will be provided through easy examples.

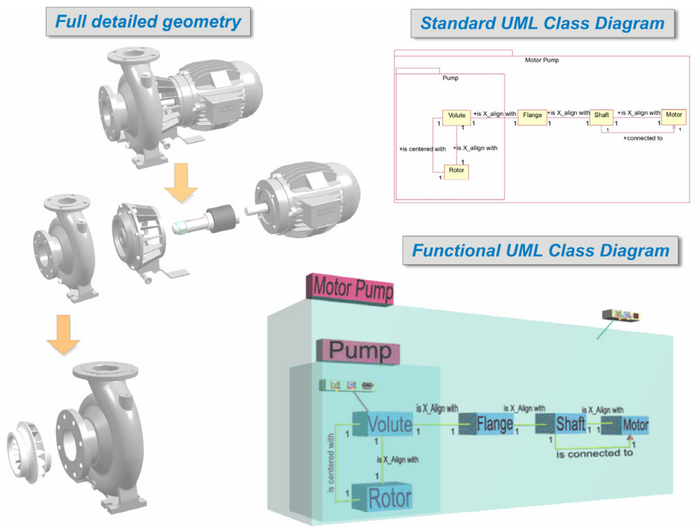


Figure 1. Detailed geometry and UML objects (classes) in standard and functional mode

In Figure 1 the same mechanical assembly has been presented in 3 different faces: the full detailed geometry, the standard UML Class Diagram and the Functional UML Class Diagram. 3D boxes represent classes of objects that are related each others between solids lines (*generic association*). A bi-directional association connects two classes within a role defined by the label placed near the boxes. The association can also have different adornments that let to better characterize the role between classes. Each UML entities can has a note () symbol connected represented by the solid box with a cut corner, that specify a remark for that entity. The dashed line is used to link UML entities with their notes objects. A multiplicity string specifies the range of allowable cardinalities that a set may assume.

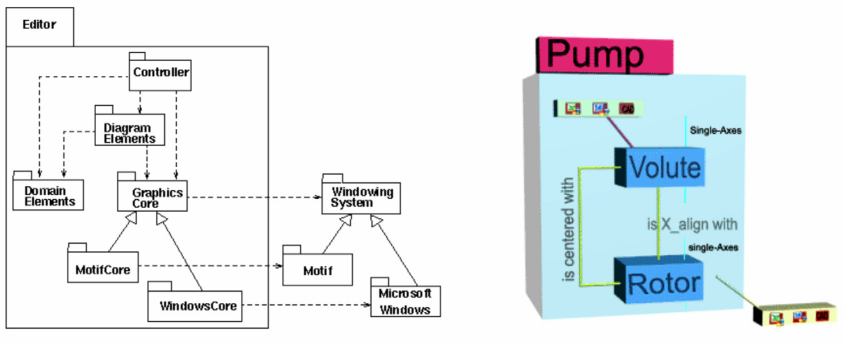


Figure 2. Packages and dependencies

A set of several classes can form a Package. Each class represent a sub-operation that the entire package is able to give. A package (see Figure 2) is bound to a class by a dependency role (dash line). A dependency indicates a semantic relationship between two (or more) model elements. Static and dynamic states of a software system are the two main aspects which globally characterize a software: the same may be considered also in an industrial product. From the static point of view, *Class Diagram* (Figure 1 and 2) lets to define classes of objects with their attributes, inheritance relations with other classes. In terms of design, classes are entities, components or sub-assemblies that can interact with others. Furthermore the UML enables to describe different kind of associations constraining classes and then objects to a particular behaviour with each others. In the UML class diagram, each class entity has an arbitrary number of parameter and methods that defines its behaviour. As the unified language was developed specifically to build C++ software framework, each parameter has a visibility scope attribute which regulate data access from others classes. Other different diagrams like *Use Case Diagram*, *State Diagram* and *Interaction Diagram* let to represent the dynamic behaviour of the entities involved in the system activity.

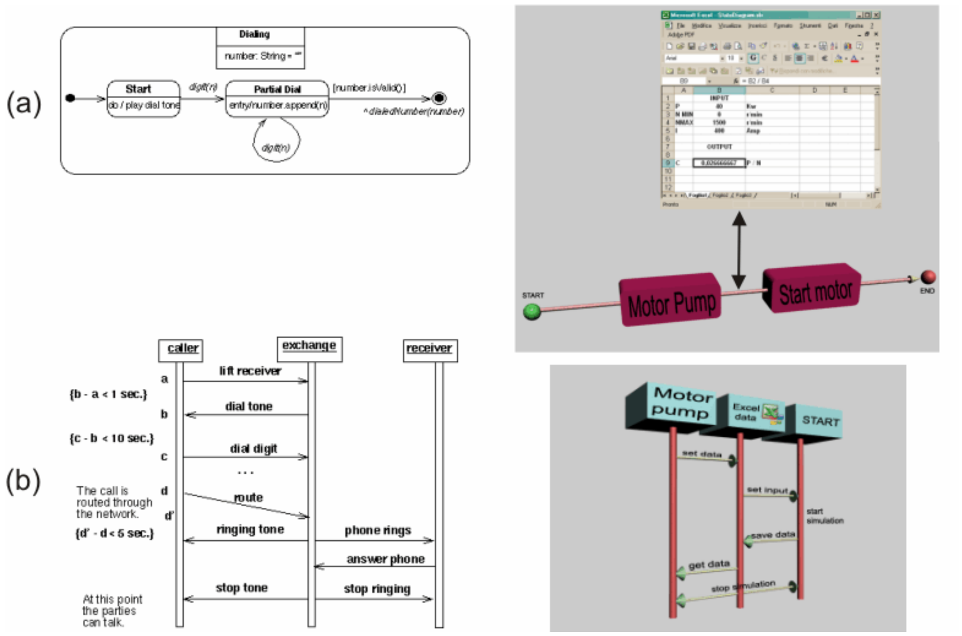
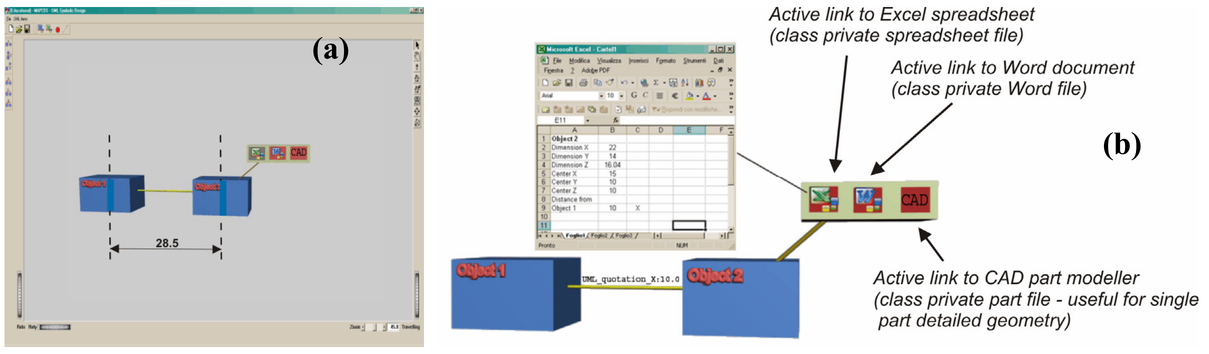


Figure 3. Sequencial State Diagram (a) and Interaction Diagram with concurrent objects (b)

The *Use Case Diagram* shows the relationship among actors and use cases within a system is a graph of actors. It describes a set of use cases enclosed by a system boundary, communication (participation), associations between the actors (stick man) and the use cases, and generalizations among the use cases. The *Sequencial State Diagram* shows the sequences of states that an object or an interaction goes through during its life in response to received stimuli, together with its responses and actions. A state diagram is a bipartite graph of states and transitions. It is also a graph of states connected by physical containment and tiling. The entire state diagram is attached (through the model) to a class or a method (an operation implementation). A pattern of interaction among objects is shown on an interaction diagram. Interaction diagrams come in two forms based on the same underlying information but each emphasizing a particular aspect of it: sequence diagrams and collaboration diagrams. The *Interaction Diagram* shows an interaction arranged in time sequence. In particular, it shows the objects participating in the interaction by their "lifelines" and the messages that they exchanged arranged in time sequence.
Using F-UML graphics, designers can build functional schemas with respect to a product or part of it, instantiating several UML objects and defining relations and constrains between them in the same manner as computer programmers build software. This approach is completely affine to an Object-Oriented Programming reasoning, but the final product is a SF-CAD functional report that describes the whole functionality that the final product will own, instead of a C++ code. Further more, our Object-Oriented Functional Design (OOFD) approach let user to specify characteristics of single object independently from their final CAD model, but closely connected.

3. The SF-CAD Architecture

The software we developed is a pioneering proposal of Symbolic Design tool that enable designers to begin a project basically describing objects by their functional relations. Referring to the UML *Class Diagram*, each entity represents a class of objects with particular parameters that can describe different features. These parameters can regard various properties of the objects, for example its dimensions (design parameters), its position in a 3D space (along single-axis, double-axis or three-axis alignment) (functional paramenters). The UML paradigm reveals its power in describing entities, and their functional properties, which will be part of the final product. In this manner, desingers have a complete report about all models, together with their functional characteristics, assembly sequence order, geometric dimensions, and many others information.
But SF-CAD also marks a step ahead in the integration between diagramming software tools and real geometric entities by directly establishing basic constraints inside the environment. The shift of component relative positioning in 3D and their coarse constraining from PPD up to the conceptual stage, makes the designer able to think at a more real object and facilitates the startup of the fine tuning of the geometry. So the interface is completely tri-dimensional (Figure 4 (a)) with common zooming and 3D viewing commands (Open Inventor based interface). This setup is due also to a correct design procedure: a top-down approach. In fact, most commercial CAD systems favour the buttom-up procedure, because the first step in the design consists in the single component modelling and only after several part completed, the assembly comes up. Thus this vision seems to be more global and correct, following a path from "macro" to "micro".


(a)
28.5
CAD
(b)
Active link to Excel spreadsheet
(class private spreadsheet file)
Active link to Word document
(class private Word file)
Active link to CAD part modeller
(class private part file - useful for single part detailed geometry)
Microsoft Excel - Cartel1
File Modifica Visualizza Inserisci Formato Strumenti Dati Finestra ? Adobe PDF
E11
Foglio1 Foglio2 Foglio3
Pronto
NUM
Object 1
Object 2
UML_quotation_X:10.0

	A	B	C	D	E	F
1	Object 2					
2	Dimension X	22				
3	Dimension Y	14				
4	Dimension Z	16.04				
5	Center X	15				
6	Center Y	10				
7	Center Z	10				
8	Distance from					
9	Object 1	10	X			
10						
11						

Figure 4. SF-CAD GUI (a) and Interaction Diagram with concurrent objects (b)

Every entity instanced is labelled, either to improve the visual effectiveness, either to avoid the time and money waste in component resume and positioning in the assembly: a clear string over the block in the Class Diagram with commercial component code and references replaces efficiently the use of 3D part standard, allowing the operator to focus on design functions and parameters.
In association with a class instance, the software automatically prepare three active links with office automation applications and a CAD part modeller (Figure 4 (b)).
The aim of these links is to give the operator an handy and simple access to all type of parameters: geometric, physical and economical and so on. Parameters are stored in a object's private spreadsheet file, which is kept

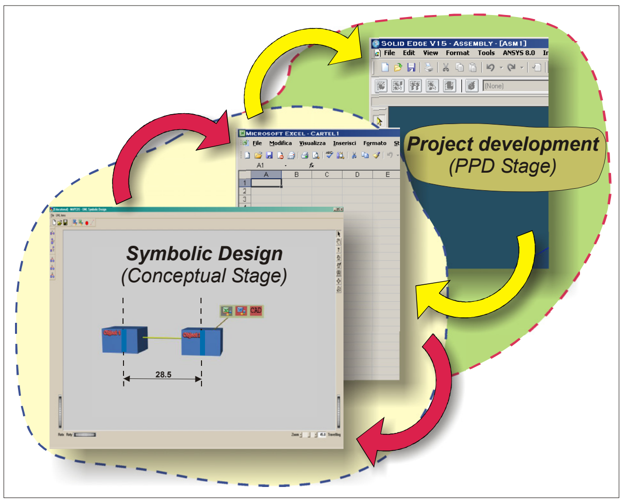


Figure 5. SF-CAD integration with a commercial CAD (Solid Edge)

syncronized with the MPD (Master Parameter Database), described in the next chapter. At any time the operator can modify these parameters, setup formulas and perform calculations.

The link to commercial CAD environment

The functional entities diagramming by UML and all active links instanced within the 3D SF-CAD interface do need a common data exchange to a commercial CAD system in order to facilitate the start of the development stage and, at the same time, the intrinsic parametrization established in the functional decomposition activity. SF-CAD provides also a basic mechanism to automatically collect data from class private spreadsheet files into a common storage: that is another Excel file, we call MPD (see Figure 5). During the Project Development stage, the operator starts an assembly (for example in Solid Edge Assembly) and may link the MPD parameters as external variables in order to setup a full parameter-driven and variational geometry. In this way the SF-CAD keeps an independent interface, but ties up a close connection to any commercial CAD, able to read variables from external database.

Finally the proposed environment demonstrates clearly its attitude to capture, store and visualize in a simple and intuitive way the design knowledge, allowing also smart changes to very different configurations.

4. Results and conclusions

In this work we present an innovative 3D interface for Symbolic Design. The UML language was employed to design and fix all constrains that regulate all interaction between product entities. UML formalism allow to define classes of objects which represents levels of abstractions for all different steps in PLM. Furthermore UML allow us to project an innovative connection between the functional and resource optimization aspects expanding a traditional Process and Product Development process in order to integrate project control flow with conceptual design. Results returned by first SF-CAD use are very impressive and push to further development and refinements.

References

Booch, G., Rumbaugh, J., Jacobson, I., "Unified Modelling Language User Guide", Addison Wesley, 1999.

Chen, R., Sgroi, M., Lavagno, L., Martin, G., Sangiovanni-Vincentelli, A., Rabaey, J., "Embedded Systems Design Using UML and Platforms", System Specification and Design Languages (Forum Design Languages 2002), CHDL Series, Kluwer, 2003.

Dwyer, T., "Three dimensional UML using force directed layout", Technical Report TR2001/25, Department of Computer Science, The University of Melbourne, Parkville, Australia, 2001.

Fernandes, J. M., Machado, J. R., Santos, H. D., "Modeling Industrial Embedded Systems with UML", 8th ACM/IEEE/IFIP International Workshop on Hardware/Software Codesign (CODES 2000), San Diego, CA, USA, ACM Press, (ISBN 1-58113-214-x), pp. 18-22, May/2000.

Flemming, U., Erhan, H., Ozkaya, I., "Object-Oriented Application Development in CAD. A Graduate Course", Proceedings of ACADIA 2002 Conference., G. Proctor, ed., pp. 27-38, 2002.

Machado, J. R., Fernandes, J. M., Santos, H. D., "An Object-Oriented Approach to the Co-Design of Industrial Control-Based Information Systems. In 4th APCA Portuguese Conference on Automatic Control (CONTROLO 2000), Guimarães, Portugal. (ISBN 972-98603-0-0), 2000.

Pahl, G., Beitz, W., "Engineering Design: A systematic approach", Springer, 1988.

Pavkovic, N., Marianovic, D., "Entities in the Object-oriented design process Model", Proceedings of International Design Conferences – Design 2000, Dubrovnik, May 23-26, 2000.

Alfredo Liverani, Prof. Ing.
University of Bologna, DIEM
v.le Risorgimento, 2, Bologna, Italy
Telephone: +39 051 2093452, Telefax: +39 051 2093412
E-mail: alfredo.liverani@unibo.it