C. Cauvet, D. Rieu, B. Espinasse, J.P. Giraudin, M. Tollenaere (1998), « Ingénierie des systèmes d'information produit : une approche méthodologique centrée réutilisation de patrons », INFORSID 98, Montpellier, 13-15 mai, 1998.

Product Information Systems Engineering: a methodological approach focused on re-use of patterns

Corine CAUVET[1] - Dominique RIEU[2] - Bernard ESPINASSE[1] - Jean-Pierre GIRAUDIN[2] - Michel TOLLENAERE[3]

1 - DIAM-IUSPIM, Université d'Aix-Marseille, Domaine universitaire de Saint Jérôme, 13397 Marseille Cedex 20 FRANCE - E-mail: prenom.nom@iuspim.u-3mrs.fr
2 - LSR-IMAG, LSR-IMAG, BP72, 38402 Saint Martin d'Heres Cedex FRANCE - E-mail: prenom.nom@imag.fr
3 - GILCO-ENSGI, 46 avenue Félix Viallet, F-38031 Grenoble, Cedex FRANCE - E-mail: prenom.nom@ensgi.inpg.fr

Abstract: In industrial organisations, product information systems (PIS) support technical data management of products and their design/production processes. The current economic context makes these PIS strategic for these organisations. Currently, these organisations have many difficulties in developing these systems. The purpose of this research is to define a methodological framework adapted to PIS engineering and allowing these difficulties to be reduced. In this paper, first, we identify these difficulties and show the need of a re-use approach for a "deviation" development of these systems. Second, we distinguish different forms of re-use already introduced in software engineering, with particular emphasis on design pattern re-use. The proposed methodological framework is mainly based on the re-use of patterns throughout the PIS development process. These patterns are used to define reoccurring development problems in these systems and to allow re-use of associated solutions already defined. The paper illustrates this approach by defining and specifying a pattern adapted to a mechanical design problem.

Key words: Design Patterns, Business Patterns, Product Information Systems (PIS), Technical Data Management Systems, Re-use.

Category: Researcher, Industrial Application

1. Introduction

Internationalisation of economy in the 1990s calls for a drastic reduction in the market release lead times of new products together with an enhancement in the quality of such products. The resulting complexity of industrial organisation is therefore leading companies to adopt new practices such as simultaneous engineering or new structures such as the distributed enterprise or the virtual enterprise. [VER 94, TER 92]. In this context, product information systems (PIS) allowing, first, effective management of the documentary database accompanying product development and, second, rationalisation of all the development process tasks and their breakdown between the various entities, have become a strategic stake for industrial companies. The reactivity of such companies to the changes involved in the product development process requires controlled circulation of technical documentation, rigorous management of its upgrading, together with the necessary traceability for quality control: all these functions must be performed by the PIS.

Considerable research work and, in particular, a number of European projects have already expressed their interest in sharing and integration of technical data in industrial companies. Consequently, associated with the AIT initiative [AIT 97] (Advanced Information Technology Design and Manufacturing), we can quote in the first place the RISESTEP project [RIS 97] interested in the implementation and validation of STEP shared databases for simultaneous engineering, followed by the PISA project [PIS 94] whose aim is to develop a platform for information sharing, including a methodology and tools for information modelling and for the integration and validation of models. Finally let us quote the OPAL project (Integrated Information and Process Management in Manufacturing Engineering) [OPA 97] whose aim is to provide concepts for high level integration of processes and information in the field of engineering and production. These projects are particularly concerned with the problems of integration and sharing technical data, problems that are extremely sensitive in PIS development. However, they do not fully deal with the actual engineering of the PIS, i.e. the methodological aspects relating to the design and production of such systems. Since PIS development is strategic for industrial companies, the stakes associated with these methodological aspects are therefore strategic also. Our research, focusing on these aspects, therefore consists of defining a methodological framework for the development of Product Information Systems (PIS), associating computer (DIAM-IUSPIM, LSR-IMAG) industrial engineering (GILCO-ENSGI) and sociology (CRISTO-CNRS) university laboratories and a major industrial company (Groupe SCHNEIDER).

The purpose of a PIS is to manage information concerning products and product families throughout their life cycle (from design through to production). The current economic competition context tends to reduce this life cycle more and more, meaning that industrial companies are obliged to develop numerous PIS as quickly as possible, which will remain operational only during the lifetime of the product(s) concerned. Consequently, PIS-adapted engineering must enable "deviation" development of such systems, i.e. make it possible to design and produce new PIS from PIS already designed and produced, thus allowing re-use of existing software components and specification elements. The methodological approach that we propose in this paper is focused on the re-use of patterns and is based on various research work already carried out in the software engineering field.

In this paper, we shall first develop the problem of PIS engineering, relating to the very specificity of these systems, and we shall define the main methodological problems involved in PIS development currently faced by industrial companies. Then, after we have shown the need for a re-use approach in PIS engineering, we shall distinguish various forms of re-use already introduced in software engineering, with particular emphasis on re-use of patterns. We shall then develop the methodological approach that we have adopted, based on re-use of patterns throughout the PIS life cycle (from formulation of needs to implementation and evolutive maintenance). We shall present the approach that we used to identify and specify business patterns. Finally, we shall conclude on current work and the future prospects of our research.

2. The PIS engineering problem

2.1. PIS specificities

The PIS occupies a key position in today's industrial companies. It supports management as a whole of technical information, of the documentary database accompanying product development and of all the tasks in the development process and their breakdown between the various entities, etc. The PIS assumes the form of a socio-technical system, characterised by organisational and computer-related aspects. It is structured around four main components:

- information on the products concerned by the PIS and which form its core. These products must be defined, characterised, followed up and, in particular, upgraded, both generally but

also specifically according to their application in a Study/Design/Manufacture/Sales/After-Sales process. These products are not inter-independent and their links are varied: structuring, dependency, family links, etc.;

- the documents which form the external part of the PIS: these are technical documents adapted to the functions of the various people involved (drawings, instruction sheets, maintenance documents, etc.);
- the exchange files with the other systems: these are for example calculation tools, simulation systems, etc. which are vital elements in the opening of this system type;
- the procedures («workflow ») which codify the activities [SCH 97a, 97b, 97c] and the roles of the various partners involved (designer, supplier, seller, etc.), the rules governing organisation, exchanges, decision-making, access rights, etc. It is from consideration and formulation of each person's contribution to the resolution of the problem and of his/her various constraints (internal organisation, available human and material resources, etc.) and from people's co-operation with one another that a solution will emerge.

Faced with the need for industrial companies to develop PIS, a software package offer was created, proposing technical data management systems (TDMS) [RAN 95]. Large industrial companies (Schneider, Boeing, etc.) increasingly use such systems to implement PIS. Most TDMS, for example the Metaphase product [SDR 96], are in actual fact large tool kits designed to define and use existing components.

2.2. Methodological problems relating to PIS development

Just like most information systems, the development cycle of a PIS consists of chronological stages: formulation of needs, design, production and maintenance. Throughout the PIS development cycle, industrial companies currently experience technical, methodological, organisational and human problems that we shall briefly describe below.

- Formulation of needs: this is an important stage whose aim is to draw up a contractual file between the partners, i.e. the software engineer and the user. This file, which may be updated, specifies the constraints and objectives of the target information system. This initial stage is complicated by the lack of "formal" patterns that can easily be understood by the user and by the lack of procedures able to draw up clear, unambiguous requirement specifications.
- Design: based on formulation of needs, this stage defines the PIS specifications that will be considered by the application managers in charge of their implementation. At present this specification has no real continuity. The PIS project manager normally draws up two separate files: one for the customer and one for the software engineer team. Any consistency between what is expected and what is actually delivered is purely intentional.
- Production: this stage makes increasing use of TDMS, tool platforms adapted to characterisation of items, bills of materials, documents, procedures, etc. whose implementation is extremely complex and complicates the consideration of the above-mentioned specifications. This problem does not only arise during specification and implementation of a PIS, but also when the system is upgraded in the company to allow for changes in concepts (e.g. objectives), organisation (e.g. new information flows, upgrading of quality procedures) or technical aspects (e.g. introduction of an ad-hoc software, version change of the TDMS software package) affecting the PIS.
- Evolutive maintenance: the purpose of a PIS is to support a work organisation involving human and material resources. The needs of this organisation change continually and require evolutive maintenance of the supporting PIS, resulting in definition of new functions able to generate organisational changes and evolutive maintenance of the software parts of the system. Neither implementation nor upgrading of PIS in the organisation is currently dealt with methodically, thus generating considerable costs and malfunctions.

With respect to these problems concerning each stage in the PIS development cycle, we must not forget that the purpose of these systems is to support the product life cycle, i.e. accompany it, within the industrial organisation, throughout design, manufacture, production,

etc. Industrial companies are therefore obliged to constantly develop new PIS for each new product or product family. The life cycle of industrial products is becoming increasingly shorter: that of the supporting PIS also. One of the main very familiar problems in PIS development is the displacement of the target to be achieved during development, resulting in the delivery of systems that are obsolete even before they are put into production.

3. A methodological approach focused on re-use of patterns

3.1. The need for re-use in PIS development

The problems that we have mentioned above and, in particular, the need to minimise the duration of the various stages in the PIS life cycle, lead us to propose a new methodological approach. This approach, if it is to be adapted to the specificity of these systems, needs to be different from those recommended by traditional methods. In actual fact, systemic methods such as object-oriented methods are not suitable for PIS design. First, none of them completely covers PIS complexity: for example they do not provide patterns suitable for representation of processes, product versions, etc. Second, the patterns they propose are too generic to be either accessible to the users validating them or easily adaptable to a specific field, in this case the PIS. Moreover, these methods do not encourage capitalisation and re-use of concepts

An efficient means of considerably reducing the length of the various PIS development stages is to allow "deviation" specifications, both as regards capitalisation and re-use of concepts already encountered and consideration of the software resources (components and systems) available.

Consequently, the re-use approach, already effective in software engineering, is a key factor to our PIS development methodological approach, both for specification and production of PIS and for their evolutive maintenance. We deliberately place ourselves in an approach for capitalisation and re-use of acquired knowledge as regards product and process patterns, at each stage of the PIS development cycle:

- concerning formulation of needs, the aim is to re-use partial formulations of needs taken either from standard PIS associated with standard products and manufacturing processes, that may be provided by TDMS editors, or from PIS already developed in the industrial organisation;
- concerning design and evolutive maintenance, re-use of specifications already proposed by the TDMS or taken from other already designed PIS, should allow a marked reduction in lead times;
- concerning production, re-use of standard software components proposed by these TDMS (which, do not forget, are mainly tool kits) enables the PIS implementation stage to be speeded up. However, these software components are barely or poorly documented, and the PIS project manager must control them completely if he is to prepare detailed specifications indicating what needs to be recovered and how. Only features in these specifications that are different from existing features can be processed.

In order to understand such capitalisation and re-use of acquired knowledge as part of this PIS engineering methodology, we considered it essential to study the various re-use techniques already developed in software engineering (object-oriented) and in particular the technique for re-use of patterns that we have chosen for the PIS.

3.2 Various forms of re-use in software engineering

A general definition of the term "re-use" could be: a new development approach by which a system can be built from existing components. Today this approach is extensively used in software engineering, and different forms of re-usable software components have already been proposed. Three major approaches can be identified: toolkits, frameworks and patterns:

- The toolkit approach [POU 95] is the oldest and most basic. It offers software component sets, whose search, composition and adaptation remain the responsibility of the developer. The granularity of these components is directly linked to the builders of a programming language (class, procedure, function, etc.) and their level of abstraction is low since these components provide neither the context in which they can be used nor the adaptations that can be made to them.
- Frameworks [WIL 90, FUK 93] are in most cases dedicated to a specific application area, and propose global standard architectures for an area. Frameworks have been proposed for graphic interface design [WIL 90, WEI 88] and operating systems development [MAD 89]. They have a real advantage over toolkits, in that the developer no longer has to choose the classes, supply the interconnections, discover which methods are available and find the ones that must be called and in what order. Frameworks conceal this complexity by offering a higher level of abstraction. However, in some situations, their size may make them too rigid.
- Patterns were introduced by Alexander [ALE 77] in the field of architecture. This approach is currently experiencing a real success with the design patterns of E. Gamma [GAM 94]. Design patterns are descriptions of communicating objects and classes that are customised in order to solve a general design problem in a specific context. This form of component has several advantages over the above approaches. First, the granularity of the pattern provides a very modular reasoning unit, as each pattern exists to solve a standard problem. Furthermore, integration in the same pattern of a standard problem and a solution forms a component search and integration aid. Naturally, problems have still to be solved regarding pattern composition and organisation in order to ensure effective re-use.

The re-use of patterns is, in our opinion, the most suitable form of re-use for PIS engineering, as it can be used in all stages of the PIS development cycle (formulation of needs, design, implementation). The patterns used for formulation of needs provide solutions for application area problems, whereas those used, for example, for implementation provide solutions for technical problems in the context of specific TDMS.

In the next section, we shall develop the re-use of patterns and show its advantages for PIS design.

3.3 Re-use of patterns

The pattern concept was initially proposed in the field of architecture [ALE 77, ALE 79] before being more recently used in software engineering [GAM 94] [BUC 96] [COP 96] [FOW 97]. Alexander compares a pattern to a formulated know-how: "Each pattern describes both a **problem** that **frequently** occurs in your environment and the architecture of the **solution** to this problem, so that you can **use** this solution millions of times without ever **adapting** it twice in the same way ».

Patterns were presented by K. Beck and W. Cunninghan [BEC 87] as an adaptation of Alexander's pattern language to object-oriented design and programming. As part of information systems engineering, P. Coad [COA 92] proposes simplifying system analysis by identifying needs according to seven pre-defined patterns. R. Johnson, [JOH 93], proposes a breakdown of applications into frameworks, based on design patterns that can be re-used and that are made up of pre-developed object classes. C. Rolland's team [ROL 93] [CAU 96]

continues these approaches in two directions by proposing greater integration of dynamic aspects and introducing re-use of knowledge on the application areas. Pattern catalogues are currently proposed [WHI 94, VLI 96] and distributed. The catalogue proposed in [GAM 94] contains twenty-three patterns for software object-oriented design, and is at present the most representative and best formulated.

In E. Gamma's opinion [GAM 94], the four basic areas for documenting a pattern are: pattern name, description of the problem to be solved, the static/dynamic solution and the advantages of applying the pattern. The general form of a pattern is given in the figure below:

Name: pattern name;
Intention: the problem to be solved;
Synonyms: similar patterns in other pattern languages;
Motivation: a pattern application scenario, specific problems;
Application: situations in which this pattern can be used;
Structure: a graphic representation of the pattern using the OMT notation;
Participants: describes the classes and/or objects participating in the pattern and their responsibilities;
Collaboration: describes how the participants collaborate to assume their responsibility;
Advantages: presents the advantages of the pattern;
Implementation: the implementation tricks and tips;
Code sample: code fragments illustrating pattern implementation in C++ or Smalltalk;
Applications: real application examples of this pattern;
Related Patterns: other patterns used with (or by) this one.

Figure 1: Formal description of patterns in [GAM 94]

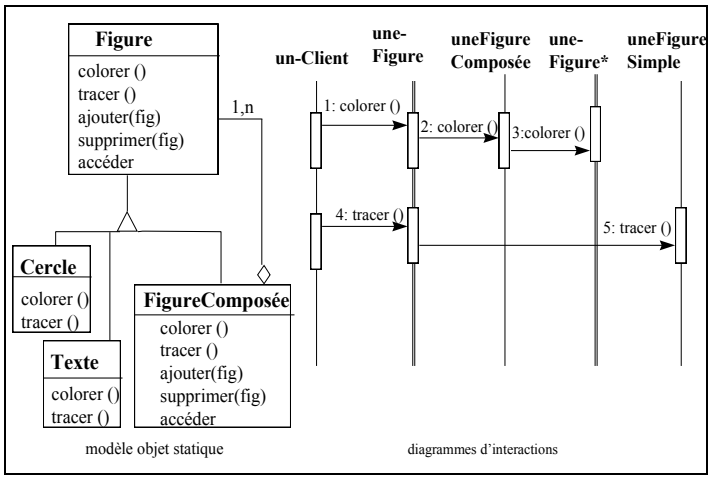
To illustrate the notion of pattern, we partially present the design pattern called '*Composite*' in [GAM 94]. A similar pattern called '*Compound Part-Part*' was also proposed by P. Coad [COA 95].

Name: 'Composite'

Intention: manage a recursive object composition.

Motivation: Graphic editors allow recursive elaboration of composite figures from simple, predefined figures. A solution is to define one class to manage complex figures and another to manage basic figures (text, circle, etc.). In this case simple objects are treated differently from composite objects, thus making applications weighty.

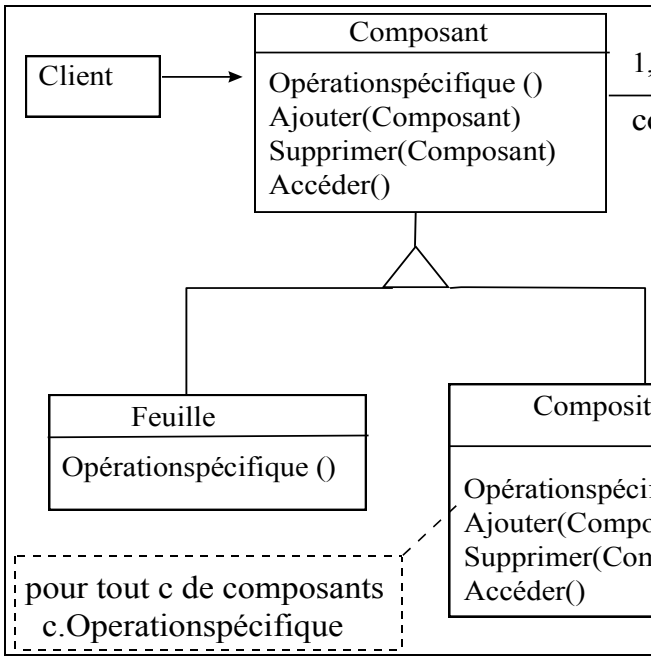
The solution proposed by the '*Composite'* pattern is to define an abstract class (noted *Figure*) which represents both composite and simple objects (or leaves). The *Figure* class contains primitive operations such as *plot* and *colour* as well as component management operations (*access, delete, add a component*).



The sub-classes (*Circle, Text*, etc.) implement graphic primitives such as *colour* or *plot* in order to colour and plot a circle, text, etc. The *CompoundFigure* class also performs the *colour* and *plot* operations by recursive calls to the operations of its components: in order to colour a compound figure, all its figures (simple and compound) must be coloured.

Structure and Participants:

- *Component*: defines the common interface of the leaf and composite objects.
- *Leaf*: defines the behaviour of the leaf objects.
- *Composite*: defines the behaviour of the composite objects and implements the component management operations.
- *Customer*: handles the leaf and composite objects via the Component interface.



Consequences:

- Defines simple and composite object class hierarchies;
- Simplifies use of these objects for the customer who can handle them uniformly;
- Simplifies the addition of new components.

3.4 Re-use of patterns in PIS development

Re-use of patterns can be implemented in all stages of the PIS development cycle. This approach is both feasible and advantageous for the following reasons:

- **Diversity of people involved.** In industrial companies, technical information is intended for a whole host of people with well-defined businesses, knowledge and roles: project manager, engineering and design manager, industrialisation manager, project engineer, draftsman, digital code preparer, calculation engineer, etc. A pattern-based approach enables all the problems and solutions specific to each business to be capitalised in a customised manner.

- **Maturity of PIS design.** A pattern-based approach can be developed only for mature disciplines, i.e. those for which there is both a consensus established by a community of individuals around a finite set of problems and a variety of known solutions for solving these problems. The field of PIS engineering satisfies this condition, as terminological reference frameworks and standard procedure guides already exist. However, a large part of this knowledge continues to be dispersed among those working in this area, and an effort to acquire and represent this knowledge would contribute greatly to the rapid development of numerous PIS.

- **Documentation of PIS architecture.** The majority of TDMS builders offer software toolkits that simplify PIS implementation. However, these toolkits are rarely or poorly used, not because the artefacts that they implement are too far removed from the objects to be implemented, but because they are barely or poorly documented. Moreover, most of these toolkits are organised according to the solutions they offer and not to the problems they solve. The result is product software that is hard to understand and upgrade. Development of a PIS from patterns has two advantages: first, the PIS is systematically documented and, second, identification of new patterns is simplified.

- **Homogeneity of re-usable components**. The notion of pattern can be considered to be a generic concept (not dependent on a specific area or language) that can be used to describe in homogeneous and modular form a wide variety of component types. The notion of pattern can be used, for example, to describe both software design problems (and their associated solutions) and business problems (and their associated solutions). In both cases, only the nature of the problem is different. It seems possible to propose a methodological framework for PIS design in which the concept of pattern can be used at the various development levels (business patterns, software design patterns, b patterns, etc.).

4. Towards a methodological framework for pattern based PIS

4.1 Business patterns and process patterns

The methodological framework that we propose is based on a consistent set of models of varying levels of abstraction that can be prepared by re-use of patterns. Each level proposed must enable a problem to be solved (of a conceptual, organisational, technical nature, etc.) specific to PIS development. Just as for management information systems design [NAN 96] [ESP 97], two main aspects must be taken into consideration in PIS design: first, an organisational aspect specifying the organisational information system (OIS) and with which the "business" patterns will be associated, and, second, a technical aspect (computer) specifying the part of this OIS which will result in computerisation. This is the computer-based information system (CIS) which will result in specifying and re-using the "software" patterns.

- At organisational level, **business patterns** are very important, particularly in the stages concerning formulation of needs and definition of functional specifications. They must be able to take account of a set of information needs associated both with the various industrial processes of design, manufacture, production, quality, etc. and with the various businesses and hence the people whose co-operation is essential to carrying out these processes. Note that these people may be inside the industrial organisation considered, but also external, as in the case of subcontractors or of a distributed or virtual enterprise. In the definition of the organisational level, two forms of modelling are essential: product modelling and process modelling [HAR 97]. With respect to product modelling, the aim is to use and adapt the modelling solutions taken from information systems engineering. Adaptations are necessary in order to take into account product life cycle throughout the various processes (design, manufacturing, etc.) and product upgrading through changes (product version notion). With respect to process modelling, we shall first base ourselves on industrial engineering work and particularly on entreprise modelling and integration [LAD 95] [VER 96], as well as on new organisational techniques of the BPR type (Business Process Reengineering) [JAC95] and on work carried out in information systems process engineering [ROL96]. This field proposes process patterns allowing representation of "workflow" processes, as well as richer process patterns, integrating in particular the notion of decision [ROL 93] [ROS 91] [JAR 92]. A decision-oriented process pattern is useful for representing industrial processes which, by nature, result from decisions made by people involved. The co-operative nature of these industrial processes should also be taken into account in the process pattern proposed.

- At technical level, the definition of **software patterns** is strongly linked to the TDMS which are at the basis of PIS development. The generic nature of modelling based on software patterns stems from its independence from a TDMS. This modelling is the expression of a technical solution that takes two vital problems into account: implementation of product and process patterns and communication of the PIS with other systems. With respect to implementation of product and process patterns, the TDMS use database models (relational or object) and "workflow" models. In both cases the models proposed in the tools are extensively used and can thus be integrated in the methodological framework proposed. With respect to modelling of exchanges between the PIS and other systems, the format/syntax approach such as proposed in Step/Express [ARB 94] can be used. Although a certain amount of research work has already been conducted in the area of semantic integration, in our opinion format/syntax integration is sufficient for PIS.

Definition and specification of software patterns associated with PIS are based on the functions proposed by most TDMS [RIE 97] and the design pattern catalogues already available in software engineering [GAM 94] (see chapter 3). With respect to the "business" patterns, mainly used for formulation of needs and definition of functional specifications for PIS, the aim is to identify them from the area's activity. In this section we are concerned with identification and specification of business patterns, based on that of the Gamma software patterns

4.2 Identification and specification of business patterns

Business patterns provide the descriptions of the products and processes to be managed by the future PIS. These descriptions mainly formulate the area's semantics. There are two types of business patterns: **Product** patterns providing product structuring models with their documentation and **Process** patterns providing process description models including people's roles. At present we are mainly concerned with studying Product patterns.

In order to illustrate the specificity of this pattern class, we shall once again consider the 'Composite' pattern developed earlier. This is a pattern that can be re-used in most application areas. However, it can be customised according to the composition semantics used in a specific area. In point of fact, many research works [OUS 97] have shown that there is no

single object composition semantics. A component may or may not depend existentially and functionally on its composite. Composite (or component) properties may or may not be distributed towards its components (or composite). A component may or may not be shared by several composites, etc.

PIS form a never-ending source of such semantics, particularly since a product can be described from various points of view [TOL 95]: structural, functional, geometric, hydraulic, etc. Each point of view can be modelled by a composition structure: a function is made up of sub-functions, a structural component contains structural elements, etc. However, the composition structure semantics vary from one point of view to another. Sometimes even the same point of view can use several composition structures with different semantics. Moreover, inter-representation links must be expressed in order to maintain consistency between the various points of view: this is, for example, the case between structural elements composing a product and the functions that this product has to perform.

Identification and specification of business patterns must necessarily be based on area analysis, enabling a terminological reference framework to be established for this area.

4.3 Un exemple de patron métier pour le domaine des SIP en conception mécanique

6. Remerciements

Les auteurs tiennent à remercier Stéphane Guignard de la Société PCO Technologies et Paola Conforti de la Société Schneider Electric pour leur collaboration.

Bibliographie

[ALE 77] C. ALEXANDER, S. ISHIKAWA, M. SILVERSTEIN, and al.. *A Pattern Language*, Oxford University Press, New York, NY, 1977.

[ALE 79] C. ALEXANDER. *The Timeless Way of Building*, Oxford University Press, New York, NY, 1979.

[APP 97] B. APPLETON. *Patterns and Software* , Essential Concept and Terminology, http://www.enteract.com/~bradapp/docs/patterns-intro.html.

[ARB 94] S. ARBOUY, A. BEZOS & al. *STEP: Concepts fondamentaux*. AFNOR 1994.

[AIT 97] E.J. WAITE. *AIT - Advanced Information Technology for Design and Manufacture*, Proc. ICEIMT'97 International Conference on Entreprise Integration and Modeling Technology, Eds. K. Kosanke, J.G. Nell, 1997.

[BEC 87] K. BECK, W. CUNNIMGHAN. *Using Pattern Languages for Objet-Oriented Programs,* OOPSLA-87, 1987.

[BUC 96] F. BUCHMANN, R. MEUNIER, & al. *Pattern-Oriented Software Architecture. A System of Patterns*, WILEY & Sons, 1996.

[CAU 96] C. CAUVET, F. SEMMAK,.*Semantic units and connectons: towards domain knowledge reuse*, Proc. of the IFIPW.G.8 Conference Domain Knowledge for Interactive system Design, Geneve, Switzerland, 1996.

[COA 92] P. COAD. *Object-Oriented Patterns*, Communication of ACM, Vol 35 n° 9, Sep. 92.

[COA 95] P. COAD, D. NORTH, M. MAYFIELD. *Object Models Strategies Patterns & Applications*, Yourdon Press Computing Series, 1995.

[COP 96] J. O. COPLIEN. *Software Design Pattrens: Common Questions and Answers*, http://st-www.cs.uiuc.edu/users/patterns/.

[ESP 97] B. ESPINASSE, D. NANCI. *Merise et l'approche orientée objet: du couplage avec OMT à une troisième génération*, Revue Ingénierie des systèmes d'information, Hermes, Vol 5, N° 4, Octobre 1997.

[FOW 97] M. FOWLER. *Analysis Patterns, Reusable object models*, Addison-Wesley, 1997.

[FUK 93] A. FUKANAGA, W. PREE, T. KIMURA. *Functions as data objects in a data flow based visual language*, ACM Computer Science Conference, Indianapolis, 1993.

[GAM 94] E. GAMMA, R. HELM, R. JOHNSON, J. VLISSIDES. *Design Patterns: Elements of Object Oriented Software Architecture*, Addison-Wesley, 1994.

[HAR 97] Y. HARANI. *Une approche multi-modèles pour la capitalisation des connaissances dans le domaine de la conception*, thèse de doctorat de l'INPG Grenoble, 1997.

[JAC 95] I. JACOBSON, M. ERICSSON, A. JACOBSON, *The object advantage: Business Process Reegineering with object technology*, Addison Wesley, 1995.

[JAR 92] M. JARKE, J. MYLOPOULOS, J.W. SCHMIDT, Y. VASSILIOU, *DAIDA - An environnement for evolving information systems,* ACM TOIS, vol 10, n°1, 1992.

[JOH 93] R. JOHNSON. *How to design frameworks*, Tutorial Notes, OOPSLA'93, 1993.

[LAD 95] P. LADET, F. VERNADAT. *The dimensions of Integrated Manufacturing Systems Engineering. Integrated Manufacturing Systems Engineering*, Ed. P. Ladet, F. Vernadat, Chapman & Hall, 1995.

[MAD 89] P.W. MADANY, R.H. CAMPBELL, V.F. RUSSO, D.E. LEYE NS. *A Class Hierarchy for building Stream-oriented file systems*, In Proc. Of the ECOOP'89 Conference, Nottingham, UK, 1989.

[MUL 97] P.A. MULLER, *Modélisation Objet avec UML*, éditions Eyrolles, 1997.

[NAN 96] D.NANCI, B.ESPINASSE et al., *Ingénierie des systèmes d'information: Merise deuxième génération*, Sybex, 1996.

[OPA 97] *OPAL, Integrated Information and Process Management in Manufacturing Engineering*, Esprit 4 Project, 1997.

[OUS 97] M. OUSSALAH, J.P. GIRAUDIN, F. BOUNAAS, D. RIEU, & al. *Ingénierie des objets: Concepts, techniques et méthode,s*. InterEditions, Mai 1997

[PIS 94] *PISA, Platform for Information-Sharing by CIME Applications*, Esprit 3 Project, 1992-1994.

[POU 95] J.S. POULIN. *Populating Software Repositories: Incentives and Domain Specific Software*, Journal of Systems Software, 30, pp 187-199, 1995.

[RAN 97] H. RANDRIAMPARANY. *Les Patrons Orientés Objets: définition et utilisation,* DEA Systèmes d'Information MATIS, Grenoble, Juin 97.

[RAN 95] J.M. RANDOING. *Les SGDT*, Editions Hermes, 1995.

[RIE 97] D. RIEU, M. TOLLENAERE et al. *Patrons d'Objets pour les SGDT*, 2ème congrès international Franco-Québécois: le génie industriel dans un monde sans frontières , 3-5 Septembre 1997.

[RIS 97] *RISESTEP, Enterprise Wide Standard Access to Step Distributed Databases*, Esprit 4 Project, 1996-1998.

[ROL 93] C. ROLLAND. *Modeling the Requirements Engineering Process*, Proc. Fino-Japanese Seminar on Conceptual Modeling, 1993.

[ROL 96] C. ROLLAND. *L'ingénierie des processus de développement de systèmes: un cadre de référence*. Revue Ingénierie des Systèmes d'Information, Hermes,vol 4, n°6, 1996.

[ROS 91] T. ROSE, M. JARKE, M. GOCEK, C. MALTZAHN, H. NISSEN. *A Decision-Based Configuration Process Environment*, Software Engineering Journal, No 6,5, 1991.

[SCH 95] H. ALBRECHT SCHMID. *Creating the architecture of a manufacturing framework by design patterns*, OOPSLA 1995.

[SCH 97a] T. SCHAEL. *Théorie et pratique du Workflow, Des processus métier renouvelés*, Springer-Verlag, 1997.

[SCH 97b] T. SCHAEL. *Cooperative Process and Workflow Management for Enterprise Integration*, Proc. of ICEIMT'97, International Conference on Entreprise Integration and Modeling Technology, Eds. K. Kosanke, J.G. Nell, Spriger Verlag 1997.

[SCH 97c] A.-W. SCHEER, R. BOROWSKY, S. KLABUNDE, A. TRAUT. *Flexible Industrial Applications Through Model-Based Workflows*, Proc. of ICEIMT'97, International Conference on Entreprise Integration and Modeling Technology, Eds. K. Kosanke, J.G. Nell, Springer Verlag1997.

[SDR 96] SDRC, Metaphase, http://www.sdrc.com/nav/software-services/metaphase/

[TER 92] G. DE TERSSAC, P. DUBOIS. *Les nouvelles rationnalisations de la production*, Cépadués-Editions,1992.

[TOL 95] M. TOLLENAERE. *Modélisation objet pour la CFAO: modèle de conception*, projet Shood, Rapport de fin de contrat 1995, Région Rhône-Alpes, Pôle Productique.

[UML 97] *Unified Modeling Language*, Version 1.0, January 97 , Rational Software Corporation. Http://www.rational.com.

[VER 94] F. VERNADAT. *Future R&D Directions for CIM Deployment*,, European Workshop on Integrated Manufacturing Systems Engineering, Grenoble, France, dec 12-14, 1994, pp. 3-6.

[VER 96] F. VERNADAT. *Enterprise Modelling and Integration: Principles and Applications,*. Chapman & Hall Ed, 1996.

[VLI 96] J. M. VLISSIDES, J. O. COPLIEN, AND N. L. KERTH. *Pattern Languages of Program Design 2*, Addison-Wesley. 1996.

[WEI 88] A. WEINAND, E. GAMMA, R. MARTY. *ET++ An Object-oriented Application Framework in C++,* In OOPSLA'88, Special Issue of SIGPLAN Notices, 23(11), 1988.

[WHI94] B.G. WHITENACK. *RAPPel: a Requirement Analysis Process Pattern Language for Object Oriented development,* http://www.bell-labs.com/user/cope/Patterns/Process/RAPPeL/rappel.html

[WIL 90] D.A. WILSON, L.S. ROSENSTEIN, D. SHAFER. *Programming with MacApp*, Reading, Massachusetts, Addison-Wesley.