Volume 2, No. 1, Jan-Feb 2011

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Customer Satisfaction within an Organization driven Aspect Oriented Business Component Model

Atsa Etoundi Roger
Department of Computer Sciences,
University of Yaoundé I, UYI
Yaoundé, Cameroon
roger.atsa@uy1.uninet.cm

Fouda Ndjodo Marcel
Department of Computer Sciences,
University of Yaoundé I, UYI
Yaoundé, Cameroon
marcel.fouda@uy1.uninet.cm

Atouba Christian Lopez*
Department of Computer Sciences,
University of Yaoundé I, UYI
Yaoundé, Cameroon
achristianlopez@yahoo.fr

Abstract: The Component Based Software Development is an activity which saw its importance increase within time. However, limitations have been detected in classic approaches of components development; notably weakness in the ability to reuse components. It is in most cases caused by crosscutting concerns. We propose in this paper, an Aspect-Oriented Business Component Model based on the concepts of Aspect approaches to increase the reusability of these business components. It is constructed from knowledge bits of user requirements in an organization, enhanced by performance indicators related to customer satisfaction of that organization. Because we cannot predict the axes of the developer's concerns, it advocates the gradual incorporation of these throughout the development process. This model's main advantage is that it allows the substitution of some modular integration units with the suitable one. The defined model integrates as basic concerns only concerns relative to business and those relating to the satisfaction of the organization's service delivery. The obtained model offers the possibility to apply substitution among internal elements of a business component in order to reuse certain aspects of a model.

Keywords: Business Process Modeling, Requirement Engineering, Software Component, Software Development, Subject-Oriented Programming

I. INTRODUCTION

The crosscutting Concerns are non-business features (security, interface, data persistence, etc) necessary for the implementation of computer systems. A developer is often faced with this kind of functionality when developing a large application. In this case, even if you apply a good vertical modularization of business concerns, there will always be a problem of horizontal concerns that cross all business modules. In [7], it appears that these cross-cutting concerns induce, in traditional approaches to development, two fundamental problems: the problems of dispersion and tangled code. Thus limiting the capacity for reuse and evolution of computer systems [7, 9] since these reusable parts, include specific details for a specific use case. Particularly, details of data persistence, security, man-machine interface, etc. However, reuse is presented in the literature as a fundamental concept in contributing to reducing costs and production time of computer systems. In addition, this concept emphasizes the notion of "business component"; and a business component is a materialization of the reusable parts of an information system. In order to increase the capacity for reuse of business components, it is essential to have a business component model capable of being adapted to fit in different situations. We believe that such a solution is possible only if there is a multidimensional separation of crosscutting concerns, notably data persistence, security, man-machine interface, etc. Aspect approaches are an alternative to these problems and require a multidimensional separation of different concerns [7, 9]. However, they are confronted with the lack of steps needed, from a business process requirement; identify the steps in the implementations of aspect oriented computer applications.

In the literature, several approaches exist for modeling business processes. However, research [1, 2, 3, 4, 5, 6] have particularly caught our attention, because they, through the Triangular Modeling Approach (TAM) [6] of a business process, allows the inclusion of client satisfaction of services rendered by a business process, and minimizing misunderstandings between developers and users of computer applications [1, 2]. We believe that this research [1, 2, 3, 4, 5, 6], reconciled with those of [7,9] the ongoing on Aspect approaches which allows us to get an adaptable

aspect-oriented business component model and enrich the inclusion of client satisfaction of service provided by the business process. This capability will be useful when discriminating among a list of components, those that best meet the problem. Moreover, these business components will certainly develop applications based computing satisfaction of service recipients of the organization. According to [10], it will be possible to make dynamic adaptability with such business components.

We propose in this paper, an approach that takes advantage of the development by Aspect [10] to define a business component model, based on a model of needs of a business process incorporating the recommendations of [6]. We hope that such a model will be interesting for an organization that is concerned with the satisfaction of its customers. Furthermore, provided the separation of cross-cutting concerns, such a model is easily adaptable and scalable.

Our work in the sequel will revolve around four sections. The first section will be devoted to Related Works, the second section will be devoted to basic concepts of our approach, the third section to the process of building aspect-oriented business components; last section is devoted to conclusion and perspectives.

II. RELATED WORKS

A. Business Components

The reuse of the knowledge of a domain, and most especially that from a particular business and which constitutes a business domain (BD) has grown significantly, because it has been the subject of much research [19] [23] [22] [21]. Such an approach is intuitive, from the moment we realize that certain activity domains result to the frequent manipulation of identical concepts. We then think to analyze these concepts and to create computer abstractions that could be reused in new developments. Thus, reusable components, which are not only for solving purely computer technical problems, but that meet the specific requirements of a particular domain, may also make easier, faster and less costly implementation of information systems in which they relate. It is in this sense that the Business component approach (BC) was proposed [19] [24] [20]. Thus such components are used for example in the fields of medicine, finance or accounting [26]. Research in the field of BC gave rise to numerous technologies based on components, which are now proposed and used in all activity sectors. Several definitions of the concept of business component exists in the literature, but that which retained our attention was given by [8] because it is a summary of definitions given by [19, 20, 23, 25, 27, 28]. [8] defines a business component as follows:

Definition (1): Business Component

A BC is a representation of an active concept in a business domain. A BC may be a composition of artifacts that ensure the completeness of its solution [8].

[8] tells us that, according to [19] [29], business components are classifiable into three types of knowledge: (i) business components of type entity: they represent real and "static" elements of the domain activity in question, and generally correspond to the elements identified during the design of data models, (ii) business components type process: they represent activities (business processes) of the domains in question. Users base themselves on this type of business components in accomplishing their tasks. From this fact, the process of manipulating and using business components of this type, (iii) the utility business components: this refers to business components that can be implemented in Information Systems relating to different contexts. They are used by business components of the process type and of the entity type. They are of smaller granularity than both categories of business components mentioned above. A measure, an address, a monetary value, as well as components such as spatial reference as points, lines or polygons, are examples of business components of type utility.

BC of type process in the context of organizations are presented as unstable elements, because they need to evolve to enable the organization to remain competitive [8]. In the following, the concept of business component will refer to the business component of the process type.

It appears, however, from the work of [7] that there are a number of limitations in component approaches particularly: (i) non readability and non intelligibility of code: entanglement (dispersion respectively) of code of one or several cross-cutting concerns with those (respectively, in one) relative to the set of basic concerns, often produces a final code involving many cross cutting concerns. This reduces the readability of the code that is less understandable, (ii) poor traceability: the cross-cutting concerns with their code dispersed in that related to the rest of the application, are difficult to locate in the final code and therefore not traceable; (iii) the maintainability and evolutivity are difficult: the code which is not readable and less understandable, it is very difficult to locate, maintain and evolve, the various concerns of the application. As a result, any evolution or development of one or more cross cutting concerns, is very complex since it affects all basic concerns in which it is involved; (iv) the low efficiency and productivity: the simultaneous definition of many cross cutting concerns at the same time and in the same modular unit, moving the attention of the developer away from the final goal of the application, to additional requirements, thus limiting its effectiveness and productivity, (v) low code reuse: the code of various cross cutting concerns being dispersed, it is not reusable. Thus, the code based on the concerns of being "non proper" (mixed with the cross cutting concerns that affect them), it is difficult to reuse in many other usage contexts of use.

In the paragraph below, we will introduce the concept of cross cutting concerns and that relating to the separation of concerns.

B. What is the Separation of Concerns?

Modern applications require more and more features in order to cope with growing problems; and can include for example: distribution and competition of data, the man-machine interfaces, real time considerations, persistence management or robustness to failures. The introduction of

these features increases the development for several reasons: (i) their implementation is dependent on changing technologies (software platform) (ii) they are rarely considered and identified during the design phase. These findings have given rise to a new paradigm: "the separation of concerns" [13]. This paradigm advocates the separation of concerns both during the design [11] and during the implementation. The separation of concerns highlighted three trends: (i) applications are composed of different concerns, (ii) the complexity of applications is constantly increasing, and (iii) the number of concerns required by applications is also increasing. The separation of concerns seeks to identify the different facets of a system such as: functional parts (structures and behaviors that match the business part of the application) and non functional parts (code synchronization, display of signs, treatment of persistence, transaction management, etc.).. The first distinction between different categories of concerns, is consistent with a design and implementation of a simplified, better understanding, a decrease in the coupling between concerns and more generally, greater re-use.

A concern is a generic concept describing a homogeneous entity. The notion of concern leads to new questions in the design and implementation: what are the entities that are destined to become a concern? Or, what concern is it made up of? We return to the problem of identifying different sorts of concerns.

Concerns should be separated from each other to allow their reuse in different contexts. The challenge is to overcome the problems of coupling inter concerns. Compared to the object paradigm, the separation of concerns allows to gain modularity and expressiveness in the design or implementation. To be reused, the concerns are composed between themselves and the rest of the implementation of the application. This phase of composition often requires mutual adaptations between the different concerns. Separation of concerns dislocates a set of problems related to the reuse toward the composition phase. This "weaving" of different concerns involves a usually non-orthogonal composition, which is to say that concerns may alter the semantics of other concerns. The separation of concerns has been subsequently improved and this improvement is known as the multidimensional separation of concerns. It corresponds to a separation of concerns that meets the following prerequisites:

A. To be able to categorize the concerns in several arbitrary dimensions; a dimension represents a set of concerns with common characteristics.
B. The simultaneous separation of concerns on an arbitrary number of dimensions, without having one privileged dimension that would course the decomposition along the other dimensions in a favored manner.
C. To be able to identify and encapsulate any type of concern.
D. To be capable of supporting new concerns and new dimensions of concern when introduced during the development cycle.
E. Take into account the concerns that have between them an assembly and whose behavior may interfere with one or more concerns.

An implementation that would provide full support for multidimensional separation of concerns would allow "a la carte" remodularization. For a developer, it would have as main interest to choose the most appropriate modularization unit, based on one or more concerns.

Multidimensional separation of concerns is a set of very ambitious targets. They are based on both languages at the application design. However, no existing technique meets these objectives and an important research activity has yet to be conducted [14].

In the following, we shall use for simplicity the term "separation of concerns" instead of "multidimensional separation of concerns". The following paragraph is dedicated to presenting the basic concepts of aspect approaches.

C. Aspect Approaches

AOP [15] is a model of separation of concerns based on the object paradigm (although the model was subsequently applied to other paradigms). This model formalizes in applications the interweaving of concerns which are not taken into account by the object paradigm. This intertwining (also called crosscutting) is the result of a weaving operation between trends that are associated to concerns. It results from the lack of separation between the component application concerns. AOP proposes new entities to address intertwining.

Intertwine elements are called aspects. One aspect is an abstraction of a concern, whose characteristic is to apply a set of classes. The implementation of an aspect is interwoven with the rest of the implementation. The object paradigm is always used to model behaviors that are naturally hierarchical.

Finally, since aspects are entire modules, we need to able to compose them with the rest of the system. The places where the aspects intervene with the rest of the program are called joint points. A join point can have a variety of granularity: a particular method, a set of methods, all methods of a class, all methods of a set of classes, etc.. Every join point has an associated contextual information that is usable to know the where it applies.

Aspects have methods that are attached to one or several joint points. Once attached to a joint point, the method is executed whenever the flow of program execution reaches this point. A modifier may specify the time of execution from the joint points: before, after, around, after exception or after return of value. Moreover, these methods have an additional instance variable named JoinPoint that encapsulates contextual information captured from the junction in progress: intercepted message, class addressed by the message, parameters, etc..

An additional step should be added to run a program with aspects: either a new stage of compilation that will compose the aspects and the "normal" program before sending it to the native compiler language, either a modification of the interpreter to be able to compose aspects at runtime. This step is called "of aspect" (aspect Weaver) and it implies that at some point of weaving the execution of aspects and objects are composed to provide the desired behavior. [16] and [17] proposed for this purpose examples of aspect weaver implementation.

Aspects have the ability to inherit from other aspects and encapsulate variables and instance methods or class. They can be instantiated and multiple instantiation policies are possible: an instance of the same appearance for all objects of a class, each object can have a proper instance of the same aspect, or a single common instance.

In the case of a reflective environment, the aspects specificities must also address the requirements for reflexivity. It is then necessary for the implementation aspect to be completely dynamic, for example see [18]. for example, to enable the changing of an aspect at runtime.

In conclusion, aspect oriented programming allows a modular implementation of concerns intertwine with the rest of the system. This paradigm has as benefits the advantages of modularity, which guarantees a simpler code, easier to develop and maintain and has a greater potential for reuse.

Despite the potential of Aspect approaches, software engineers seem to have no interest in the consideration of satisfaction of beneficiaries of services provided by the developed application. Moreover, according to [7], ultimately, the stakes are improving the quality of productivity. This highlights the need to take into account the vision of those for whom the services are intended for by the organization. In 2010, a number of studies [6] have been conducted on the inclusion of satisfaction of beneficiaries of services of organization, in the modeling business processes. This work resulted in a triangular business processes model. The next section will focus on the synthesis of work [1, 3, 5, 6] on the business processes.

D. Modeling Requirements of A Business Process

In [1], a goal oriented approach for defining a business process requirements model was defined. The approach proposed by [1] is goal oriented for the definition of a business process requirements model, taking into account their level of importance and limitations inherent to these requirements. The level of importance of a goal is the credit a user associates to this goal. Constraints are non-functional requirements related to what that goal must be met. The representation of the expressed requirements or knowledge bits proposed in [1] is defined as follows: $\partial = (\psi, \omega, \lambda, \delta, \nu)$ where: v is the name of a knowledge bit; ψ is the context in which the goal is defined; ω is the goal; λ is the business rule; δ represents the constraints; ν is the level of importance. This approach revolves around four main activities: elicitation of user requirements, selection of different goals, transformation of requirements into knowledge bits and ultimately the development of the requirement model. This approach has the advantages of: - reducing misunderstandings raised by [1] between application developers and users, - the reconciliation of the users requirements in the formal representation of those requirements, and finally, integration the level of importance of various aspects of the system and the constraints inherent to these requirements. However, this approach has as main weakness: (i) the lack of indicators on the validation of level of importance attributed to requirements, (ii) the lack of formalism for defining business rules, (iii) the lack of formalism for the representation of constraints, and lack of a formalism for describing the contexts of each of knowledge bit . Regarding shortcomings (ii) and (iii), the work of [3] helped overcome this difficulty by formalizing the specification of a business rule, this led to the formalization of the context of a knowledge bit : $nom\_règle = (contexte_{nom\_règle}, description_{nom\_règle})$

where: $contexte_{nom\_règle}$ is the context part of the business rule; $description_{nom\_règle}$ is the description part of the business rule. Each of these parts has been subject to detailed presentation in [3,4]. Regarding the shortcomings (i) the work [6] have helped raise this concern. These work has contributed significantly to the consideration of quality of service as perceived by beneficiaries of services of an organization in the modeling its business processes. These works, through the triangular modeling, have significantly enriched the modeling of business processes.

However, given the capital role of this research, it is essential that the reusable modules of information systems take these considerations into account. Thus, the definition of a new business component model is needed.

The next section will be devoted to basic concepts of our approach of definitions of a new model for business components which shall be “aspect oriented”.

III. BASIC CONCEPTS

In the previous section, the pre-requisites for understanding the concepts developed in the section below were presented.

In the works [1, 2], a representation of an expressed requirement was proposed. Unknowingly, the authors implicitly used a cross-cutting concerns separation approach. The representation of requirements proposed in [1,2] refers to the coordinates of a point in a coordinate system in which the axes are made of each component of a knowledge bit, thereby materializing, the various areas of concern that should be taken into account in defining aspects of the system to develop. However, it is impossible to say beforehand the number of axes of cross-cutting concerns that a software architect will take into account in developing an IT solution to a specific problem. Thus, we believe that we should define new concepts that take into account that the axes of concerns may change from one problem to another and by the aspirations of each IT solution designer.

Definition (2): Integrable units or Modular Integration units

Consider a hyperspace consisting of several areas of concern, we shall call integrated unit, a construction of language in question, which may be, for example, a variable declaration, a method (or function or business rule), a class, interface [7].

Given the heterogeneity of integrable units, it seems appropriate to give a formal definition of this concept. Thus, we define a integrable unit in the following manner:

$$Unity\_name = \langle port\_com, action \rangle$$

Where: *action* is either the specification of a business rule either the specification of a condition, of a function, or the specification of an interface; *port_com* describes the communication port of the modular integration unit. It is through this port that the integrable unit is composed to others in the definition of an aspect.

The specification of a business rule, of a function (method respectively) or a condition should be made following the rules described in [3], while those relative to interface is made literally. The communication port, meanwhile, will be structured as follows:

$\langle InputStream \mid Nothing, OutputStream \mid Nothing, goal, TypeLabel \rangle$

where: *InputStream* is the integrable data stream input unit; *OutputStream* is the data stream at the output of a integrable unit and *goal* is the goal that must be met by the integrable unit; *goal* indicates the goal that must be satisfied by modular integration unit; *Nothing* indicates, depending on its position in the communications port, the absence of data streams as input or output of the integrable unit; *TypeLabel* can be one of the following values: $Interface, rule, declaration, any$ where *Interface* indicates that the action part of the unit to which the port refers is an interface specification, *rule* indicates that the action part of the unit to which the port refers is a specification of a rule, *declaration* indicates that the action part of unit to which relates the port refers is a declaration of variables, *any* indicates that the communication port has no data stream at input and output. We also talks of an empty communication port.

The different streams will have the following structure:

$$XXX\_Stream = \left(Type \mid (,Type)^*\right)$$

Where: *Type* is a base type according to work [3]; XXX_Stream is either *OutputStream* or *InputStream*.

Two types of integrations of modular integration units exist: horizontal integration and vertical integration.

Definition (3): Vertical Integration

Consider two modular units U_1 and U_2, integrations, we shall say that U_1 is vertically integrated to U_2, denoted $U_1 \Delta_v U_2$, if and only if the input data stream of U_1 takes its values from intermediate results of U_2. Such units will be recorded formally as follows:

$$U_1 \Delta_v U_2 \equiv \left\langle U_1, U_2, \{U_1.p.InputStream, U_2.p.InputStream\} \right\rangle$$

Definition (4): Horizontal Integration

Consider two modular integration units U_1 and U_2, we shall say that U_1 is horizontally integrated to U_2, denoted $U_1 \Delta_h U_2$, if and only if U_2 uses the results of U_1 during its execution or rather the execution of U_2 is dependency of that of U_1. Such units will be recorded formally as follows:

$$U_1 \Delta_h U_2 \equiv \left\langle U_1, U_2, \{U_2.port\_com.InputStream\} \right\rangle$$

Definition (5): Constraints

In [1,2], the concept of constraints relative to the expressed requirement was defined, formally, we shall define a constraint as a set of modular integration units each belonging to a dimension of concern. The constraints are involved in the business rules in the form of predicates.

A. Basic Crosscutting Concern

We denote basic cutting concerns, any cross cutting concerns that we cannot do without in the modeling business processes, and this whatever the designer of the application and development approach used.

We considered in this work that there exist only two basic areas of concern: (i) functional concerns relating to the business (basic), and (ii) concerns relative to the satisfaction of customers of an organization (quality concerns (performance indicators, quality factors, etc). We believe that whatever software engineer, the modeling of a satisfaction oriented business process of customers of an organization, requires minimally taking into account these two concerns.

B. Initial Aspect

An initial aspect is a modular decomposition unit in which there are two integrated concerns that are functional concerns related to business and quality concerns.

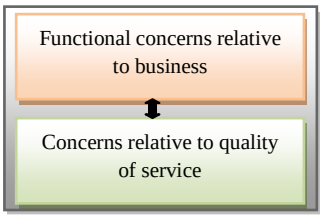

Figure 1 : Representation of Minimal Aspect

It is the representation of a point in a two-dimensional hyperspace that is the dimension of the basic functions and concerns of quality. The formal representation of the initial aspect is as follows:

$$smallAspect = \left\langle Unity_1 \left[, Unity_2\right]^+, Rule, \{port\_com\}^+ \right\rangle$$

Where: $\{port\_com\}^+ = \left\{port\_com, \{port\_com\}^*\right\}$, in the specific case of *smallAspect*, $\{port\_com\}^+ = \{U_1.port\_com, U_2.port\_com\}$; $Unity_1$ is integrable modular unit belonging to business dimension of concerns; $Unity_2$ is a modular integration unit of quality of service dimension of concerns of and *Rule* is the description of the integration rule of the two integrable units. This description is literally and refers either to Δ_h; either to Δ_v.

Our intuition has led us to this representation, because it is rare for business rules of a business process of an organization to evolve. Moreover, if they change substantially, there is a new business process. However, although there is consensus on business rules, other cross-cutting concerns mainly depend on the inspiration of the developer and the objectives assigned to him. We thought it useful to consider an evolution of the original aspect in function of the aspirations of those developers.

C. Evolution Principle of Initial Aspects

The previous representation of an initial aspect is based on the concepts of aspect approaches and particularly the Hyperspace approach [30] which proposes, that as new concerns, whenever they arise, by an incremental manner extend the hyperspace dimension with cross-cutting concerns; Thus inducing the need to extend the definition of the initial aspects in the new hyperspace. As illustrated in Figure 2, below.

Volume 2, No. 1, Jan-Feb 2011

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

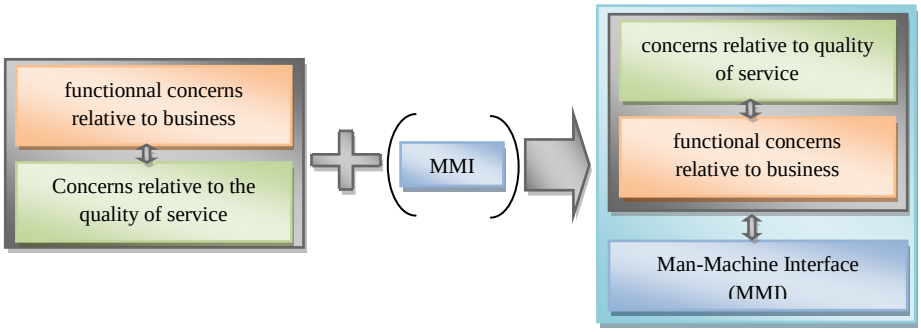

Figure 2: evolution of aspect after inclusion a new dimension of concerns

In the figure above, we have shown the inclusion new dimension of concerns in the representation of modular aspect decomposition. The '+' indicates the operator inclusion of new integrable MMI unit; it implicitly induces defining a rule for integrating the new integrable unit. To the left of the sign '+', we have an initial aspect, and to the right we have an integrable unit belonging to the new dimension of concern. The arrow indicates the final result after taking into account the new integrated unit. We denote by extended aspect, the aspect obtained after extending the initial aspect. Consider an initial aspect $smallAspect$ with $smallaspect = \langle unity_1, unity_2, rule, \{port\_com\}^+ \rangle$ and a new modular integration unit ihm with $ihm = \langle port\_com, action \rangle$ and r the integration rule induced by the inclusion of ihm. Formally the evolution of an initial aspect shall be denoted: $smallaspect + ihm = \langle unity, unity, ihm, new\_rule, port\_com \rangle$

Where: $new\_rule = rule \wedge r$; if r is a horizontal integration rule, $port\_com = smallaspect.\{port\_com\}^+ \cup \{ihm.port\_com\}$, if a rule is by cons r is a vertical integration rule $port\_com = smallaspect.\{port\_com\}^+$; $\wedge$ denotes the composition operator of integration rules.

'+' inclusion operator, integration of modular units is polymorphism and is defined as follows:

A. $+ : aspects \times unities \times rules \rightarrow aspects$ Such that $\forall a = (u_1,..,u_n, rule, p) \in aspects,\ b \in unities,$ and $r \in rules$: $+(a,b,r) = \langle u,...,u,b, rule \wedge r \rangle$

B. $+ : unities \times unities \rightarrow P(unities)$ Such that $\forall u, v \in unities$, $+(u,v) = \{u,v\}$ if $u \neq v$ and $+(u,v) = \{u\}$ if cons $u = v$.

C. $+ : aspects \times P(unities) \times P(rules) \rightarrow aspects$ Such that $\forall a = (u_1,..,u_n, rule, p) \in aspects,\ b = \{b_1,..,b_m\} \in P(unities),$ and $r = \{r_1,...,r_m\} \in P(rules)$: $+(a,b,r) = +\left(\left(...+\left(+(a,b_1,r_1),b_2,r_2\right)...\right),b_m,r_m\right)$

where: $aspects$ is the set of all aspects, $unities$ is a finite set of all modular integration units; $rules$ is a finite set of all the domain rules, $P(unities)$ the set of parts of $unities$, m represents a number of elements or any part of $unities$ or of $rules$.

While the conjunction operator is defined as follows: $\wedge : rules \times rules \rightarrow rules$

In order to simplify the writing, we shall write $a + b$ instead of $+(a,b,r)$, in lieu of $+\left(\left(...+\left(+(a,b_1,r_1),b_2,r_2\right)...\right),b_m,r_m\right)$,

We shall write $a + \sum_{i=1}^{m} b_i$; whereas $+(u,v)$ shall be $u + v$

***Property 1:* Commutativity of Composition Rules**

Consider two integration rules r and m, induced by the operator inclusion of modular integration units, the following entries are equivalent: (1) $m \wedge r$; (2) $r \wedge m$.

The process of inclusion of new concerns is called the extension process aspects. Within the extended aspect, several possibilities can be envisaged for the interaction between the integrable units contained in the aspect of modular decomposition.

In the following, the function $\dim type : Unities -> String$ such that for any modular integration unit u, $\dim type(u)$ returns a string representing the dimension of membership concerns of u.

D. Some Mathematical Characteristics

Definition (6): Right Commutativity

Consider three elements a,b,c and a relationship ϑ, we shall say that ϑ is commutative to right if and only if: $(a \vartheta b) \vartheta c = (a \vartheta c) \vartheta b$

Definition (7): Right Associativity

Consider any four elements a,b,c,d and a relation ϑ, we say that is right associative to a if and only if: $a \vartheta \left((b \vartheta c) \vartheta d \right) = a \vartheta \left(b \vartheta (c \vartheta d) \right)$

1) Commutativity at the Extremity

Consider an aspect $a = \langle unity_1, ..., unity_n, rule, port \rangle$ and two modular integration units $b = \langle action, port_b \rangle$, and $c = \langle action, port_c \rangle$ we shall show that the inclusion relation '+' is commutative

to right (or **extremity**).

[a] Evaluation of $(a+b)+c$, $(a+b)=\langle unity_1,...,unity_n,b,rule \wedge rule_b,port'\rangle$ where: $rule_b$ is the rule induced by the inclusion of b into aspect; $port'=port$ if $+\equiv\Delta_v$, else $port'=port\cup\{port_b\}$.

Assume that $a'=(a+b)$, $(a'+c)=\langle unity_1,...,unity_n,b,c,rule \wedge rule_b \wedge rule_c,port''\rangle$ where $rule_c$ the rule is induced by the inclusion of c into aspect ; $port''=port'$ if $+\equiv\Delta_v$, else $port''=port'\cup\{port_c\}$.

[b] Evaluation of $(a+c)+b$, $(a+c)=\langle unity_1,...,unity_n,c,rule \wedge rule_c,port'\rangle$ where: $rule_c$ is the rule induced by the inclusion of c into aspect a; $port'=port$, if $+\equiv\Delta_v$, else $port'=port\cup\{port_b\}$.

Assume that $a'=(a+c)$, $(a'+b)=\langle unity_1,...,unity_n,c,b,rule \wedge rule_c \wedge rule_b,port''\rangle$ where: $rule_b$ is the rule induced by the inclusion of b in aspect a'. $port''=port'$ if $+\equiv\Delta_v$ else $port''=port'\cup\{port_b\}$.

From a) and b), $(a+b)+c=(a+c)+b$ because '+ ' is right commutative or simply '+ ' is commutative at the extremity.

***Property 2*: Equivalences of entries**

The entries below are equivalent: (i) $(a+b)+c$;(ii) $(a+c)+b$; (iii) $a+(b+c)$; (iv) $a+(b)+c$;(v) $a+b+(c)$; (vi) $a+b+c$; (vii) $a+c+b$; (viii) $a+(c+b)$.

***Property 3*: Equivalence of entries**

Consider any aspect and a modular unit of integration, we have:

2) Associativity at the Extremity

Consider an aspect $a=\langle unity_1,...,unity_n,rule,port\rangle$ and three modular integration units $b=\langle action,port_b\rangle$, $c=\langle action,port_c\rangle$, and $d=\langle action,port_d\rangle$. Show that the inclusion relation '+' is associative to the right (or end), that is to say $(a+(b+c))+d=a+b+(c+d)$.

Assume $a'=a+(b+c)$, the appearance obtained after taking into account the integration and modular integration units b and c, we have $(a+(b+c))+d=a'+d$. According to Property 2, $a'=(a+b)+c$, as a result $a'+d=(a+b)+c+d$.

Assume $a''=a+b$, we have $(a+b)+c+d=a''+c+d$ from Property 2, we can write, $a''+c+d=a''+(c+d)$. By replacing by its value, we get $a+b+(c+d)$.

Consider a any aspect, and a set of modular units of integrations $b_1,...,b_n$ (n is a nonzero integer) to be taken into account in appearance of a, the appearance obtained after taking into account the n modulars integrations units will be represented as follows:

$$a'=a+\sum_{i=1}^{n}b_i$$

Axiom* 1**: ***Equivalence between Integration Rules

Consider two integration rules a and b, we shall say that a is equivalent to b, denoted $a\equiv b$, if and only if a and b are all integration rules, either vertical or horizontal.

Axiom* 2**: ***Equivalence between Modular Units of Integrations

Consider two modular integration units $u=<action,p>$ and $v=<action,q>$, we shall say that u is equivalent to v, denoted $u\equiv v$, if and only if $u.action=v.action$.

Axiom* 3: *Equivalence between Initial Aspect

Consider two initial aspects $a=<a_1,a_2,r_a,p_a>$ and $b=<b_1,b_2,r_b,p_b>$ we shall say that a is equivalent to b, denoted $a\equiv b$, if and only if the following are satisfied: (1) $a_1\equiv b_1$ and $a_2\equiv b_2$ or $a_2\equiv b_1$ and $a_1\equiv b_2$, (2) $r_a\equiv r_b$

Axiom* 4: *Equivalence between Business Aspects

Consider two business aspects a and b, a_0 and b the initial aspects respectively associated with the business aspects a and b, we say that the business aspects a and b are equivalent, denoted $a\equiv b$, if and only if $a_0\equiv b_0$.

Axiom* 5: *Substitution of Business Aspects

Consider two business aspects a and b (with $a=a_0+\sum_{i=1}^{n}a_i$; $b=b_0+\sum_{i=1}^{m}b_i$), the substitution of modular integration units of a with those of b, denoted $a\lhd b$, is defined as follows:

$$a\lhd b=a_0+\sum_{u\in C}u$$

Where: C is a set of modular integration units defined as follows: $u\in C\leftrightarrow u\in A,\acute{o}v\in B/\dim type(u)=\dim type(v)\grave{u}$; $A=\sum_{i=1}^{n}a_i$; $B=\sum_{i=1}^{m}b_i$.

Axiom* 6: *Substitution of a Modular Integration Unit

Consider a business aspect a and a modular integration unit b (with $a=a_0+\sum_{i=1}^{n}a_i$) the substitution of a modular integration unit b in a, denoted $a\rhd b$, is defined as follows:

$$a\rhd b=a_0+\sum_{u\in C}u$$

Where: C is a set of modular integration units defined as follows: $C=\{b\}\cup\left(\bigcup_{u\in A}\{u/\dim type(u)\neq\dim type(b)\}\right)\grave{u}$;

$$A=\sum_{i=1}^{n}a_i$$

Axiom* 7: *Restriction of Aspect

Consider two hyperspaces A having m (where m is an integer) dimensions of concern; and B having p dimensions of concerns (where p is an integer). Let A and B be such that $1\le p<m.$ let a, $a=a_0+\sum_{i=1}^{n}a_i$ an extended aspect belonging to the hyperspace A, we denote $\sum_{i=1}^{p}b_i$, the set of integration modular units of a which belong to the axes of concern are found in hyperspace B, the extended aspect $a'=a_0+\sum_{i=1}^{p}b_i$ is called restriction of a in the hyperspace B.

All these axioms help to show that the initial aspects

can be adapted into requirements. Consequently, business aspects, as derivatives of initial aspects, the same applies.

IV. CONSTRUCTION APPROACH OF ASPECT ORIENTED BUSINESS COMPONENT.

In the previous section, it was defined a set of basic concepts necessary to understand the approach we propose in this section. The approach we propose in this section is based on [1, 3, 6] relating to the modeling of business processes from the perspective of requirements. In [3] a formalism for the specification of business rules was defined. This formalism advocated the use of predicates in the sequence of a business rule. These predicates can be grouped into three broad categories: (i) the first is reserved for storage predicates. These are predicates that refer to persistent data (registration, update, etc...), (ii) the second is reserved to observers update predicates (this concept was defined in [6]) (iii) interoperability predicates. These predicates refer to those used for the interaction between the business rule and the external environment. These refers to predicates that relate to editing, input, and (iv) action predicates, they refer to those reflecting the business processing. The approach we propose in this section is based on specifying a business rule to determine the resulting aspect-oriented business component, as well as all necessary dimensions of concern.

A. Construction of Business Components Oriented Aspect

Consider a expressed requirement $b=(\psi,\omega,\lambda,\delta,\nu)$ in terms of Section 2.4,

Step 1: Construction of the initial aspect

This paragraph shows how, to construction an initial aspect from an expressed requirement.

[a] Construction of the modular integration unit $unity_1$ belonging to the dimension of concerns relative to the business rule. By definition $unity_1$ is expressed as: $unity_1=\langle port\_com, action\rangle$ Where: $action=\lambda$; $port\_com$ by definition is expressed: $\langle InputStream, OutputStream, goal, TypeLabel\rangle$, with $InputStream=\lambda.Contexte$; $OutputStream=\lambda.\text{results}$; $TypeLabel=rule$; $goal=\omega$.

[b] Construction of integrations modular units $unity_2$ belonging to the dimension of quality concerns. According to [6] several different types of observers can be associated with a task and therefore a business rule. When there is more than one observer associated to a rule, the following process must be repeated for each of them. By definition, for any observer associated with the rule λ, $eval\_unity$ expressed as: $eval\_unity=\langle port\_com, action\rangle$ where: $action=f_{eval}$, $InputStream=(type)$, $type$ indicates the type of the observer. $OutputStream=numeric$. $TypeLabel=rule$; $goal=users\ satisfaction$.

[c] we consider that there exist m observers associated to the rule λ, and $(eval\_unity_1,...,eval\_unity_m)$, the modular integration units each encapsulating an evaluation function to an observer respectively. The initial aspect of an expressed requirement, noted $smallaspect$, shall be defined as follows:

$$smallaspect=\left\langle unity_1, eval\_unity_1,...,eval\_unity_m, rule, \{port\}^+\right\rangle$$

Where: $unity_1, eval\_unity_1,...,eval\_unity_m$ taken under the preceding conditions; $rule$: "create joint points in the sequence part of λ, where there is use of the observers update predicate";

Step 2: evolution of the original aspect

This follows in step 1 above.

[a] obtain from the specification of the rule, all categories of predicates. Above we have listed a few of them. It may, however, happen that there are more categories than those previously listed. In the following, we focus only on those that do not refer either to quality concerns, or action concerns.

Each category of predicates refers to a dimension of concerns. Therefore, the set of modular integration units for each dimension should be determined. Suppose there exist n, except for quality or actions concerns that are already contained in the initial aspect, we shall denote $\bigcup_{i=1}^{n} cat_i$, the set of categories of predicates consider; $\bigcup_{j=1}^{p} unity_j$, all the set of p modular integration units of the same dimension of concerns that must be taken into account in the evolution of the original aspect of the expressed requirement. We shall denote $\bigcup_{i=1}^{n}\bigcup_{j=1}^{p} unity_{i,j}$ the set of all modular integration units of all dimensions to take into account in the evolution of the initial aspect. The aspect obtained after changing the original aspect will be formally defined as follows:

$$aspect=smallaspect+\sum_{i=1}^{n}\left(\sum_{j=1}^{p} unity_{i,j}\right)$$

Where: $unity_{i,j}$ indicates the $j^{ième}$ modular integration unit to be considered in the dimension of concern formed by the $i^{ième}$ category of predicates.

Definition (8): Business Aspect

Consider an expressed requirement $b=(\psi,\omega,\lambda,\delta,\nu)$ in the previous conditions, and $smallaspect$ the initial aspect associated with the requirement b. A business aspect is an aspect obtained after considering all the concerns encapsulated by different categories of predicates of a business rule. Formally, the business aspect will be represented as follows:

$$aspect=smallaspect+\sum_{i=1}^{n}\left(\sum_{j=1}^{p} unity_{i,j}\right) \quad (1)$$

Lemma (1): Completeness of aspects

[a] Any business aspect of is said to be complete;

[b] any business aspect can be written as:

$initialaspect + \sum_{u \in P(unities)} u$ where: $initialaspect$ is the initial aspect; $\sum_{u \in P(unities)} u$ is the set of changes of $initialaspect$.

Definition (9): Aspect Oriented Business Component

An aspect oriented business component is a unit formed by the composition of several business aspects. Formally, we define any business component c as follows: $c = \left\langle a[,a]^{+}, op \right\rangle$ where a is either a business aspect or a aspect oriented business component; op is a function of composition.

Several operators exist in the literature, those which retained our attention are:

- $[]$: is the alternative choices operator. This operator connects the input-output of business aspects (aspects oriented business components respectively);
- $+$: This operator assembles the aspects oriented business components (business aspects respectively) so as to encapsulate them in the order to create a new business-oriented aspect. components

Consider $BCaspects$, the set of business components; $BusinessAspects$, the set of business aspects, the set of parts $BusinessAspects$. Polymorphic operators composed of the above mentioned are defined as follows:

- $[]: BCAspects \times BCAspects \rightarrow BCAspects$
- $[]: BCAspects \times Aspects \rightarrow BCAspects$
- $+: Aspects \times Aspects \rightarrow BCAspects$
- $+: BCAspects \times BCAspects \rightarrow BCAspects$

B. Business Requirement Model to Aspect Oriented Business Component Model Transition

This part describes the process of transforming the business process requirement model a process to an aspect oriented business component model. The process of transforming the business process requirement model into an aspect-oriented business component is the bottom-up (ascending) process. That is to say from leaves to the root.

Step 1: creation of business aspects

Transform the basic requirements into business aspect.

Step 2: Creation of aspect oriented business components

[a] For any expressed requirement b (where b is an intermediate requirement), chooses from leaves to the root, first transform it into an intermediate aspect (extended aspect) which lacks only the transformation of the deterministic choice operator;

[b] Consider $b_1,...,b_n$, ($n \geq 2$) the set of sub requirements of b. We assume that $a_1,...,a_n$, are business aspects associated with the above mention sub requirements and that is a is the intermediate aspect of b in which lacks just the transformation of the deterministic choice operator. The business component associated with the knowledge bit is constructed as follows:

$$C = \left(...\left(\left(a' + a_1\right)[]a_2\right)[]a_3...\right)[]a_n$$

Where: a' is the aspect obtained after removal of the deterministic choice operator.

[c] the process b) must be repeated for all intermediary knowledge bits, from leaves towards the root.

The model obtained after transformation of the requirement model is the model of aspect-oriented business components associated with said business process.

C. Some Axioms on Adaptability of Aspect Oriented Business Components

<u>Axiom</u> 8: Completeness of Aspect-Oriented Business Component

An aspect-oriented business component will be called complete if and only if all aspects occupations that compose part of the same hyperspace.

<u>Axiom</u> 9: Restriction of Aspect Oriented Business Component

Consider two hyperspaces A having m (where m is an integer) dimensions of concern, and B having p dimensions of concerns (where p is an integer). Let A and B be such that $1 \leq p < m.$ let $a,\ a =< a_1,...,a_n, op >$ an aspect-oriented business component for any a_i :

(a). if $a_i \in A, (i \in N^*, 1 \leq i \leq n)$ we note $a_i^{'}$, the restriction of a_i in the hyperspace B, $a' =< a'_1,...,a'_n, op >$ is called restriction of a in hyperspace B ;

(b). if $a_i (i \in N^*, 1 \leq i \leq n)$ aspect-oriented business component, then there exists a nonempty set A_{a_i}, of business components such that $\forall b \in A_{a_i}, b =< b_1,...,b_t, o_p >$, where: are b_i any business component, and o_p the composition operator of b_i. for each business component e of A_{a_i}, apply (i). Intermediate aspect-oriented business components intermediate remain unchanged. Only those belonging to A_{a_i} are restricted.

Axiom 10: Extension of an Aspect Oriented Business Component

Consider two hyperspaces A having m (where m is an integer) dimensions of concern, and B having p dimensions of concerns (where p is an integer). Let A and B be such that $1 \leq p < m.$ Let $a,\ a =< a_1,...,a_n, op >$ an aspect-oriented business component for any a_i :

(a). if $a_i \in B,(i \in N^*, 1 \le i \le p)$,we note $a_i^{'}$, the extension of a_i in hyperspace A, $a' =< a'_1,...,a'_n, op >$ is called the extension a in the hyperspace A;

(b). if $a_i (i \in N^*, 1 \le i \le n)$ is an aspect-oriented business component, then there exists a nonempty set of business components such that $\forall b \in A_{a_i}, b =< b_1,...,b_t, o_p >$, where: b_i are any business aspects, and o_p the composition operator of b_i. for each business component e of A_{a_i}, apply (i). Intermediate aspect oriented business components remain unchanged. Only those belonging to A_{a_i} are extended.

<u>Axiom</u> 11: Substitution of a Business Aspect in Aspect Oriented Business Component

Consider an aspect oriented business component $a =< a_1,...,a_n, op >$ and a business aspect b, the substitution of a business aspect $a_k (k \in N^*, 1 \le k \le n)$ of a, by the business aspect b, denoted $a.a_k \rhd b$, is defined as follows:

$$a.a_k \rhd b =< a_1,...,a_{k-1}, b, a_{k+1},...,a_n, op >$$

<u>Axiom</u> 12: Equivalence between Aspect-Oriented Business Components

Consider a and b two aspect-oriented business components, we shall say that a and b are equivalent if and only if the business aspects that compose them are equivalent.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed in a first step, defining a set of new concepts in relation to the definition of an aspect oriented business component model, in a second step, we defined an approach for building these aspects oriented business components from a business requirement model to a business process, thereby creating a multidimensional separation approach for modeling business processes. This approach introduces the consideration of the aspect concept in the business components, thus inducing the principle of partial evolution of an aspect oriented business component. That is to say that some modular integration units may change without destabilizing the other parties. Thus, it may occur that it is necessary to change the man-machine interface without touching the part concerning performance indicators. In addition, the component model in its definition already incorporates the concept of taking into account new concerns, that is to say that if one dimension of the concern is found missing, taking into account the modular integration unit belonging to this dimension will be done without having to touch the part already designed. To achieve our objective, we initially defined the concepts of initial aspect, intermediary aspect, business aspect; and then the transitory rules from the requirements models to business components model. However, we did not insist on treatment associated with input/output of different business components in the composition of oriented aspect business components because that is the subject of ongoing work. In the coming days, we plan:

– To finalize work on the aspects oriented business components. Especially on the sections relating to the treatment of input-output of such components during their composition;
– To define a code generator from the model of aspect oriented business components;
– To define a platform for identifying a system requirements model and in the same vein to identify reusable requirements;
– To enrich the work on selection of software components.

The purpose of all these works is to implement a component-based development platform from a requirements specification closer to the human language and which takes into account the expectations of the beneficiaries of the service rendered by the business process. This course will surely minimize misunderstandings between developers and business executives, and produce systems based on software components of lower costs while managing the changing requirements in a business process.

VI. REFERENCES

[1] Atsa Etoundi Roger, Fouda Ndjodo Marcel, Atouba Christian Lopez, A Goal Oriented Approach for the Definition of a Business Process Requirement Model. *International Journal of Computer Applications* 9(7):1–7, November 2010.

[2] Roger Atsa Etoundi, Marcel Fouda Ndjodo, Atouba Christian Lopez, *A Goal Based Approach for QFD Refinement in Systematizing and Identifying Business Process Requirements*, International Journal of Computer Science Issues, Vol. 7, Issue 6, pp 343-350, 2010.

[3] Atsa Etoundi Roger, Fouda Ndjodo Marcel and Atouba Christian Lopez, A Model based Business Process Requirement Rule Specification. *International Journal of Computer Applications* 11(9):17–24, December 2010.

[4] Atsa Etoundi Roger, Fouda Ndjodo Marcel, Atouba Christian Lopez, *Business Process Requirement Engineering*. International Journal on Computer Science and Engineering, volume 2, n° 9, December 2010.

[5] Atsa Etoundi Roger, Fouda Ndjodo Marcel, Atouba Christian Lopez, *A Formal Approach for the Inclusion of Key Performance Indicators in a Business Process*. International Journal on Computer Science and Engineering, volume 3, n° 9, January 2011.

[6] Atsa Etoundi Roger, Fouda Ndjodo Marcel, Atouba Christian Lopez, Abessolo Alo'o G., Knowledge Management Driven Business Process and Workflow Modeling within an Organization for Customer Satisfaction, International Journal of Engineering Science and Technology, pp 7350-7362, December 2010

[7] Ouafa Hachani, Patrons de conception à base d'aspects pour l'ingénierie des systèmes d'information par réutilisation, thèse, Université Joseph Fourier-Grenoble I, Novembre 2006.

[8] Rajaa SAIDI, Conception et Usage des Composants Métier Processus pour les Systèmes d'Information, thèse, Institut Polytechnique de Grenoble, Septembre 2009.

[9] Laurent QUINTIAN, JADAPT : Un modèle pour améliorer la réutilisation des préoccupations dans le paradigme objet, thèse, Université de Nice Sophia-Antipolis, Juillet 2004.

[10] Daniel CHEUNG-FOO-WO, Adaptation Dynamique par Tissage d'Aspects d'assemblage, Thèse, Université de Nice-Sophia Antipolis, Février 2010.

[11] Carver L., Griswold W.G., « Sorting out Concerns », Workshop on Multi-Dimensional Separation of Concerns, OOPSLA'99, 1999.

[12] Fayad M. E., Schmidt D. C., Johnson R. E., Building Application Frameworks, Addison-Wesley Publishing Co., 1999.

Volume 2, No. 1, Jan-Feb 2011

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

[13] Lopes C.V., Hursch W., Separation of Concerns, Technical Repport NU-CCS-95-03, Northeastern University, Boston, February 1995.
[14] Tarr P., Harrison W., Ossher H., Finkelstein A., Nuseibeh B., Perry D., Summary of Workshop on MultiDimensional Separation of Concerns in Software Engineering, ISCE 2000.
[15] Kiczales G., Lamping J., Mendhekar A., Maeda C., Lopes C.V., Loingtier J.M., Irwin J., « Aspect Oriented Programming », ECOOP 97,1997.
[16] Fradet P., Südholt M, « AOP: towards a generic framework using program transformation and analysis », Workshop Aspect Oriented Programming, ECOOP 98, 1998.
[17] David P.C., Ledoux T., Bouraqadi-Saâdani N.M.N. « Two-Step Weaving with Reflexion using AspectJ », Workshop Advanced Separation of Concern, OOPSLA 01, 2001.
[18] Ségura-Devillechaise M., Menaud J.M., Muller G., Lawall J.L., « Web Cache Prefetching as an Aspect : Towards a Dynamic-Weaving Based Solution», AOSD 03, 2003.
[19] Herzum P., Sims O., "Business Component Factory: a Comprehensive Overview of Component-Based Development for the Enterprise", Wiley Computer Publishing, 1999.
[20] Heineman G., Councill W., "Component-Based Software Engineering: Putting the Pieces Together", addison-Wesley, 2001
[21] Hassine I., "Specification et formalisation des demarches de developpement a base de composants metier : la demarche Symphony", These de doctorat, Institut National Polytechnique de Grenoble, Grenoble, 2005.
[22] Cauvet C., Ramadour P., « Les composants metier dans l'ingenierie des systemes d'information », Vuibert (Ed.), Composants : concepts, techniques et outils, 2005.
[23] Barbier F., Atkinson C., "Business Components", Business Component-Based Software Engineering, Kluwer, vol. 705, Chap. 1, pp. 1-26, 2002.
[24] Bass L., Buhman C., Comella-Dorda S., Long F., Robert J., Seacord R., Wallnau K., "Volume I: Market Assessment of Component- Based Software Engineering", Carnegie University, Software Engineering Institute, TECHNICAL REPORT CMU/SEI- 2000-TR-008, ESC-TR-2000-007, Mai, 2000.
[25] Ambler S.W., "Process Patterns: building Large Scale Systems using Object Technology", SIGS Books, Cambridge University Press, Decembre, 1998.
[26] Andro T., Chauvet J.-M., « Objets metier », Eyrolles, Paris, 232 pp., 1998
[27] Cherbakov L., Galambos G., Harishankar R., Kalyana S., Rackham G., "Impact of service orientation at the business level", IBM Systems Journal 44(4): 653-668, 2005.
[28] Herzum P. and Sims O., "The Business Component Approach", OOPSLA'98 Business Object Workshop, Octobre, Canada, 1998.
[29] Schmid H.A., "Business entity and process components", Springer (Editor) OOPSLA'99, Business Object Design and Implementation, Denver, USA, 2 Novembre, 1999.
[30] H. Ossher and P.L. Tarr — Hyper/J^{TM}: Multi-dimensional separation of concerns for JavaTM. In Proceedings of the ICSE 2000, International Conference on Software Engineering, Limerick, Ireland, June 2000.