http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (6): 1146-1148, 2006
ISSN 1812-5638

A New Textual Description for Software Process Modeling

Atil Fadila and Ghoul Said
LRI Laboratory, University of Badji Mokhtar, BP 12, Annaba, Algeria
Department of Software Engineering, Philadelphia University, P.O. Box, 1101 Amman (11910), Jordan

Abstract: In the present study, we present a new textual approach for software process modeling that presents many conceptual advantages with regard to actual works in the domain. In fact, a software process is regarded as a set of sub-processes, which cooperates for realizing the same objective. This vision is natural and present contribution concerning construction and reuse of software process's methodologies. In this approach, it's even possible to associate versions of roles to the same process.

Key words: Activity, methodology, role, software process

INTRODUCTION

The software process development includes a complex set of activities dependent on each other which will give software products. The description of these activities and products are generally designed under the term cycle life development of software.

A description of the life cycle of software contains a clear indication of the implied activities, out products and their relations. It is a description which aims enriching and helping comprehension, as well as controlling the development software projects. Sight that with time, the software projects become increasingly large and complex, it is clearly proved that concentrate only with what is developed, i.e., the products, was not sufficient to guarantee development of software products of high quality, reliable and maintainable. For this reason, the need to give a particular attention to how the product was developed, i.e., the process, becomes very important.

A process model is a generic description of a class of software process. A model is independent of a particular project, but it can be adapted to the customer requirements, or instantiated to produce a particular software process adjusted with the needs for a particular project.

During years, a variety of software process models were conceived to structure and to describe the construction process of software systems. More recently, software process modeling treats new challenges amplified by the investigations with which industry of software must confront. Since modeling software processes is a difficult and expensive task, all approaches presented in the literature solve some problem, however, they are not yet entirely satisfactory.

In this study, we propose modeling software processes by a simple and natural way, using object approach. The problem that we have confronted is due to the fact that this approach doesn't allow the modeling of all dynamics and constant change of reality. To solve this problem, we attempt to use the role concept to express changes and to allow the definition of one or more methodologies controlling the behavior of the software process.

SOFTWARE PROCESS MODELING

The software process is a factor key which allows providing software systems with large quality, because it intends controlling and transforming the user's needs into a software product which meets these needs.

The term software process joins all activities that have to be achieved in order to develop software. The methods for implementation of activities depend on the type and content of development projects and technology used. For the same type of projects, the same sequence of activities and the same methods for their implementations are used (Horvat *et al*., 2000).

We can also define the software process as a sequence of operations required for building up various information objects (specifications, prototype documentation, test cases, code…) that compose a software product (Rueher and Michel, 1990). The software process can be split into sub-processes, but it is often very hard to find a good decomposition and to describe the complex way in which they must communicate. Processes are dynamic, hard to comprehend and to reason about.

Corresponding Author: Atil Fadila, LRI Laboratory, University of Badji Mokhtar, BP 12, Annaba, Algeria
Tel: +213 7 1159006 Fax: + 213 3 8872436

A process model is the formal expression of a part of the process, with the goal to understand, communicate, improve, support or automate the process (Estublier, 2005). Process technology supports a process in order to consistently reach the goal within predefined time, budget and quality constraints.

A software process model (SPM) is a descriptive representation of the software process structure, used as a reasoning support, allowing its understanding and its progress (Ghoul, 1995).

Analysis of any process get appears two levels: structural level which represent objects on which process's activities perform and methodological level describing the policies which lead the process and its component methods.

SPM = ({Methodologies: Policies, Mechanisms}, {Structures})

ROLE APPROACH

The concept of role is a concept which is not specific to a particular field, but it is rather regarded as a concept to integrate into the data models. This concept is intuitive and important allowing a simple and natural modeling of applications and facilitates comprehension (Atil *et al*., 2004; Atil *et al*., 2005).

The decomposition of the application domain into hierarchy of data abstraction is simple; on the other hand the determination of the behavior of these abstractions is not also easy.

Once the decomposition of data is made, we are in the presence of several functions that we must assign to the hierarchy of classes. These functions are not basic or low level functions which will be easily attached to each data abstraction but rather form part of the behaviors related to the application domain. These behaviors or high level functions are usually specified by the term Role.

During the description of a system, the experts of a given field often speak in term of these roles and functions. Some object oriented methods are more favorable to discover roles that others; the roles of the application domain can be found by inspecting the dynamic and operational views of the system such as use cases and the interaction diagrams between objects.

A fundamental aspect of the roles is that they are often detailed with a given situation. In example, a same person can play several roles; she can be a bank clerk and at the same time employees representative.

As in the real world, all the roles cannot exist concurrently; it is unreasonable to model the same person to play at the same time the role being an employee and the role being unemployed. Simple constraints can be defined automatically in the role with deactivation of other currently active roles on the object, when the role is activated. The opposite applies as well; if a role is deactivated, it can deactivate other roles or it can activate other roles previously deactivated. The call of the methods on inactive roles is not possible (Graversen and Beyer, 2002).

The mission of role modeling is to reduce complexity at the time of the large scale design; such as complexity due to the size of the tasks of design. This is done by supporting the separation of the concerns and reusable design (Atil *et al*., 2004, 2005).

PROPOSED APPROACH

In our approach, a software process can be simple or complex, i.e., compound of a set of sub-processes which cooperate in order to achieve the same objective. So, they are related to each other in different way: Serving, using and communicating with each other. From the way in which they treat one another, processes have different perspectives of each other. These perspectives define the role that a process may play towards another. A role is formed as a set of behaviors of the process. Different roles exist for different purpose and the roles played by a process may change over time.

In this approach, process's activities can exist in many versions and can be organized during time in many different manners. Each acceptable organization of activities defines process behavior (a methodology of its working). In this way, behavior presents the associated process as a states machine (McGregor and Dyer, 1993). The process's behavior according to a determined objective defines its role and the role is then a sensible series of activities.

The modeling of such process is essentially based on the definition of the set of composing sub-processes, of dependencies between its activities and of roles that it offers (Fig. 1).

A formal process define a generic software process model, offering some alternatives, from which, we can generate specifics software processes (Real processes). The generation is done according to an appropriate behavior and allows then the solving of a particular problem.

In the Interface part, we identify the list of roles that the process offers. Each role defines a perspective that a process may play towards another.

Process <Process Name>;
Interface <Interface description: Identification of roles>
Sub-Processes <Definition of the set of composing sub-processes>
Functional Dependencies
<Definition of functional dependencies>
Organizational Dependencies
<Definition of the set of roles>
End <Process Name>

Fig. 1: Definition of a formal process

In the Sub-processes part, we present the list of sub processes that compose our process.

The functional dependencies part regroups all data flux and control flux dependencies. They must be verified every time and are explicitly defined by the relation function that has a changeable semantic. The function dependency expresses that a set of target activities TA depends on an optional set of initial activities IA under the optional constraint Ctr. When all activities of IA are executed, activities of TA could be executed under the constraint Ctr.

Formally, this dependency is defined with: [IA] [Ctr] → TA, were Ctr is defined with <condition; value; sense>. The condition attribute defines conditions that must be satisfying in order that dependency being valid. Value attribute defines the data flux required by this dependency. Finally, sense attribute defines the semantic of dependency, which can be repetition (*), implication (/\), exclusion (¬), equivalence (~), instantiation (∋), etc.

Finally, the Organizational dependencies part allows the modeling of behaviors of the software process. We describe each role of the process which is an organization of activities during time (with Synchronous and Alternation dependencies) as well as their hierarchical organization (with Aggregation dependency).

Synchronous: This dependency allows ordering activities in time. It's expressed with: Syn a1, a2, ..., an Endsyn. Activities none implicated in a Syn dependency may be executed in any order.

Alternation: It's a dependency, which allows establishing a nil order between a set of activities. These activities are then alternated and could constitute a varying activity. By nil order, we imply that only one of concerned activities can be executed. This activity will be determined dynamically according to explicit or deduced contextual knowledge. It's defined with: Alt a1, a2, ..., an Endalt. Only one activity ai (i = 1, n) must be executed and all the others will be ignored.

Aggregation: It allows constructing a complex activity with hierarchical composition (designed by an identifier) of different activities. If the composition is designed with an identifier, this last will indicate the resulting activity. Such dependency will be expressed with: {IA1, IA2, ..., IAn} <∅;∅;U> → TA, were TA is the identifier of the resulting activity. None designed composition don't construct a complex activity.

Owing to such model, we can define a formal process that can be independent of any problem and from which we can generate a real process as an instance that can take part in development of specific software processes.

Therefore, according to need, we can define or modify different methodologies (behaviors). The instance's methodology, generated from a process, imposes to this last a controlled behavior that can be automated. This vision offers a considerable benefit for software processes modeling.

CONCLUSIONS

Modeling software processes is a difficult and expensive task. It's confirmed by diversity of software processes modeling approaches presented in the literature and which are however, not satisfactory.

The goal of this study is to present a new and natural modeling approach of software processes based on a textual description. This approach is based on the concept of role which define a perspective that a process may play towards another. It is formed as a set of behaviors of the process. Different roles exist for different purpose and the roles played by a process may change over time.

We note that benefit of our approach is in the methodology of the process which is more explicit, more formal and especially well structured. We have then the possibility of formal verification of methodologies and reasoning and we can then construct complex methodologies with a modular manner by reusing composing process's methodologies.

REFERENCES

Atil, F., S. Ghoul, D. Meslati and N. Bounour, 2004. Modeling Software process using roles. 17th Intl. Conf. Software Sys. Eng. Applications (ICSSEA) Paris.

Atil, F. *et al.*, 2005. Role based software process modeling. ISPS'2005, 7th Intl. Symp. Programming Sys., Algiers.

Estublier, J., 2005. Software are processes too. Software Process Workshop (SPW), Bejing.

Ghoul, S., 1995. Methodological and structural aspects in software processes models. Ph.D Thesis, University of Annaba.

Graversen, K.B. and J. Beyer, 2002. Conceptual programming using roles. IT University of Copenhagen.

Horvat, R.V. *et al.*, 2000. SoPCoM model for evaluation of the software processes complexity. EuroSPI 2000, Copenhagen, Denmark.

McGregor, J.D. and D.M. Dyer, 1993. Inheritance and state machines. ACM/SIGSOFT, pp: 61-69.

Rueher, M. and C. Michel, 1990. Using objects evolution for software processes representation. Proc. 22nd Annu. Hawaii Intl. Conf. Sys. Sci., 2: 121-130.