Computación y Sistemas Vol. 1 No. 4 pp. 228-238
© 1998, CIC - IPN. *ISSN 1405-5546* Impreso en México

Software Process Modeled with Objects: Static View

Hanna Oktaba
Universidad Nacional Autónoma de México
IIMAS Posgrado en Ciencia e Ingeniería de la computación
Circuito Escolar, Ciudad Universitaria
México, D.F. 01000
e-mail: oktaba@servidor.unam.mx

Guadalupe Ibargüengoitia González
Universidad Nacional Autónoma de México
Facultad de Ciencias, Departamento de Matemáticas,
Circuito Escolar, Ciudad Universitaria,
México, D.F. 01000
e-mail: gig@hp.fciencias.unam.mx

Article received on January, 1998; accepted on March, 1998.

Abstract

This paper presents an attempt to exploring the modeling with objects as a way of structuring and systematize the fundamental concepts of the software process (SP) with the basic objective of understanding them better.

The classes and relationships between them are used to model the static view of the SP. The relationships of aggregation and association express the structure of dependences between the basic concepts of the SP, which are: the phases, the activities, the products, the roles and the agents. The relationship of inheritance is used to express the different specializations of the previous concepts.

The general model applies to the detailed description of the phase of analysis with the purpose of understanding their basic activities, the involved roles, the generated products and the relationships between them.

Key words:

Software Engineering, software process, phase, activity, product, role, model of objects.

Introduction

The Software Engineering gained importance as a discipline when the software systems became so complex that it was no longer possible to develop those of handmade way by one person. During the last 30 years different methods and technics have been developed in order to facilitate the development of systems. However, it hardly make scarce 10 years, the interest for the Software Process(SP) has arisen, which Fuggetta [Fuggetta, 1995] defines as:

" It is a group of people, organization structures , rules, policies, activities and procedures, components of software, methodologies and tools used or created especificaly for conceptualize, develop, offer a service, innovate and extend a product of software."

As it becomes clear from this definition, the software development process is a very complex activity that involves interdisciplinary aspects.

The SP most relevant characteristics, according to Huff [Huff, 1996], are the following:

Concurrency and distribution - the SP is a group of tasks (activities) carried out in a concurrent way , and in general, distributed between several people.

Non determinism and insecurity - the development is non deterministic, because each error found could cause that it is necessary to re-do a part of the work and therefore, postpone or cancel other activities.

Evolution and changes - the process of software should be improved constantly in order to obtain better results in the use of resources and in the quality of the systems produced.

These characteristics underline one of the important areas of research in the Software Engineering that is the modeling of SP.

The primary objectives of modeling SP are [Huff, 1996] the following: understanding the software process, improving its acting (improvement), and carrying out their execution (enactment).

In the literature there exist works modeling the SP with several focuses. In [Huff, 1996], they are classified as follows:

Non executable paradigm- textual or graphical (e.g. IDEF0).

Paradigm based on states- states automata, Petri nets, formal grammars.

Paradigm based on rules- expert systems, Prolog, systems of planning.

Imperative paradigm - Ada like specification language.

The purpose of this work is to use the object-oriented modeling to understand easier the complexity of the static structure of the SP, and to teach it to students and groups of developers trained in the OO technology. The idea of using the techniques of software development for the description of the process is not new [Osterweil, 1987], but it seems that the use of object-oriented modeling has not been explored thoroughly.

The motivation in order to use the concept of class to model the static view of the SP is the following:

- The relationship of generalization/ specialization between classes allows the gradual modeling of the software process, classifying the concepts from a very general level to more specific ones. This graduation helps in the understanding of the model.

- The relationships of aggregation and association offer a mechanism to describe the dependences between the different concepts that one is modeling.

Definition of the software process and their components

The software process is a composition of phases, activities, artifacts and resources (including the humans).

The phases, better known as the life-cycle phases of the software, constitute significant steps of the software process. Each phase contains several activities that are carried out with the purpose of generating an important product (artifact), like for example, the document of requirements specification or the document of design. Examples of phases are: analysis, design, code, installation, etc.

The activities (or tasks) are the key pieces of the process. They define the actions (procedures) that must be carried out in a given moment of the development. In general, an activity requires one or more input artifacts and generates the output artifact(s). An activity also requires resources, particularly human resources, that associates via the concept of role. The availability of the input artifacts and of the resources imposes certain temporary order in the execution of the activities.

The artifacts that are the inputs and outputs of the activities, could be a set of very varied documents, for example, diagrams of design, code, plans of tests, reports, user's manuals; as well as sets of documents of several types.

The most important resources in the modeling of SP are those that are represented by roles, which are able to carry out the activities of the process. The roles are assigned to agents in turns. An agent is a human being or an automatic tool that executes an activity. The resources could also be: working hours, money, computer laboratories, etc. For the purposes of this paper, we will only consider the roles and the human agents as resources.

Abstract model of the software process

The object-oriented modeling of the SP begins with the identification of the basic classes.

Software Process- it is the class representing the general concept that one is modeling.

Phase- it is the abstract class representing a phase of the software life-cycle.

Activity- it is the abstract class representing the activities of the process.

Artifact- it is the abstract class representing several products that are generated and exchanged.

Role- it is the abstract class representing the several roles associated to the activities.

Agent- it is the person or tool, that plays a certain role in the process.

The general definitions enunciated in the previous section introduce the first relationships of association and aggregation between these classes. The class Process Software is made up of several instances of the Phase class. This class, in turn, has several objects of the Activity class attached to it. The objects of this class could be added to other objects of the same class, because an activity could be made up of several auxiliary activities. The Activity class also adds at least an input artifact and an output one, represented by the instances of the Artifact class. The objects that represent artifacts could also be made up of other artifacts. It also exists certain association between

the Role class and the instances of the Activity class and, finally, there exists an association between the Agent class and the possible instances of several Role(s) that an agent will carry out. The Fig.I presents the diagram of classes with the relationships mentioned in the UML notation [UML, 1997].

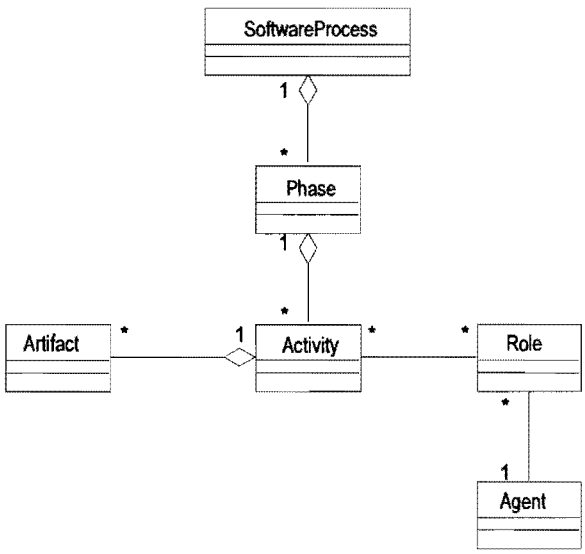

Figure 1: Class diagram software process.

Specialization of phases

Several models of the software development life-cycles exist. The most outstanding examples are: the waterfall model [Royce, 1970; Böhem, 1981], the spiral model [Böhem,. 1988] and the iterative and incremental model for the object-oriented development [Booch, 1994]. All these models contain the following phases, although they sometimes use different terminology:

Analysis- it represents the activities that lead to the understanding and documentation of the requirements of the software system and the modeling of the problem domain.

The analysis includes the customer validation of the documents that contain the agreement on the scope of the system. *Design- its objective is to do the maping of the requirements and the problem domain model for the computer environment that will make possible the implementation.*

It contemplates the architectural and the detailed design of the components. The design activities verify also that all the requirements specified in the analysis are covered.

Code and Tests- it is the implementation based on the design using programming environments.

It contemplates the unitary and integration tests in order to verify the correspondence between the designe and the implementation.

Installation- it refers to the system delivery to the client, and its setting in operation in the real environment of execution.

It includes the customer and the final users validation of the system.

Maintenance- it includes the corrections, modifications and extensions to the system after liberation.

It includes the configuration management and the version control.Upon analyzing the activities of maintenance, in any of their modalities of correction, modification and extension, it is observed that these correspond to the usual activities of the development process of a new system with the difference that the input artifact belongs to a system already existing and, therefore, the procedures in order to carry out these activities will have a particular definition. The configuration management and the versions control, that distinguish the maintenance phase, should in fact be defined from the beginning of a new system development. Consequently, we decided not to include the maintenance as a different phase in the software process by considering that its activities correspond to the other phases.

The specialization of the Phase class, according to the classification presented here, sample in Fig.2.

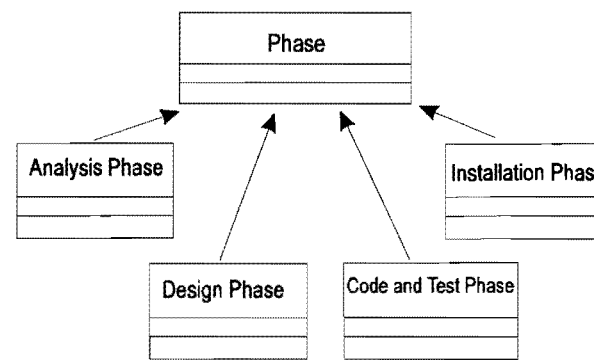

Figure 2: Phases specialization.

Specialization of activities

The activities of the software process are divided in four basic groups: production, control, technology and communication (Fig. 3).

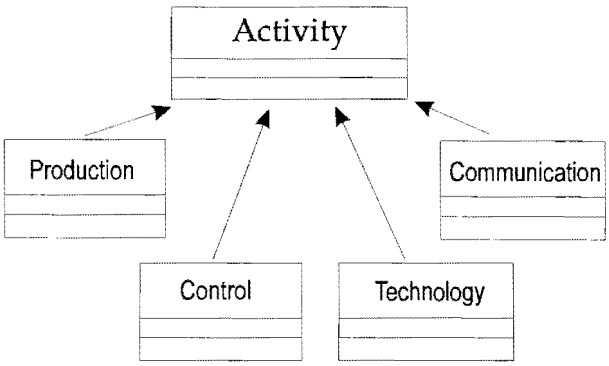

Figure 3: Activity specialization.

Production

The activities of production are those that conduct directly to the construction of the software system in question. Some examples of the productive activities are: the analysis, the design and the code. Their input artifacts are respectively, the customer requirements for the analysis, the analysis documents for the design and the design documents for the code. The output artifacts are also obvious. Their actions could be described according to the method or the particular technique that is used. For example, for the process based on the object-oriented technology the use of the strategies and patterns of Coad [Coad *et al.*, 1995] or the use cases of Jacobson [Jacobson *et al.*, 1992] could be chosen for the analysis; for the design, the techniques of Rumbaugh [Rumbaugh *et al.*, 1991] or Booch [Booch, 1994]. Both activities could be documented evenly with the UML notation. Some of the object-oriented languages like C+, Smalltalk or Java could be chosen for the code.

The documentation of work products, including the code documentation, is another production activity that is time consuming and that requires discipline. Its importance for the development team during all the phases of the SP is clear. A similar degree of importance, for the client and the final user, have the manuals of the system whose documentation should be also included as a legitimate activity of production.

To the group of production activities the construction of prototypes (prototyping) is added. The prototypes serve, in general, to clarify any doubts on the requirements definition or to confirm the design proposals. The prototypes are almost always discarded and do not form part of the final product. However, we consider appropriate to include the activity of prototype construction as an activity of production because it requires the same process as the production of a complex system, only that to a minor scale. One could also consider the construction of demonstration programs (demos) as the production of prototypes, although starting from a system that already exists. The Fig.4 shows the first level of specialization of the Production class.

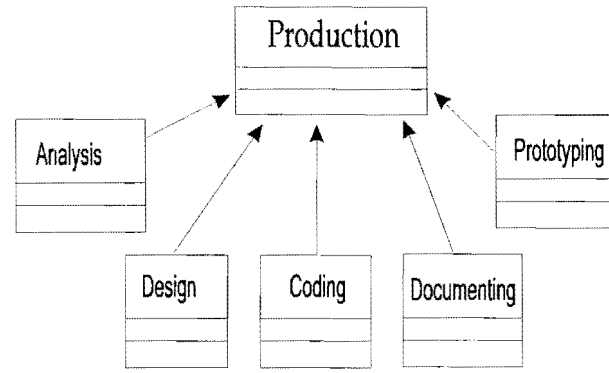

Figure: 4 Production activities specialization.

Control

The control activities are those that help to verify the state of the process and of the products generated by the other activities. Therefore, they specialize in two types: those related to the process control and those related to the product control (Fig.5).

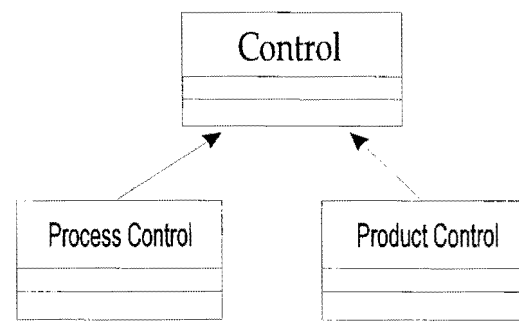

Figure 5: Basic specialization of control activities.

Process control

The activities devoted to process control are divided in two types: those carried out from the point of view of the process management and those carried out from the personal perspective of each one of the agents that participate in the process. A few examples of the management activities of process control are the: methodology selection, planning, progress monitoring, modification of the process and post

mortem evaluation upon finishing the project. An example of personal control activity of the process we mention that of the time record that takes an agent to carry out an activity. The latter is related to the explicit introduction of the time measurement in the SP. Due to it, one could carry out the quantitative evaluation of time dedicated to the development of the project and to each one of the several activities of the process. Fig.6 shows the specializations of the Process Control class.

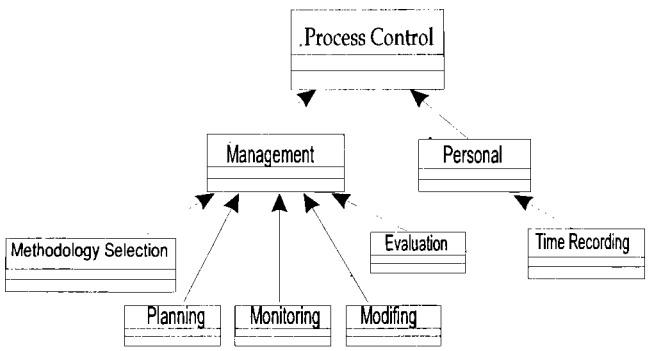

Figure 6: Process control activities specialization.

Product control

Another type of control activity is the one which supervise the product (product control). The most important activities are those devoted to find and fix defects (defect control). To this type of activities belong those that correspond to the search of defects made by the human agents in the products of analysis, design and the code . These activities are: the reviews carried out by oneself, the inspections made by colleagues, the verifications made by the development team, and the validations carried out by the client. Also, in the group devoted to the activity of searching defects, are included the compilation and the testing, which are carried out with the support of automated tools. The compilation without errors is, of course, considered as part of the activity of code.

The activity of defect fixing is the one that removes the errors found in the products. When it is code defect fixing the activity is known as debugging. This activity is included as a specialization corresponding to the defect control.

Configuration

A very important aspect of product control is the configuration management that includes, among other things, the version control. A configuration defines the final product components that are given to the client. The changes made to the components of a configuration should be properly controlled, documented and eventually reflected as the components' version changes.

Measures

Another specialization of the product control activities is the one that refers to register measures. This in turn specializes in the activity dedicated to the registration of the products founded and fixed defects (defect record), and another activity which refers to the counting of product units (unit record). For example, the number of classes in the diagrams of analysis or the number of lines of code. These two activities, defects and units record, belong to activities that generate basic measures for the quantitative evaluation of the quality of the product and of the process.

Standards definition

The standards definition of the work products permit, upon adopting these by the members of the development team, the easy identification and the understanding of the same. For this reason, we include this activity as another specialization of the product control. Fig.7 shows the several specializations of the Product Control class.

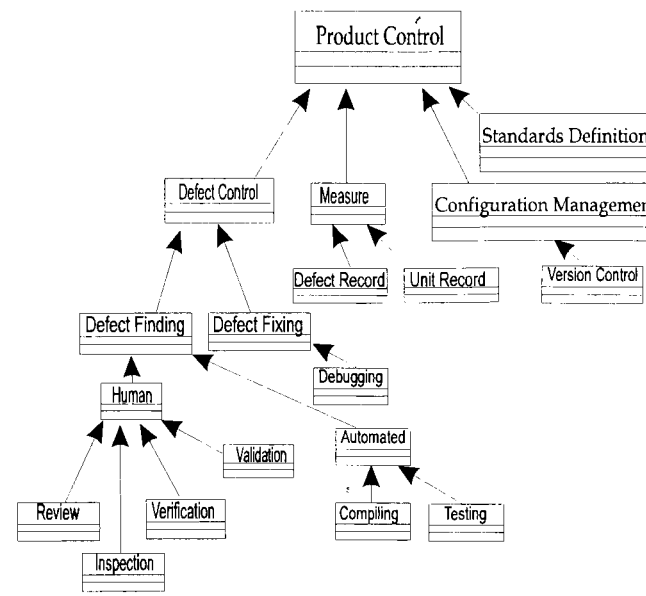

Figure 7: Product control activities specialization.

Technology

The activities of technological type are the software and hardware evaluation, the staff training on the methodology or tools and those that facilitate the reuse. The latter could include the creation, management and access to libraries of reusable components (Fig.8).

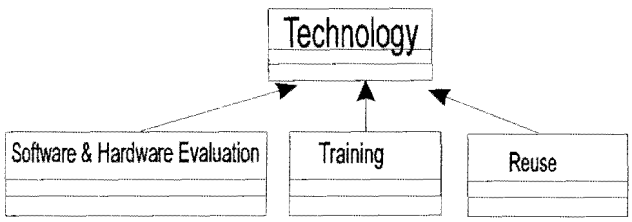

Figure 8: Technology activities specialization.

Communication

The communication with the customer and that carried out between the members of the developement team, is a very important aspect of the software process. In great measure the succes of the project depends on the efficient and on time communication. The activities of communication could be basicaly classified in meetings and those that are carried out through the exchange of documents or products. The meetings require the attention of more than a person at the same time (synchronous communication), while during the exchange of documents or products it is not necessary (asyncronous communication). In both cases the means of communication could be physical (room, mail) or electronic (videoconference, e-mail). Fig.9 represents the basic specialization of the communication activities.

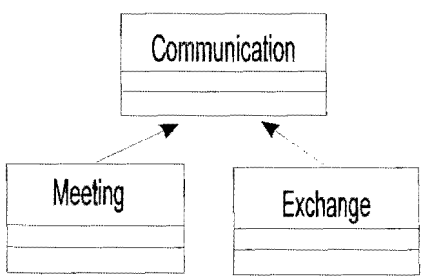

Figure 9: Communication activities specialization.

Specialization of roles

The classification of roles reflects, in a way, the activities classification. On one hand, roles are clearly related to the production activities like, for example, the analyst, the designer, the programmer. On the other hand, the roles of process control are played by the manager or project leader. The roles related to the product control could be carried out by the tester of code and the integrator of the system.

Recently, roles of technological type have arisen, like the expert in human-machine interfaces, in data bases, in computer nets, and all kinds of roles related to the reusability, like: the evaluator of the reusability at company level, the manager of the reusable component libraries and the reusable components supplier. As the technological part one could also add the roles related with the training in the development techniques and the tools.

With regard to the communication, projects have been mentioned in which it is useful to have people whose role is that of being used as bridge between the technical team and the customer, or the technical team and the manager of the project or simply between several subteams of the same project [Coplien, 95]. This type of roles could help reducing the amount of communications between the members of the project, and any misunderstandings (which can lead to redoing the work) due to the lack of opportune information.

Finaly, we included the role of customer that plays an important part in some activities of the SP, like the definition of requirements or the validation of the products.

Fig. 10 shows the hierarchy of classes that represents the specialization of roles.

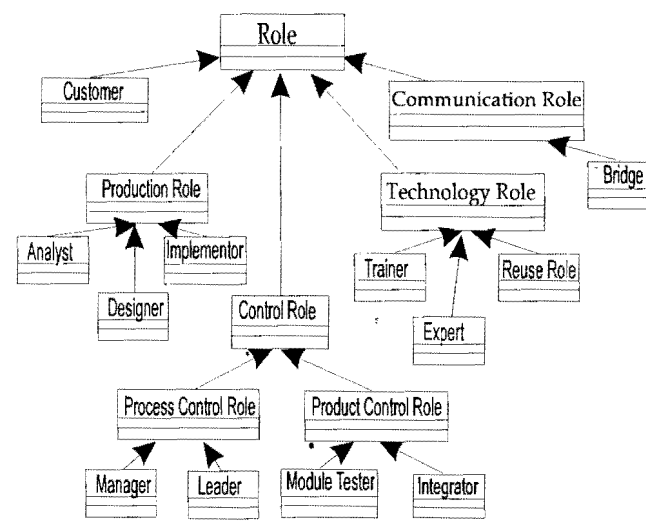

Figure 10: Roles specialization.

Artifacts specialization

The activities need the input artifacts and, in general, they generate the output artifacts. It is not surprising then, that their classification reflects the classification of the activities in great measure.

In order to begin the activities of the SP we need a document that defines the basic requirements of the client (customer request). This document could be part of a contract with a customer, or it could be an initial description of the system made by the developement company that wants to take out the product to the general market.

The artifacts, that are generated and circulate while the different activities of production are carried out, are the documents of analysis, design and code.

The documents generated for control purposes could be specialized in those which support the process control and those used for the product control. Examples of documents for the process control are: the project plan, the schedule, the checklists, the time recording log and the document that summarizes the information on the project. In turn, examples of documents for the product control are: the list of verification (checklist) in order to make revisions of products, the test plan, the registration of units of the product and the registration of the defects of the product.

Among the documents of technological type we find the methodology definitions, the standard descriptions (for example, the standard of code) or the descriptions of reusable components.

Moreover, there also exist documents generated by the communication activities like, for example, the electronic mail messages or paper reports of the meetings. Fig.11 represents the class hierarchy that models the artifacts.

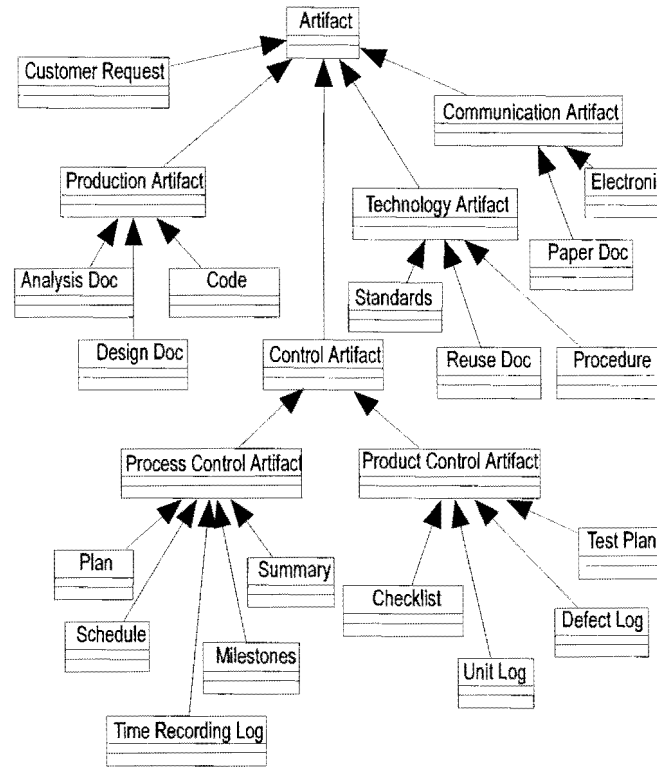

Figure 11: Artifacts specialization.

Agents specialization

In general, there are two types of agents: humans and tools. Fig. 12 offers the corresponding diagram. For the purpose of this work, only human agents will be considered.

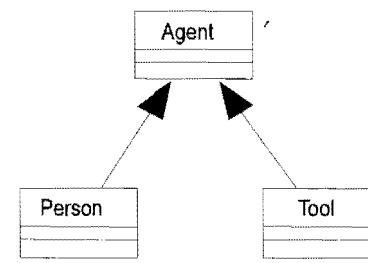

Figure 12: Agent specialization.

Static model of the analysis phase

The object model of software process shown in the previous section uses the relationship of inheritance in order to classify the basic concepts involved. This classification does not seek to be neither complete nor exhaustive. Its purpose is only to show the way one could understand the complexity of the software process step by step.

As a case of study we present the object model which helps to understand and deepen in the anlysis phase specializing the activities, the roles and the artifacts involved.

Analysis phase activities

The basic activities of the analysis phase are those that lead to the definition of the requirements of a system and of the abstract modeling of the problem domain. In order to carry out these activities in an efficient way we also include the activities of process control that allow the resources planning, the monitoring and the evaluation of the plan execution.

The activities of product control are also incorporated. The activities of this type are the internal revision and the validation made in conjuction with the client.

The phase of analysis requires also communication activities between the client, the analyst and the leader of the project in order to carry out the basic activities of production and control.

Fig.13 shows the relationship of aggregation that occurs between the AnalysisPhase class and the classes that model the activities that have been mentioned before. The general model of activities (Fig.3) also contemplates the activities of technological type. In order to simplify the modeling of the phase of analysis, it was supposed that the agents that will assume the roles for this phase do not require an additional training in technological aspects.

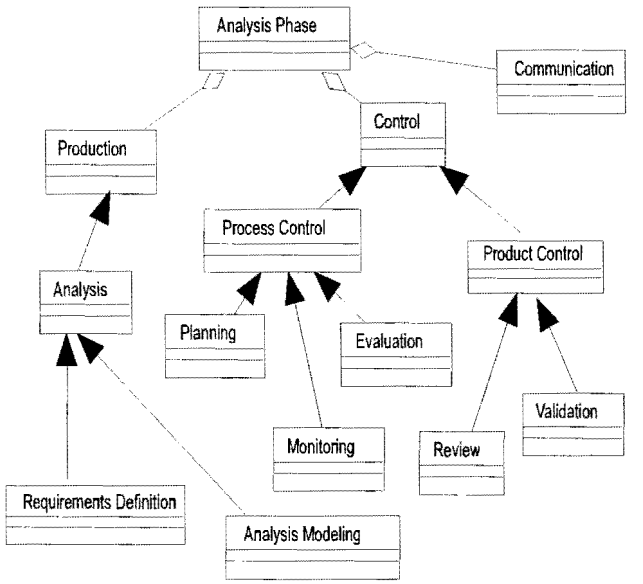

Figure 13: Analysis phase relation with activities.

Roles of the analysis phase

The minimum roles involved in the analysis phase are: the project leader, the analyst and the customer. The project leader's reponsability is to carry out the activities of process control. Fig.14 models the association relationship between the class Manager and the activities of the analysis phase that correspond to it.

The analyst rôle consists executing the activities of production like the definition of requirements and the creation of the abstract model of the system. The analyst is also responsible to do the revisions of their products in order to guarantee their better quality. Fig.15 shows the association relationship between the corresponding activities and the analyst.

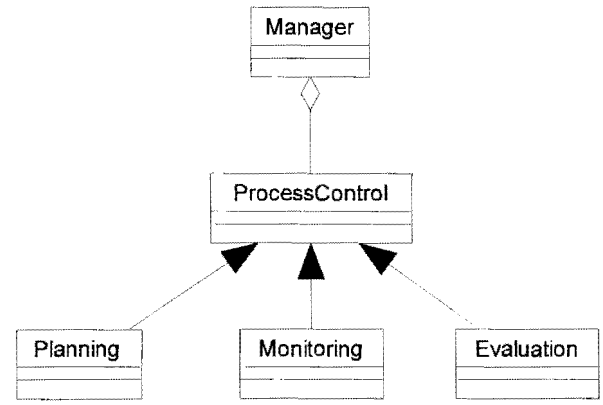

Figure 14: Manager relation with the control process activities.

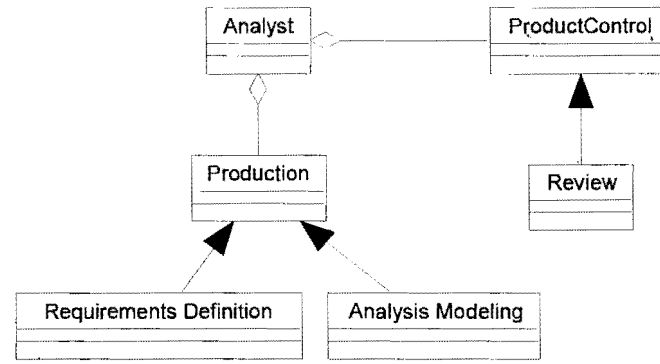

Figure 15: Analyst relation with the production control acitvities.

It corresponds to the customer to do the validation of the analysis artifacts in order to discover any possible defects or misunderstandings. Fig.16 shows the relationship between the Customer class and the validation activity.

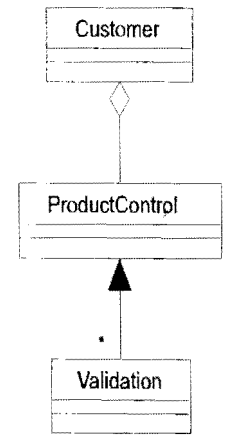

Figure 16: Customer relation with the product control activity.

Documents and products of the analysis phase

The artifacts that are managed during the phase of analysis are shown in Fig.17. The initial document is the definition of the basic requests made by the client, which, in general, are specified before we start the project. The central products of this phase are: the detailed specification of the requirements and the document of the abstract model of the system. The last could be made up of several documents whose content depends on the method of modeling selected for the analysis phase. The analysis phase control documents are the plan and final summary.

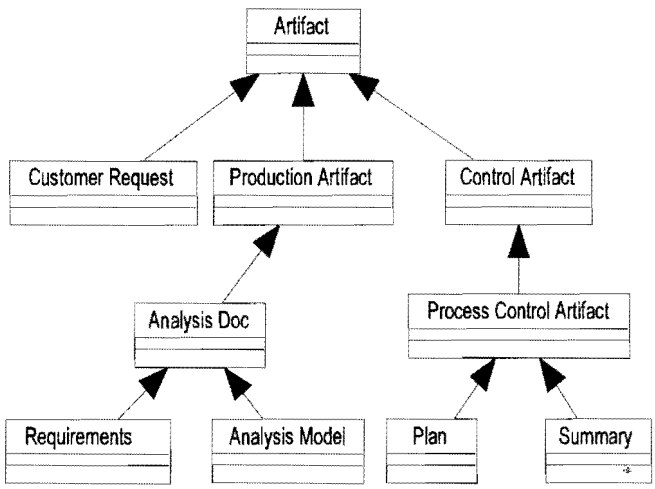

Figure 17: Analysis artifacts hierarchy.

Relationship between the input and output artifacts and the analysis activities

The activity of requirements definition converts the initial requests of the customer in a more precise specification. This document is used as an input for the modeling activity which, according to the applied method, leads to the construction of one or more models of the problem domain. Figs.18 and 19 relate the input and output artifacts to the production activities of the analysis phase.

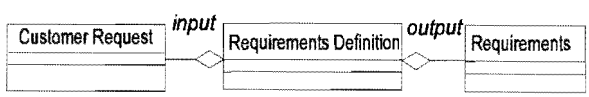

Figure 18: Requirements definition activity and its relation with the input / output artifacts.

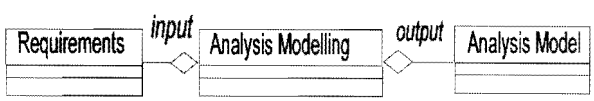

Figure 19: Analysis modeling activity relation with the input/output artifacts.

The revision activities done by the analyst and those of validation, carried out by the client take as the input artifacts the requirements documents or the analysis models, and they return the same documents with the modifications request or, simply, approved. In the first case, the returned documents serve as the input to the corresponding production activities and the process is repeated until the revision and the validation are approbatory. Figs.20 and 21 relate the corresponding classes.

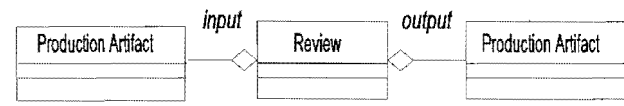

Figure 20: Review activity relation with the input/output artivacts.

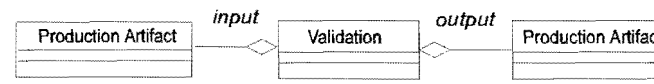

Figure 21: Validation activity and its relation with the input/output artifacts.

The planning activity, that initializes the phase of analysis, takes, as a starting point, the customer requests and generates the plan of activities with a schedule and the responsabilities assignation. The monitoring activities take the plan and the products of analysis as an input and conclude with the plan in which the progress or a modified plan is registered. The final evaluation of the phase of analysis generates a summary of the resources, times, costs, etc., of this phase. Fig.22 shows the aggregation relationship between the corresponding classes for the planning activity. The monitoring and the final evaluation diagrams are similar.

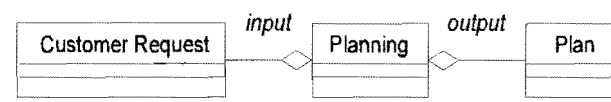

Figure 22: Planning activity and its relation with the input/output artifacts.

Communication activities

In general, the communication activities such as meetings or the exchange of documents (Fig. 9), do not have a great importance and are considered as something implicit. However, the communication between the agents involved through their roles in the system development has a crucial importance for the success of the team work. In the case of the analysis phase modeling we consider important to introduce the explicit communication activities which help to carry out the activities of another type.

Communication between the client and the analyst

In the analysis phase there exist at least two activities that require the communication between the customer and the analyst. The first one is during the requirements specification, which is almost imposible without direct meetings between both parts. Fig.23 models the corresponding relationships between the classes.

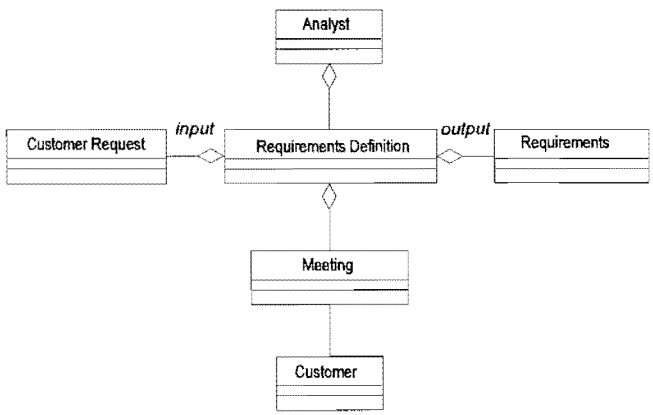

Figure 23: Meeting activity and its relation with the requirements defintition.

Another occasion for the communication between customer and analyst occurs during the validation of the analysis artifacts. In this case, it is enough to carry out the communication through the documents exchange (Fig.24).

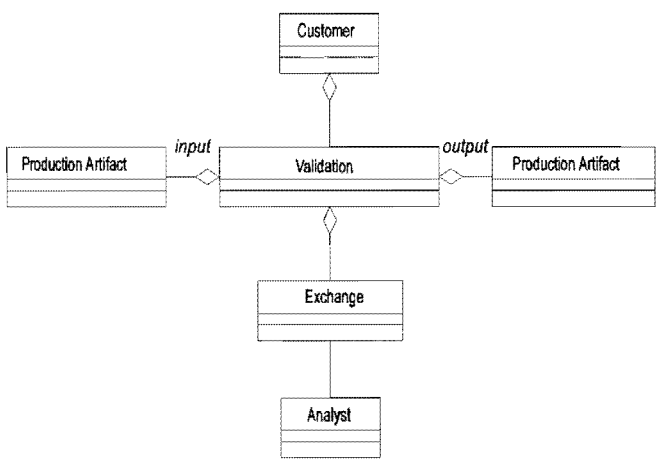

Figure 24: Exchange activity and its relation to validation.

Communication between the project leader and the analyst

Good communication between the members of the development team is very important. Particularly, in the analysis phase, in order to make the planning, the monitoring and the final evaluation the project leader needs to communicate with the analyst. The form of communication could vary. It does not matter if it takes place through meetings or documents exchange, the problem is that it should be effective and opportune. Fig. 25 models the relationships between the corresponding classes in the case of the planning activity. Other activities of process control are modeled in a similar way.

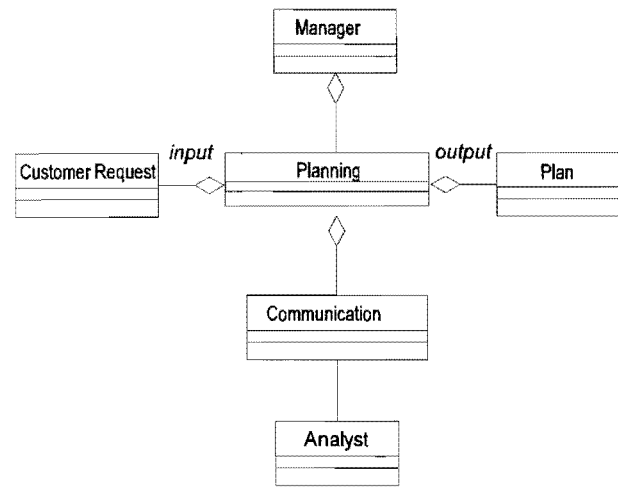

Figure 25: Communtication activity and its relation to the planning activity.

Conclusions and future works

This paper presents an attempt to exploring the object-oriented modeling as a form of structuring and systematize the fundamental concepts of the software processes. The basic goal is to understand them better.

The abstracts classes and the relationships between them were used to model the general view of the SP. The relationship of inheritance was used in order to express the different specializations of the basic concepts as activities, artifacts, roles and agents.

The classification shown here is not complete. One could create new specializations according to the type of activities that one wants to include. What we have been able to verify is that the relationship of inheritance between classes is a good vehicle to clssify the information in this respect.

In the second part, we developed an example applying the general model to the analysis phase specifying the static structure of the basic activities of production, control and communication, the involved roles and the generated artifacts. The modeling through relationships of specialization, aggregation and association of classes allowed the gradual expression of the dependences between several elements of the process. It was also easier to deepen in details.

We consider that the model is general and simple enough to be used as a guide for teaching in the area of Software Engineering and for the training of development teams. We have distinguished between the different roles, different activities of production, process control, product control, communication, and different description of artifacts. We hope we are offering a model complete enough, so that it can be useful in the practice in order to define concrete models of processes.

The future work is to continue the static modeling of the other basic phases of the software process. The general purpose is to define the minimum set of roles, activities and artifacts that cover the complete process.

On the other hand, it could be interesting to explore the dynamic views (interaction diagrams and state-transition diagrams) of the object model in order to include the sequence aspects of the activities execution and their temporary dependences.

References

Böhem, B., *Software Engineering Economics*, Prentice Hall, 1981.

Böhem, B., "A Spiral Model of Software Development and Enhancement", en Thayer, Richard, ed. *Software Engineering Project Management*, IEEE Computer Society Tutorial, Catalog Number EH0263-4, 1988, pp.118-127.

Booch, G., *Object-Oriented Analysis and Design, with Applications* 2nd edn. Redwood City, CA: The Benjamin/ Cummins Publishing Company, 1994.

de Champeaux, D., *Object-Oriented Development Process and Metrics*, Prentice Hall, 1997.

Coad, P. with D. North and M. Mayfield, *Object Models - Strategies, Patterns and Applications*, Yourdon Press, Prentice Hall, 1995.

Coplien, J. O., "A Generative Development-Process Pattern Language", en *Pattern Languages of Program Design*, Reading, MA: Addison Wesley, 1995, pp.183-237.

Finkelstein, A., J. Kramer and B. Nuseibeh (Eds.), *Software Process Modelling and Technology*, , Research Studies Press, John Wiley & Sons, 1994.

Fuggetta, A., *Il Processo Software, Aspetti strategici e organizzativi*, il Cardo editore in Venezia, 1995.

Gamma, E., R. Helm, R. Johnson and J. Vlissides, *Design Patterns - Elements of Reusable Object-Oriented Software*, Reading MA: Addison-Wesley, 1994.

Huff, K. E., "Software Process Modelling", in *Software Process*, Eds. A. Fuggetta and A. Wolf, *Trends in Software 4*, J.Wiley&Sons Ltd., 1966, pp.1-24.

Jacobson, I., M. Chisterson, P. Jonsson and G. Overgaard, *Object-Oriented Software Engineering - A use case driven approach*, Addison-Wesley, 1992.

Lehman, M. M., "Process Models, Process Programs, Programming Support", *Proceedings of 9th International Conference of Software Engineering*, March 30-April 2, 1997, Monterey, Cal., USA, pp.14-16.

Osterweil, L., "Software Processes Are Software Too", *Proceedings of 9th International Conference of Software Engineering*, March 30-April 2, 1997, Monterey, Cal., USA, pp.2-13.

Royce, W. W., "Managing the Development of Large Software Systems", en Thayer, Richard, ed. *Software Engineering Project Management*, IEEE Computer Society Tutorial, Catalog Number EH0263-4, 118-127, 1988. Reimpreso de Proceedings of IEEE WESCON, 1970, pp.1-9.

Rumbaugh, J., M. Blaha, W. Remerlani, F. Eddy and W. Lorensen, Englewood Cliffs, NJ: Prentice Hall, 1991.

UML: Unified Modeling Language, Rational Software Corporation, Version 1.0, 13 January 1997.

Hanna Oktaba *was awarded a PhD degree in Computer Science, at the University of Warsaw, in Poland, in 1981. From 1974 to 1983 she had an assistantship and then a lectureship in the Computer Science Faculty of the University of Warsaw. Since 1993 and to the present day, she gives lectures on Computer Science to Master students of the IIMAS, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (Research Institute of Applied Mathematics and Systems), at the Universidad Nacional Autónoma de México (UNAM). From March 1990 to May 1997, she coordinated the above postgraduate course. Her main areas of interest are: Object Oriented Technology, Software Engeneering and Quality Models.*

Guadalupe Ibargüengoitia G. *has got a Masters degree in Computer Science awarded by the UNAM. She lectures at the Mathematics Department of the UNAM's Science Faculty since 1975. She teaches Computer Science subjects in Mathematics and Computer Science courses, at the UNAM. She has a lectureship in the Computer Science Masters course of the IIMAS since 1993. Her main areas of interes are: Software Engeneering, Object Oriented Technology and Data Bases.*