Oktaba, Hanna; Ibargüengoitia González, Guadalupe
Software Process Modeled With Objects: Static View
Computación y Sistemas, vol. 1, núm. 4, abril-junio, 1998, pp. 228-238
Instituto Politécnico Nacional
Distrito Federal, México

Available in: http://www.redalyc.org/articulo.oa?id=61510405
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, agent, inheritance, aggregation, association.

Introduction

The Software Engineering gained importance when the software systems became so complex that it became longer possible to develop those of handmade person. During the last 30 years different methodologies have been developed in order to facilitate the development of the systems. However, it hardly make scarce 10 years ago, the Software Process (SP) has arisen, which [Fuggetta, 1995] defines as:

“... is a group of people, organization structure, policies, activities and procedures, components of the software, methodologies and tools used specifically for conceptualize, develop, offer, innovate and extend a product of software.”

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

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

Concurrency and distribution - the SP is a group of activities carried out in a concurrent way, where it is necessary to coordinate and synchronize.


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 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 other artifact(s). An activity also requires resources, particularly human resources, that associates via the concept of availability of the input artifacts and of the resources in a certain temporary order in the execution of the activity.

The artifacts that are the inputs and outputs of the process could be a set of very varied documents, for example, the plans 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 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 artifact, a tool that executes an activity. The resources could be: working hours, money, computer laboratories, etc. In the purposes of this paper, we will only consider the role of 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 the products that are generated and exchanged.
- Role - it is the abstract class representing the 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 an individual of several instances of the Phase class. This class, in turn, has several objects of the Activity class attached to it. These objects of this class could be added to other objects of the same class.
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.1 presents the diagram of classes with the relationships mentioned in the UML notation [UML, 1997].

![Class diagram software process.](image)

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; Böhm, 1981], the spiral model [Böhm, 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 mapping of the requirements and the problem domain model for the computer environment that will make possible the implementation.

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

  It contemplates the unitary and integration tests to verify the correspondence between the design and implementation.

- **Installation** - it refers to the system delivery to and its setting in operation in the real environment.

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

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

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

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

![Phases specialization.](image)

Figure 2: Phases specialization.

**Specialization of activities**
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 directly linked to the construction of demonstration programs (demonstrations) and to the production of prototypes, although starting from a technology that already exists. The Fig. 4 shows the first specialization of the Production class.

Control

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

**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**

components of a configuration should be properly documented and eventually reflected as the version changes.

**Measures**

Another specialization of the product control activity is one that refers to register measures. This in turn includes the activity dedicated to the registration of the product and fixed defects (defect record), and another one refers to the counting of product units (unit record). These activities belong to activities that generate basic means for the quantitative evaluation of the quality of the product of the process.

**Standards definition**

The standards definition of the work products is adopted these by the members of the development easy identification and the understanding of the reasons, we include this activity as another specialization of product control. Fig.7 shows the several specializations of the Product Control class.
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).

![Diagram of Technology activities specialization]

Communication

The communication with the customer and that carried out between the members of the development team, is a very important aspect of the software process. In great measure the success of the project depends on the efficient and on time communication. The activities of communication could be basically 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 (asynchronous 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.

![Diagram of Communication specialization]

Specialization of roles

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

Recently, roles of technological type have arisen: the role of expert in human-machine interfaces, in data bases, in networks, and all kinds of roles related to the reusability evaluator of the reusability at company level, the management of the reusable component libraries and the reusable component supplier. As the technological part one could also analyze roles related with the training in the development tools and the tools.

With regard to the communication, projects have been mentioned in which it is useful to have people who act as a bridge between the technical teams, the customer, or the technical team and the management project or simply between several subteams of the same project [Coplien, 95]. This type of roles could help reduce the amount of communications between the members of the project, and any misunderstandings (which can delay the work) due to the lack of opportune information.

Finally, 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 represent specialization of roles.

![Diagram of Specialization of roles]
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 development 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.

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.

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

Analysis phase activities

The basic activities of the analysis phase are those defining the requirements of a system and of modeling the problem domain. In order to carry out these activities in an efficient manner we also include the
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.](image1)

**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 responsibility 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 role consists executing the activities of production like the definition of requirements and the validation of.

![Figure 14: Manager relation with the control process.](image2)

**Figure 15: Analyst relation with the production control.**

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

![Figure 16: Customer relation with the validation.](image3)
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.

The revision activities done by the analyst, validation, carried out by the client take as the input the requirements documents or the analysis model and return the same documents with the modifications, simply, approved. In the first case, the return serve as the input to the corresponding products and the process is repeated until the revision and approval are approbatory. Figs.20 and 21 relate the class artifacts.

![Figure 20: Review activity relation with the input/output artifacts.](image)

![Figure 21: Validation activity and its relation with the input/output artifacts.](image)

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.](image)

![Figure 22: Planning activity and its relation with the input/output artifacts.](image)

Communication activities

In general, the communication activities such as the exchange of documents (Fig. 9), do not have a specific diagram and can be depicted in the flowchart.
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 impossible 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 definition.

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.

and the final evaluation the project leader could vary. It does not matter if it takes place through or documents exchange, the problem is that it is effective and opportune. Fig. 25 models the relation between the corresponding classes in the case of the activity. Other activities of process control are modeled in the similar way.

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

Conclusions and future works

This paper presents an attempt to exploring the object modeling as a form of structuring and systematizing mental concepts of the software processes. The basic tool to understand them better.

The abstract classes and the relationships between them were used to model the general view of the software processes. The relationship of inheritance was used in order to define different specializations of the basic concepts as artifacts, roles and agents.

The classification shown here is not complete. It is possible to create new specializations according to the type of element that one wants to include. What we have been able to do is that the relationship of inheritance between concepts is a good vehicle to classify the information in this respect.

In the second part, we developed an example of our general model to the analysis phase specifying the structure of the basic activities for the communication, and the validation of the results.
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


Hanna Oktaba was awarded a PhD degree in Computer Science by the University of Warsaw, in Poland, in 1981. From 1981 to 1984 she had an assistantship and then a lectureship in Computer Science Faculty of the University of Warsaw. Since 1989 to the present day, she gives lectures on Computer Science subjects of the IIMAS, Instituto Interuniversitario en Matemáticas Aplicadas (Research Institute in Mathematics and Systems), at the Universidad Nacional Autónoma de México (UNAM) since March 1990 to May 1997, she gave an above postgraduate course. Her research interest are: Object Oriented Technology, Software Engineering and Quality.

Guadalupe Ibargüengoitia G. has got a Masters degree in Computer Science awarded by the UNAM at the Mathematics Department. She teaches Computer Science at the UNAM's Science Faculty since 1997. She teaches Computer Science, Mathematics and Computer Science at the UNAM. She has a license in Computer Science Masters from the Universidad Autónoma Metropolitana.