DESIGNING AN IT SYSTEM USING THE UNIFIED RELATIONAL PROCESS

Ionel Iacob[57]
Cezar Octavian MIHĂLCESCU[58]

Abstract

This material presents the Unified Relational Process, which consists of a very comprehensive set of indications regarding the technical and organizational aspects of software systems development, focused on the analysis of system requirements and design. Specifically, RUP is a guide that shows how UML can be used to develop a informatic system.

Keywords: unified relational process, it design

1. Design of an informatic system using "Unified Process - RUP"

Starting from the essential feature of the Unified Process, that of "*practical guide for the realization of object-oriented information systems*" - and, identifying by RUP, "*the most appropriate development method for the UML modeling paradigm*" - it can be said that the RUP are a viable alternative to the limited *one-way approach,* with outstanding results in all phases of realization of an informatic system.

1.1 Features of Rational Unified Process (RUP)

Through its complex structure, *RUP integrate into the life cycle of an informatic system*, *all the fundamental features of object-oriented methodologies*, ensuring the realization of high quality, *robust and reliable IT applications* - through an *iterative approach of the work phases and an efficient correlation of the system objectives with the beneficiaries requirements*, in a *limited type horizon and with predictable budgets*.

The iterative character of *the unified process* assume a new and efficient approach to the development process of informatic systems, but also a *modern character*, innovative in terms of *task structuring and work responsibilities at subproject or subsystem level*.

The development of a project using RUP involves both *an iterative and incremental process*, by efficient decomposition on subprojects and phased sequential iterations, but above all,

[57] Lecturer PhD, Romanian-American University, Bucharest
[58] Professor PhD, Romanian-American University, Bucharest

the abandonment of voluminous working documents, in the form of written text and the implementation, in all stages of work, of models and diagrams specific to the Unified Modeling Language (UML).

1.2 Phases and iterations RUP

The life cycle of a project developed with the help of RUP involves the implementation of *four phases of work*, each stage having a *well-established final result* associated with it - depending on the proposed objectives, the system architecture, the implementation of the functionalities and the final launch - but also mandatory elements of control and analysis of feedback type, in order to establish the degree of fulfillment of the objectives (for each phase) and moving to the next stage (conditioned by the full satisfaction of the current implementation requirements).

The *four phases of work described by the RUP* are associated with them - in the process of developing informatic systems - *nine workflows within the phases* (*wokflows*), grouped into *two distinct categories*: *Primary Flows and Support Flows*.

The category of *primary flows* includes all *the basic activities that underpin the development process*, such as: Business Modeling, Requirements Specification, Analysis and Design, Implementations, Hardware Testing and Configuration (Location) - and in the category of *support flows* are integrated the *processes that allow the management of the management of all the activities carried out at the level of the project*, through: Change Management (Configuration-Modification), Project management and Ensuring the Development Environment (Work).

Synthesized, *decomposition of the life cycle of projects*, *on RUP phases and flows*, is represented in the form of the following table (Table 1):

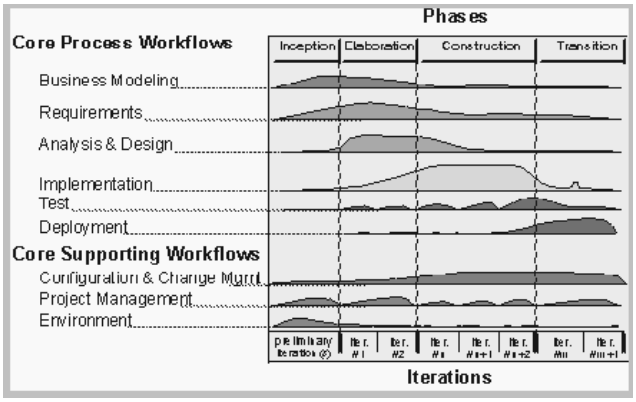

Table - Development Phases and Realization Flows RUP

In accordance with the above, *a working phase* of the *Unified Process - RUP*, is *identified by the time interval between two moments* (*critical points*) *of the project development, setting objectives, the system artifacts are made and the working directions for the next stage are identified*.

From the analysis of the work phases presented in the table, a series of information can be obtained on the *implementation of primary flows and support flows* in within the stages of realization of the new system, highlighting the importance and priority of each process, depending on the current stage of development:

- *Activity modeling* and *specification of requirements* are the primary processes with the largest share in the preliminary phases of *initiation* and *development* of the new informatic system, providing practically the necessary support for *defining the objectives* and *conditions of the project*, taking into *account the field of applicability and the specifications of the beneficiaries*;
- The processes of *system analysis* and *system design* are developed *as a unitary whole* and provide the practical foundation for the *elaboration* phase;
- Subsequently, once the *construction* and *transition* phases have taken place, the project is becoming more and more complex and advanced in terms of implementation, so that *the primary flows of hardware deployment*, *testing and configuration* now become a priority; these flows *complete the execution cycle of the informatic system* and offer solutions related to the *programming code*, *the qualitative characteristics of the product*, *the distribution and presentation of the beneficiaries*;
- In terms of *support flows*, it is observed that they are more *frequent in the final phases* of development according to the RUP (and implicitly, less used in the preliminary phases), because, with the iterative development of the project and reaching the final stages of implementation and testing of the new software product, *the management activity* (coordination, planning, management, etc.) intensifies at the system level and becomes a priority for the correct substantiation of the process of updating the changes or of efficient integration of the components made in parallel;
- The *support flow* regarding the *project management* takes place almost constantly at the level of all the implementation phases according to RUP, as it focuses on general system life cycle issues - such as: organization, management and monitoring - having, therefore, "*responsibilities*" at the level of each stage of development;
- Unlike *project management*, the *support flow* on *change management* is much more intense in the final stages of implementation, it`s overwhelming importance being directly proportional to the degree of development of the project and the complexity of the requested changes;

Software Development Process (RUP), through a *sequential and repeated approach to primary and support flows*, it is structured in four work phases: *the initiation phase*, *the elaboration phase*, *the construction phase and the transition phase*.

So:

1. **INCEPTION PHASE**

- Is the initial phase of the iterative-sequential process RUP, aiming to carry out a detailed preliminary study to identify the requirements of the future system;
- Starting from the purpose and objectives of the business, an opportunity analysis of the system is performed, which identifies: the field of the project, the solutions to be implemented to guarantee the forecasted results - by establishing efficiency criteria (for assessing success), correct assessment of potential risks and correct estimation of necessary resources;
- Considering that this stage is the first in the software development cycle, and taking into account the technical aspects mentioned above, the analysis performed in this phase will be completed by making a preliminary schedule of system execution, reflecting an overview of the project, by coordinating all four phases of work;

2. **ELABORATION PHASE**

- Within this stage we start from the *business field* (identified in the previous phase) and develop the *system architecture*, by establishing the *work plan and efficient optimization of the designed architecture*;
- Although the system architect may have different views on how to approach the system to be built, it`s *decisions must be homogeneous and define a clear and well-defined picture* of the whole system, by highlighting *the static and dynamic* aspects of the developed product;
- Decisions on the architecture of the system, within this stage, are defined on the basis of the requirements correctly specified by the beneficiaries and are implemented within the system, through *business use cases*;
- Once the use *cases are defined* and the *major risk factors of the project are eliminated*, concrete objectives can be set to be achieved for *optimizing the quality of the system architecture*, by *integrating the functionalities of the product* (reflected by the practical aspects established by the use-case specifications) in a *correct form* (system architecture), representative of the beneficiaries' requirements;

3. **CONSTRUCTION PHASE**

- In this phase, *a complex project can be structured* in several *subprojects*, each subproject representing an *iteration* in the development of the RUP, but also a permanent improvement of the functional aspects of the new system;
- Through this *iterative and incremental approach*, the risk of developing a malfunctioning system is progressively reduced, by rigorously planning the functionalities of the system, identification and elimination of risks regarding the inefficiency of the system and the implementation of error control and correction operations, through specific feedback actions;
- In the construction phase takes place the description of the requirements not specified in the development stage, the detailed design of the system, and finally, the realization and testing of the functionality of the developed system;

- *Each iteration* carried out in the construction phase, involves the implementation of *three types of fundamental works*: Resource management and process control, Development and testing of program components and Evaluation of iteration results;
- By carrying out these works, *the program components* (system artifacts) *will be established*, *location planning and iterative transition of the system will be performed and the user manual will be written*;

4. TRANSITION PHASE

- This stage of work completes the development cycle of an RUP project and has as its main objective, *delivery of the IT product to end users*;
- Depending on the degree of satisfaction of the requirements initially expressed by the beneficiaries (or their subsequent requests), during this phase, *activities will be carried out to test and improve the product made*, in order to determine the non-functional aspects of the system and to identify possible errors;
- The system version under test will be an experimental one, which will be gradually replaced by the final version, once all corrections have been made, in accordance with the results obtained in the test step;

1.3 Requirements for the development of an IT system according to the RUP

According to the implementation phases presented by RUP, for the development of an information system must be covered, in one iteration, the following primary flows: *business modeling*, *requirements specification*, *analysis and design*, *implementation and testing*.

For the actual realization of the diagrams from the design stage, the CASE – Enterprise Architect tool, developed for UML, will be used.

SPECIFICATION OF REQUIREMENTS

The *stage of specifying the requirements*, presupposes a more precise and correct definition of what the future system must ensure (according to the final objectives of the beneficiaries, specified in the previous stage of business process modeling) and involves carrying out the following groups of activities:

1. **Identification of candidate requirements**
   a) Assume, *the correct and complete establishment of the requirements of the beneficiary organization for the realization and implementation of the information system*;
   b) The (theoretical) substantiation of the imposed requirements is achieved, both directly, *through the analysis of the technical documentation*, as well as from the *information provided by the client* (*during the "system analysis" stage*), the most important situation;

2. **Organize the system using packages**
   a) *The structuring of complex systems*, in small areas, is necessary, both for a better *maneuverability of the resulting subsystems*, especially for *the detailed study of the analysis and design stages of the identified (sub) activities*;

   b) The basic idea of *organizing large software systems into small subsystems*, *structured on fields of analysis*, consists in *establishing units of behavior in the physical system*, by identifying a part of the subsystem specification (consisting of *operations*, *use cases*, etc.) and a *part of the implementation of the system* (consisting of the *definition of classes, relationships and restrictions that ensure their implementation*) - the links between the elements of the two parties being defined through a *set of collaborations*;
   c) Once these specifications are met - through the rules, concepts, and restrictions imposed by *Unified Modeling Language* (UML) diagrams - it can be considered that a *first stage in the realization of the computer system has been completed* (*successfully!*);
   d) For *structuring information systems into smaller subsystems and areas of analysis*, the UML language introduces the notion of *package* - *as an entity for grouping elements with the same design and design specification, in order to divide the system into areas*: from the lowest level of approach of the new system (with the *definition of the participating classes and objects and the correct establishment of the interactions between the actors and the system, by defining use cases*), up to the highest level (*the stage where the actual structuring of the system by fields of analysis and the grouping of object classes in packages is performed*);

2. Designing the quality control system for bearing production - using the CASE tool "Enterprise Architect"

According to the requirements of the *Rational Unified Process* (RUP), for the realization of an information system, the following primary processes must be completed: *requirements identification*, *analysis*, *design*, *implementation and testing*.

Next, *the requirements identification activities* will be exemplified, *analysis and design* in the case of "*automated bearing quality control system*".

IDENTIFICATION OF REQUIREMENTS

The main activities to be completed in this stage are:

a) Identification of candidate requirements;
b) System structuring (using packages);
c) Deepening the understanding of the context;
d) Identifying functional requirements;

e) Identifying non-functional requirements;

IDENTIFICATION OF CANDIDATE REQUIREMENTS

The project initiation documentation (vision) identifies the possible requirements for the project.

STRUCTURE OF THE SYSTEM USED IN "PACKAGES"

One of the fundamental tasks of *software system modeling* is to "*structure the system into small areas*" that are easier to handle.

The units of grouping and logical structuring proposed by the UML language are: *packages and subsystems*.

IDENTIFYING SYSTEM REQUIREMENTS WITH THE HELP OF "CASES OF USE"

Use case modeling is the most effective technique *for identifying requirements from a user perspective*.

Use cases are used to model *the mode of operation of the current system or the mode of operation of the system desired by users*.

Use cases are generally the starting point in object-oriented analysis using the UML language.

OBJECT-ORIENTED ANALYSIS

The main activities to be completed at this stage are:

- Refining the use case diagram;
- Modeling the system dynamics (using the sequence diagram);
- Modeling the static structure (using the class diagram);

REFINE DIAGRAM OF USE CASES

At this stage, other use cases and other actors can be identified, at another level of detail.

SYSTEM DYNAMICS MODELING - SEQUENCE DIAGRAMS

Sequence diagram is one of the most suitable types of diagrams for *modeling the interactions between system objects*.

In a sequence diagram, *the objects involved in the scenario are represented by vertical dotted lines*, and the messages transmitted between them, *through horizontal vectors. The messages* are represented chronologically from top to bottom, the horizontal spacing of the objects being arbitrary.

During the analysis, *the message is named from the existing system*. Subsequently, during the design, it is replaced by *the name of the method of the called object*. *The called or invoked method belongs to the class instantiated by the object receiving the message*.

Also, the primary role of the analyst is to decide, after the execution of each iteration, the direction in which the analysis process will continue or, as the case may be, if it is necessary to interrupt the process.

MODELING THE STATIC STRUCTURE THROUGH CLASS DIAGRAM

The class diagram is the main tool for *analyzing and designing the static structure of the system. The class diagram* specifies *the structure of the system classes*, *the relationships between the classes and the inheritance structures*.

During the analysis process, *the class diagram* is constructed with the aim of obtaining an *ideal solution*. When designing, the same diagram is used and modified, to comply with the implementation details.

OBJECT-ORIENTED DESIGN

For this stage are defined a series of activities that must be completed for the good development of the statistical analysis process regarding the quality control of the production:

- Static structure modeling (using class diagram);
- Modeling the system dynamics (using the status diagram or activity diagram);
- Refining the use case diagram (using the activity diagram);
- Modeling the system architecture (using the component diagram);

SYSTEM DESIGN WITH THE HELP OF CLASS DIAGRAM

During the design, the class diagram is modified to take into account a number of concrete details of the system implementation.

The category of these details includes the graphical user interface, represented by the interface objects.

BEHAVIOR MODELING THROUGH STATUS DIAGRAMS USE OF ACTIVITY DIAGRAMS

The status diagram is used *to model the dynamic behavior of a single object or class of objects*.

The status diagram captures *the sequence of states* that an object of the class goes through during its entire life cycle, in response to the stimuli received, but also the object's own responses and actions.

A status diagram will be defined for each class that exhibits significant dynamic behavior.

REFINE THE DIAGRAM OF CASES OF USE WITH THE HELP OF THE ACTIVITY DIAGRAM

The activity diagram provides a *graphical tool for modeling the processes of a use case*. *The activity chart* is similar to a *flow chart*, the significant difference being that, *the activity diagram can represent parallel processes*.

3. MODELING THE SYSTEM ARCHITECTURE

The system architecture includes the most significant *static and dynamic aspects of the system*.

The system architecture has its origins in user requirements, effected by use cases, but it is influenced by many more aspects, such as:

- The software platform on which the application will run;
- Database management system;
- Structural considerations;
- Legal regulations;
- Non-functional requirements;

The architecture of the system reflects *an image of the whole project, which highlights the essential parts, ignoring the details*. This is illustrated by *the component diagram*.

Bibliography

1. Iacob I., “Modelarea obiect orientata a sistemelor informatice”. Editura Universitara, 2011
2. Iacob I Modelarea obiect orientată a sistemelor informatice, Ed. Universitară, Bucureşti, ISBN: 978-606-591-311-0; (2011)
3. Iacob I Soluţii informatice pentru dezvoltarea aplicaţiilor economice utilizând limbajul PL/SQL şi Oracle Developer, Ed. Universitară, Bucureşti, ISBN: 978-606-591-313-4; (2011)

4. Jacobson I., Christenson, Jonsso, “Oriented Software Engineering”, Adisson-Wesley, 1999
5. Booch G., Rumbaugh J., Jacobson I., “The Unified Modeling Language User Guide”, Addison- Wesley, 2009