

# Making a success of preliminary analysis using UML

*"Pre-modeling" techniques: a bridge to the model*

© Softeam 2004

Philippe Desfray (see [About the author](#))

## Introduction

Even today, making a success of the development of a software application is a tricky business. According to a report published by the Standish Group<sup>1</sup>, for every software application development embarked upon, there is a 23% chance of failure, a 49% chance of becoming sidetracked from initial functional, schedule and budgetary previsions, and only a 28% chance of success. And these figures do not even make a distinction between the evolution of existing applications and the development of new ones, with the latter clearly running a much greater risk of failure.

The future success or failure of a development is essentially determined during the initial phases preceding actual software development itself. During these phases, the main project participants, who have the power to make key fundamental decisions (decision-makers, users, clients), build strategic objectives. An outline of the domain to be handled is sketched, and functional, budgetary, planning and architectural commitments are agreed upon. At this stage, the aim is to define realistic objectives that will be understood by everyone<sup>2</sup> and that will exhaustively cover the innovatory vision of the project. Armed with a successful preliminary analysis, a list of well-formalized requirements and a sufficiently complete domain definition, development realization allows the isolation and reduction of risks, which can be mastered through:

- the appropriate selection of development resources
- sound technical skills
- a realistic budget and schedule
- well-organized and well followed-up project management
- a development method and quality control process adapted to the topology of the project.

On what basis should the realization of initial phases be undertaken, and using which techniques and methodologies? The answer to these questions depends mainly on the context, but a wide range of techniques exists, providing useful assistance to help developers successfully carry out initial development phases.

---

<sup>1</sup> Chaos Report, Standish Group, 2001

<sup>2</sup> In some cases, such as domains which innovate in terms of products and services, the objective is not initially completely clarified. It is then necessary to define identified objectives, before iteratively proceeding with subsequent steps, until a complete project specification is obtained.

## Objective of the preliminary phases

It is necessary to define the application domains on which a system is to be put in place, and the processes that the system must support. Terminology, definitions and domain boundaries are clarified, in order to explain the problem in a clear context. In this domain, functioning modes must be explained, in the form of business procedures, but also in the form of rules and business constraints. An analysis of what already exists must be carried out, by representing it as a system whose structure, roles, responsibilities and internal and external information exchanges are shown. All preliminary information must be collected, in the form of documents, models, forms or any other representation. The nature of the products developed by the processes is explained.

From this point on, weak points, points to be improved or new points to be introduced, which constitute the basis of the definition of the future system, must be identified. The term "*system*" should be understood here in its widest sense: in the world of information systems, the system is the whole organization used to provide a service by processing operations. It is a human, material organization, dealt with only in part by the software. In the world of technical systems, it is an assembly of software/hardware/physical material, with these elements being designed jointly. The definition of the functions to be handled by the software - either developed specifically or taken off-the-shelf - as well as the definition of responsibilities between the software and the other participants, takes place during the preliminary phase.

At this stage, representations of what already exists are used, in order to present the system according to the vision and the objectives of the desired evolutions.

In the rarer case of new applications, there is not always the need to represent what already exists, but the definition of the domains and processes requires a more sustained level of work.

During the preliminary phase, a major issue consists of using representations, which allow dialog between involved parties, such as users, analysts, hierarchical managers and domain experts. The exaggerated use of modeling techniques like UML hinders understanding for certain participants, and can be detrimental to their involvement in modeling or reviews. On the other hand, the absence of a rigorous method of representation can prevent the gathering of precise, consistent and relevant information, which can be summarized or detailed according to desired dialog levels. Several techniques will, therefore, be combined, so as to specifically address each type of problem, but also to provide representations that are easily understandable by the various categories of people involved in the initial phases.

This preliminary information will provide representations that constitute the basis of a *contract*<sup>3</sup> for application developers. This contractual dimension reinforces the necessity that everyone be able to understand and share what is expressed. It introduces a very particular method of managing the deliverables resulting from this phase:

- A contract is not modified lightly. Every modification regarding both its nature and its consequences must be agreed upon. A history of contractual changes must be retained.

---

<sup>3</sup> This is only one of several elements of the contract, which must specify, for example, the repartition of roles and responsibilities, as well as organization modes during the realization phase.

- The contract must constantly be referred to, in order to justify the work subsequently carried out. It must be guaranteed that each of the contract's terms has been satisfied by the developed application, and that the components of the applications all be justified by a link to the initial contract. At this stage, traceability becomes a key factor in mastering development.

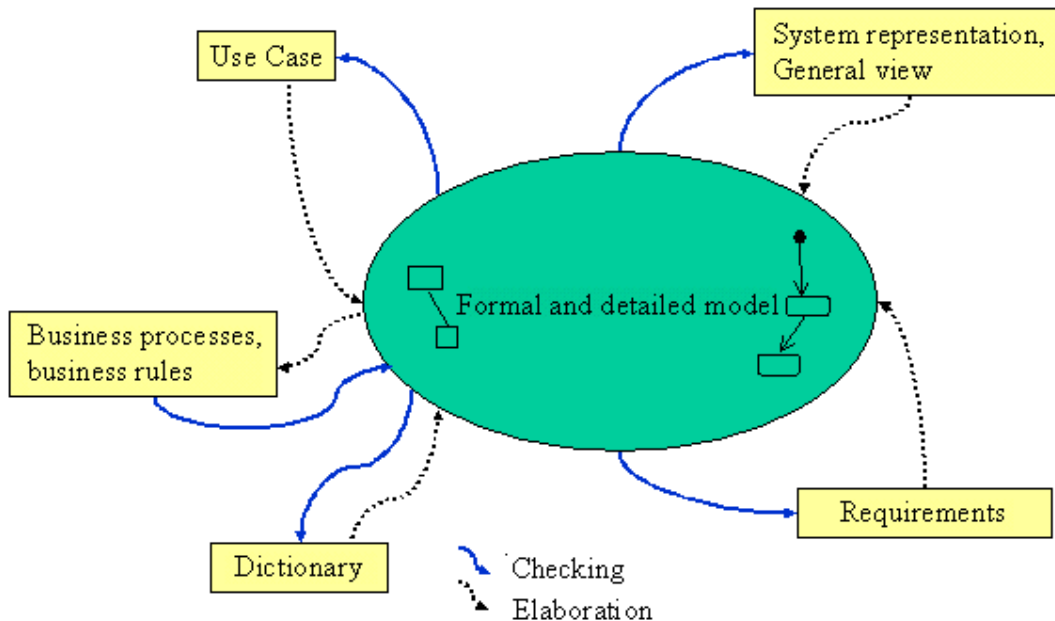
The literary reminder of the initial requirements is an important part of this contract, as it will often be necessary to refer to it when making choices (as well as to keep your feet on the ground), when laden down by technical and practical realization difficulties.

## Techniques used during the preliminary phase

### Overview

The premature production of models is a risky business. Modeling imposes a formal vision of the problem in hand. It determines choices that can turn out to be premature and easily produces an assembly of boxes and links which do not necessarily represent the reality of the problem and its objectives. Above all, it is fundamentally important to list relevant information on the nature of the problem to be dealt with and the core objectives of the project.

However, it can be beneficial to realize models. Indeed, by forcing the analyst to describe what is implicit and to consider the significant aspects of the process, these can help formalize the problem, structure the development context and allow the identification of any holes in the initial analysis. Models help make an overly informal initial base consistent. Aside from the formalization necessary to the realization of the software, the model also provides the business with the means of mastering its own concepts and clearly stating its own procedures. [Figure 1](#) shows that it is necessary to associate several informal or formal modeling techniques, so as to be able to enter all the angles of a problem without distorting it, and then to detail them in a central model, which can be fine-tuned and tweaked until it is implemented.



**Figure 1** - Association of formal and informal models during the preliminary phase

As an example, the construction of a model can start with the gathering of requirements and the construction of an application's dictionary, before continuing with the construction of a model of business procedures and a model of domain notions, which will be traced to the dictionary and initial requirements. A use case model traced and deduced from business procedures and requirements can then be used to isolate the role of the information system

within the global system. Finally, the application model will itself be traced to these initial models.

Figure 2 illustrates a development cycle, which can be used to iterate models, from the most informal to the most complete and formal. Participants who are not computer scientists can participate in informal representations.

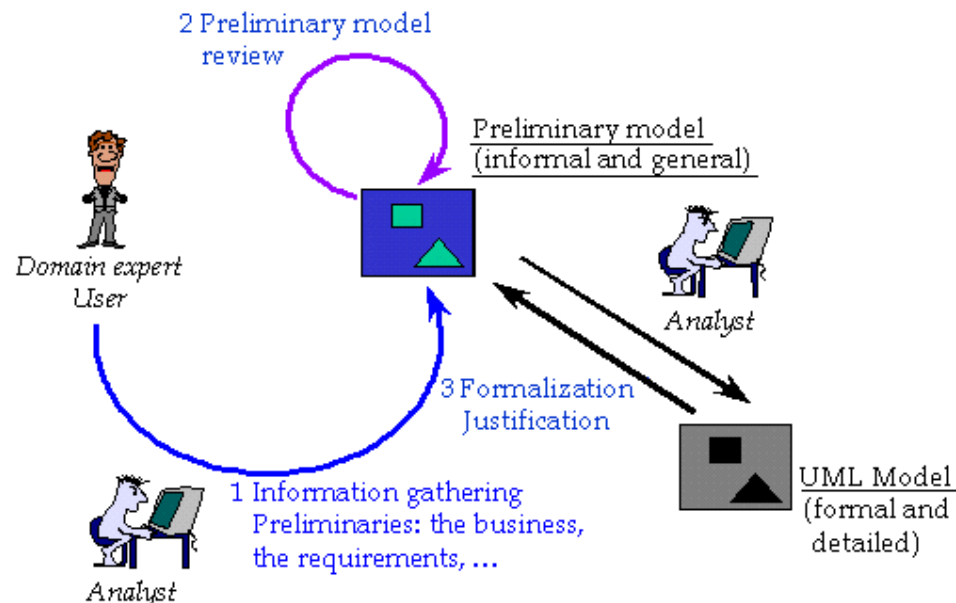


Figure 2 – Model construction steps during preliminary analysis

The following list, which does not claim to be exhaustive, presents the most frequently used techniques. In practice, their use depends on a large number of variations, including in their denomination. However, for each of them, there is a movement towards uniformization and consolidation, translated either by tools providing similar functionalities, or by books dedicated to these practices, or even by standardization efforts such as those of the OMG<sup>4</sup> on the system engineering approach, business rules or Business Process Modeling.

- **Dictionary**: Relevant terms and definitions relative to the system. The dictionary is a simple and efficient means of describing the application domain.
- **Systemic approach**: Global representation of the present and future system. The systemic approach provides details on the system's responsibilities. It draws from the *systemic analysis* techniques already known from previous information system analysis approaches, the *system engineering* approaches used in the technical domain, and the application cartography techniques used in the domain of information systems.
- **Analysis of needs and requirements**: Classified, formalized, maintained, traced, very similar practices are provided by several tools and recommended by several different approaches.
- **Business process modeling**: This technique is essential to the definition of an organization's processes. Certain tools provide notation specific to this technique, but UML provides adapted formalism, close to workflow models, called activity diagrams.

<sup>4</sup> OMG: Object Management Group, standardization organization, which has notably standardized CORBA and UML.

- **Business rules:** Description of fundamental rules, which manage an organization's functioning. Several tool families support this approach in various different forms. The UML language provides a basis for the expression of business rules through the notions of constraints, invariants and pre or post-conditions.
- **Use cases:** Description of the main use cases internal to the system. Frequently used by UML practitioners, this technique is subtended by commonly used modeling approaches.
- **Conceptual models:** UML allows the representation of the principal notions of a domain or a system, thereby formalizing preliminary knowledge of the system, notably through the use of class diagrams.

All the information gathered using these techniques is then used to constitute the system's repository. The repository is used to identify the elements present in the information system, as well as to provide lists and identifiers, which will be constantly used and referred to throughout the life cycle of system developments and evolutions.

All those participating in the system are required to validate both the models and the information gathered. Specific tools are needed to suitably present the model at various management levels to different end-users, so as to allow them to understand the organization of the process, as well as the role that each user must fulfill.

Traceability management is an important element throughout the approach. The simultaneous implementation of different techniques means that overall consistency must be managed, the completeness of each aspect must be guaranteed, and crosschecks must be allowed.

In general, the UML modeling standard provides valuable assistance during the modeling of initial phases. This assistance will be reinforced through extensions to the recently adopted UML2.0 standard (June 2003). However, UML does not cover all the needs of these phases, and extensions and complementary techniques are necessary. The table below gives a short summary of the degree of support of initial phase modeling techniques provided by UML.

Technique	UML Support	Tools
Requirement analysis	N/A	Independent dedicated tools; Objecteering/UML supports requirements as a UML extension
Dictionary	N/A	Support of the dictionary as a UML extension
Business process modeling (BPM)	Activity Diagrams The OMG will provide standard extensions for this need.	Independent tools, or UML tools (activity diagrams with dedicated extensions (profiles) <sup>5</sup> )
Business rules	Constraints, Invariants, Pre/Post conditions	Dedicated tools, or UML tools with specific extensions (profiles)
System approach	Package diagrams, subsystems, Information flows (UML2.0), assembly models (UML2.0) The OMG will provide a standard dedicated to system engineering	Specific tools for information system cartography, dedicated tools for system engineering, UML tools. These can be specifically extended (profiles) for cartography or for the support of "system engineering".
Use cases	Use cases	UML tools. Objecteering/UML provides methodological extensions for the use of use cases during initial phases.
Design models	Class diagrams	UML tools
Traceability	Notion of dependency	Techniques specific to each tool. Some UML tools (including Objecteering/UML) support traceability.

## Dictionary

The dictionary technique is the simplest and most immediate to apply. Building a dictionary simply means listing the terms, which are relevant to an application domain, and producing their related definitions. This is fundamentally important and useful work, which can be undertaken from the start. A domain's terminology represents essential knowledge, which benefits from a high level of stability once it has been finalized. It is not up to developers, but rather to domain specialists to define a domain's terminology. Once established, the dictionary is a precious guide for developers, who are able to use it to produce consistent model names. It constitutes a gauge of the completeness and relevance of a model, through the traceability between a dictionary and a model.

The dictionary can allow a certain freedom in its collection of terminology, and can include a large diversity of vocabulary, which co-exists in a single organization or domain. The model subsequently carries out a terminological reduction, by choosing to only name a thing or a concept once.

## Requirements analysis

A requirement expresses one of the objectives of the system to be realized. Formalized in the form of an identifier (name, number, and so on) and a few short sentences expressing the need itself, it has a set of properties (priority, origin, and so on), which allow it to be managed. Requirements often have a contractual nature: their identification, management and traceability are fundamental to the management of their evolution, as well as to measure the coverage and respect provided by a model with regard to expressed requirements. Traceability expressed between requirements and a model can also be used to establish impact

<sup>5</sup> The "UML profile" is a standard UML mechanism used to extend UML to a specific objective (application domain, target technology, methodology).

analysis - for example, to identify the modifications induced by the evolution of a requirement.

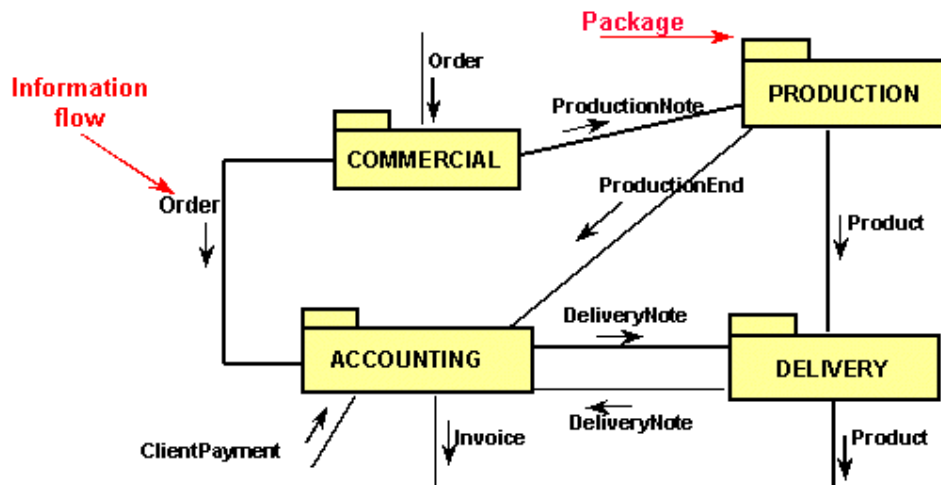
Requirement analysis can, for example, help to identify or justify the presence of a process. Three types of requirements should be identified. Firstly, external requirements are identified, for example, client requirements with regard to an organization's services. Secondly, internal requirements are described, such as the respect of quality, legal and traceability constraints. Finally, non-functional requirements related to the performance of a system, its availability, the tolerated error rate, and so on, need to be identified.

Requirement analysis can be conducted according to specific methodologies, which classify requirements into various types (for example, functional, non-functional, ergonomic, and so on), and which attach new properties to a requirement.

Based on a definition of the domain and the process provided by the dictionary, on business rules and on the modeling of business processes, requirement analysis expresses the motivations leading to the development or evolution of a system.

### **Systemic approach**

For large-scale systems, where there is often an existing system to take into account, general diagrams must be constructed, in order to transcribe a global vision of the system's components, together with their cooperation mode. For these diagrams, it is vital to establish a dialog and to have the approval of all participants. The organization model represents company structures involved in business processes with information exchanges. It highlights the responsibilities of the represented sub-systems.



**Figure 3** – Flow diagrams (Objecteering/UML) illustrating information exchanges between sub-systems

UML2.0 introduces the notion of "information flow", thereby providing a new way of presenting the sub-system's cooperation. SOFTEAM, which introduced this notion into the UML2.0 standard, has been implementing information flows in its Objecteering/UML tool for several years.

The urbanized cartography of information systems is a variation on the systemic approach, allowing the supervision of a company's application inheritance, notably through the introduction of EAI dimensions (Enterprise Application Integration) and *workflows*.

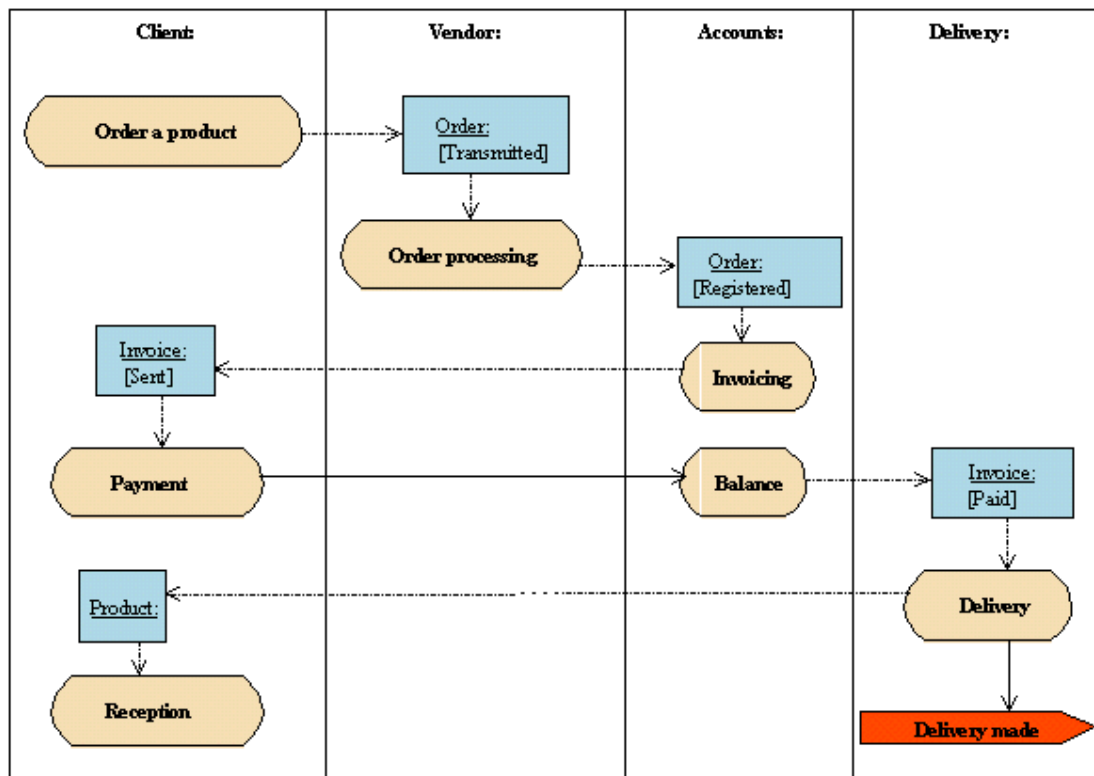
It acts as a preamble to every information system and applicative evolution, by producing a diagram of what already exists, and by explaining the urbanistic evolutions planned for the system.

System urbanization modeling approaches vary according to different companies and practitioners. UML is ever-more generally used, through its wide range of models: class diagrams for business objects, interaction diagrams to express exchanges and cooperation, activity diagrams for business processes, and package diagrams to illustrate an overview of the system. With UML 2.0 not yet in use, new tools such as the information flows discussed above, and assembly models (notions of "part" and "port", "connector", "component", and so on), which are extremely useful when describing the composition of a system, are only rarely implemented.

*System engineering* is used for technical systems such as air traffic control or weapons control systems. In these systems, which are themselves made up of other systems, components range from physical materials (planes, tanks, and so on) and human elements to hardware and software elements. A large variety of technologies is implemented. Standardization work is currently underway within the OMG, with the objective of defining a standard way of using UML2.0 (a UML profile) for all the participants of this domain. System engineering also focuses on the breakdown of a system into sub-systems and on the cooperation and intervention modes of these sub-systems. Great importance is attached to non-functional constraints, such as performance, speed, response time, memory consumption, and so on. UML2.0 assembly models constitute an element eagerly awaited by practitioners in the domain. System engineering also uses needs analysis techniques to a great extent, and relies on software simulation to guarantee that the configuration of assembled systems will be efficiently consistent, by providing the desired functional coverage alongside the planned acceptable performances, in accordance with the project specification document.

### ***Modeling business processes***

This technique is very frequently used in the context of information system development. A business process is rendered by a set of activities realized by various participants, in order to reach a specific goal. For example, "Open a new account" for a banking system or "Process a claim" for an insurance system are business processes. A business process describes the business, and not the information system. The activities of a business process can, however, be linked to the information system. Certain activities of a procedure can also be independent of the information system. The aim is to formalize flows and the synchronization of the different tasks involved in the realization of a service.



**Figure 4** – UML activity diagram representing a business process (processing of an order, partial model)

The level presented in Figure 4 is already highly detailed, and supposes a good level of consensus. Before getting to this stage, agreement must be reached on the list of processes, which are linked to the domain to be modeled. Fundamental points, typically external processes in which the client or partners are involved, must be used as a basis, before presenting internal and sometimes more conflictual processes.

## **Business rules**

Business rules are sets of rules determining business functioning. These can be rules that prevent, provoke or allow the triggering of certain processes, such as, for example, an assertion that defines or constrains certain business aspects, a term or a fact (structural assertion) that provides information on the business, or a constraint managing actions and processes within the system.

A rule must be "atomic", in other words, it must not be possible to break it down into sub-rules.

For example, common law can be seen as a set of business rules used to order our society. Business rules provide an abstract and relatively stable<sup>6</sup> model, which is independent of underlying technical choices. They provide a vision, which is complementary to the other techniques used during the preliminary phase.

UML provides the notion of constraint, used to associate constraints to all the different parts of a model. By adapting this notion to the different types of business rule (dedicated UML

<sup>6</sup> The stability of business rules can be questioned during major changes, such as changes in regulations.

profile), UML can thus support the representation of rules, with the advantage of being able to situate them in the context of the elements to which they apply.

### Examples:

- An account manager cannot manage more than two sales areas and/or 10 clients.
- A client can only place an order via the internet if he has already registered.
- Clients reputed to be less financially solvent cannot order luxury products, and can only order the minimum amount without additional guarantees.
- A married employee must only be transferred as a last resort.

### Use cases

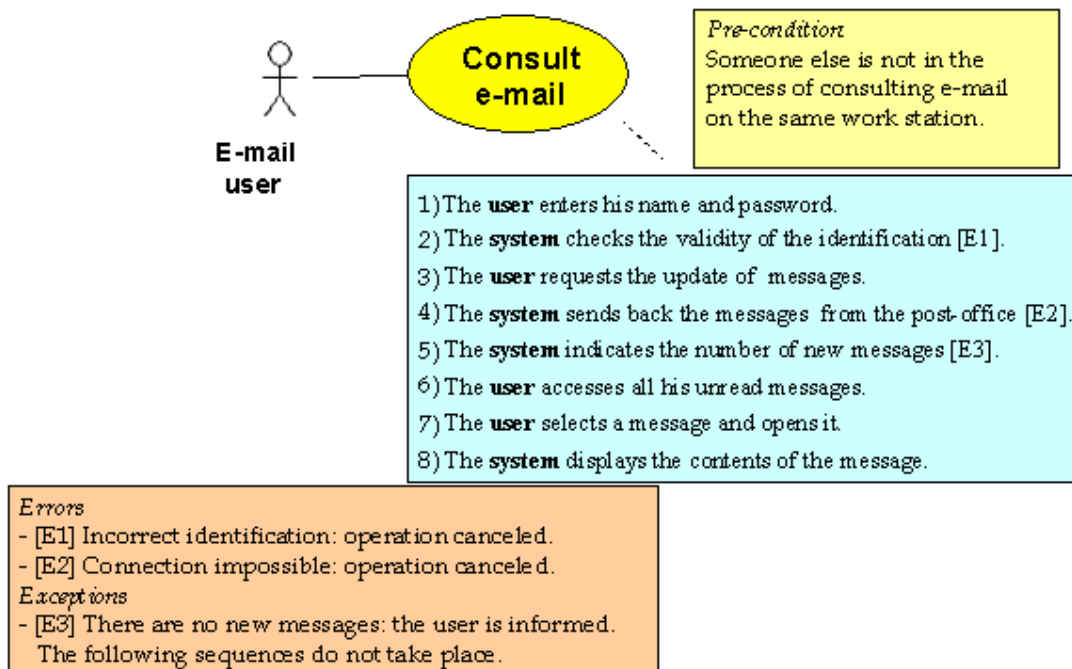


Figure 5– Example of a UML use case diagram

The use case technique is very popular amongst UML practitioners. Its application scope applies to the system to be developed, once this has been identified. The use case technique is used after the business process has been modeled, the dictionary defined and the business rules described. For example, for any activity of a business process, which uses the system, one or several use cases allowing it to be realized must be identified.

Many works have been published on the implementation of the use case model, thus providing well-documented methodological support. The advantage of use case modeling is that it allows an external definition of the system, by identifying the actors who can participate in the future system, as well as the main examples of its implementation. A use case model is a structuring tool for analysis activity and for future UML modeling.

The use case model applies in a specific context. The functional scope of the system is identified and the global requirements and general objectives of the system are well known. Its use must be restricted to a coarse granularity definition of use cases, each associated with at least one external actor, and each running a complete function (functional transaction).

Use case models sometimes slip into true functional breakdowns of the services provided by the system, which is detrimental *in fine* to the successful modeling and design of the system.

### ***Order of these modeling techniques***

The precise order of these modeling techniques is based on a methodology that will greatly depend on the context in which they are implemented. This depends on the application domain, the size and importance of the problem to be dealt with, and on numerous other factors. Frequently, only some of the techniques presented are actually implemented. Several types of representation can be built alongside each other, through an iterative approach.

The rule is to start from the best known, most general, most immediate and most stable aspects, before progressively providing more detail, whilst constantly searching for the consensus and approval of representations from other participants. On the one hand, it is important to operate according to the "top down" order, covering the entire process level by level, with each level corresponding to an even finer level of modeling detail than the previous one. On the other hand, the premature realization of the partial and detailed modeling of a particular aspect of the process must be forbidden.

The *dictionary* can always be defined first. It is used to group existing definitions without providing interpretations. *Requirement analysis* also appears very early on. Conducted at the same time as the construction of the dictionary, it is used to focus the required terminology. The *systemic approach* provides a means of rapidly providing a global vision of the system. The *definition of design models* can then precede or be conducted at the same time as the *modeling of business processes*, since processes use data. They are justified by the products created, and allow the necessity of certain intermediary data to be identified. *Business rules* enrich design models and process models. *Use cases* come last, and detail exactly what the information system is to do within the overall system.

## Tool supporting initial phase modeling

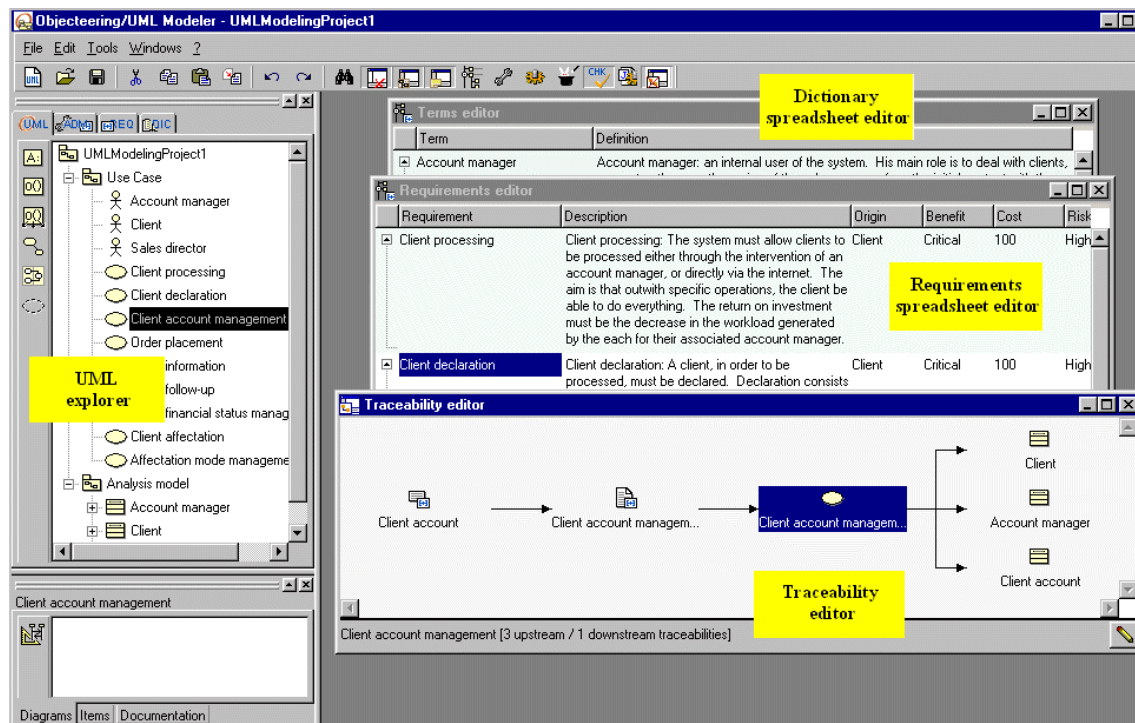


Figure 6—Objectteering/UML – support of initial phases ("UML requirements" module)

All the techniques that can be used during initial phases, whether the result of textual descriptions or modeling, must be coordinated, both by an appropriate development process, and by a federating tool. A single repository managed by the tool is needed. The tool also ensures close cooperation between different forms of representation, by maintaining the consistency of the whole. For example, traceability must be supported and if possible automated by wizards, so as to manage the degree of modeling coverage, the equivalence between different modeling levels and change impact studies.

Appropriate tools allow the provision of different views of the same model, thereby providing a dedicated ergonomics for each type of technique, such as UML graphic editors, and spreadsheet editors for dictionary and requirement definition.

UML extension and adaptation facilities (UML profiles) allow the user to adapt UML to his particular needs and approach, which are often conditioned by the company and the application sector.

Finally, the tools must provide enhanced developer productivity, which, as well as increased efficiency, ensures the acceptance of the tool and the underlying approach on the part of developers. Essential functions are automatic documentation production functions, used to produce documents that conform to a specific format particular to the company, based on work done on models or structured textual elements. The tool ensures the transition between different phases, as well as the transfer of information between phases, thereby providing a significant gain in productivity and quality.

The production of unified and consistent documentation from models, and the addition of remarks in natural language, allow documentation to be re-read and globally reviewed, including remarks, so as to allow human validation, which remains indispensable.

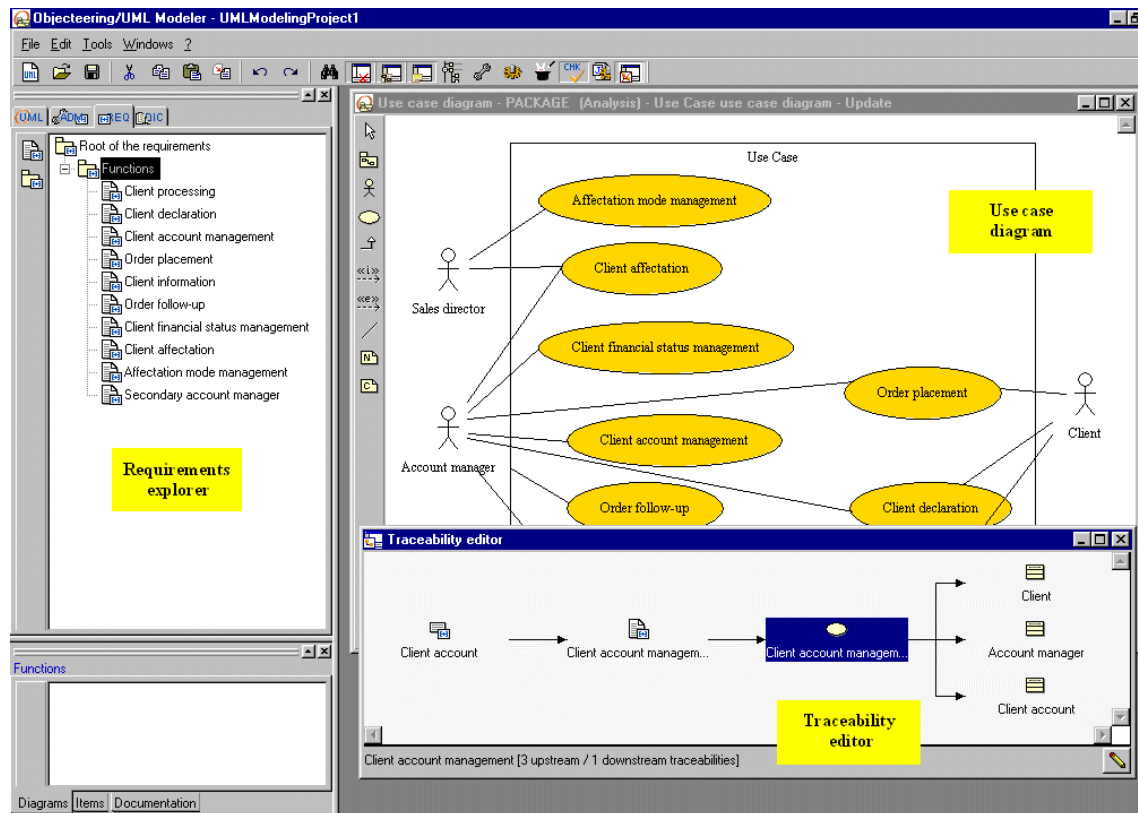


Figure 7—Objectteering/UML – joint definition of "use cases" and requirements

For further information, please see [www.objectteering.com](http://www.objectteering.com).

## Conclusion

A correctly built model provides software producers with a serious, stable and relevant functional specification. Together with the technical constraints specific to the company's IT, this specification can be used as the starting point for their work.

Furthermore, the activities implied by modeling - the clear development of business processes and the laying out of the information used – allow the company to master its own concepts and language, and to provide clear priorities: dictionary documentation, repository documentation, business rules documentation, their appropriation by users, and their validation by directors. The clear identification of these priorities allows the specification and stabilization of the computerization of the company, thereby improving the quality of reflection on its role and its development.

Modeling is thus simultaneously a useful step in the technical realization of the software, and a phase essential to the maturity of the company, in terms of its information system.

## About the author

**Philippe DESFRAY**, co-founder and technical director of SOFTEAM, is an internationally renowned expert on models and methods, notably those based on UML. Creator of the "Classe Relation" object-oriented method in the 1990s, Philippe DESFRAY has published three books, in particular "Object Engineering - The fourth dimension", published by ADDISON WESLEY in 1994, and has driven the development of the UML tool suite "Objecteering/UML". Forerunner of MDA<sup>7</sup> technology, Objecteering/UML introduced support of this approach in 1994. Since 1994, Philippe Desfray has represented SOFTEAM within the OMG, where he has actively participated in the development of several standards, most notably UML. His continuing work on model-driven development has led to him to participate in the definition of the UML standard from the very beginning, by directing the definition of new notions such as UML profiles, and to develop support of these notions within Objecteering/UML. At the head of extensive R&D activity within SOFTEAM, and in partnership with large European organizations, Philippe Desfray is committed to directly applying results across the entire range of SOFTEAM activities: consulting, training and support through the Objecteering/UML tool.

## Acknowledgements

This white paper has been enriched by the remarks and contributions of the following people, all experts in modeling, methodology and requirements document definition. I thank them most sincerely for their help.

- Arnaud Ladrière
- Laurent Lieber
- Dominique Vauquier
- Michel Volle (<http://www.volle.com>)

---

<sup>7</sup> "Model Driven Architecture": approach and technology at the base of new OMG standards.