

UML and Object Oriented Drama

Luca Vetti Tagliati, PhD Student - Birkbeck University Of London – Senior Technical Lead – Lehman Brothers (UK)

Carlo Caloro, Accademia d'Arte Drammatica "Silvio D'Amico", Rome (Italy)

Abstract

Readers of this article have probably seen, at least once, diagrams produced using the UML (*Unified Modeling Language*). Some of you have perhaps used UML for your own work and know that the UML can also be applied to a variety of analytic disciplines, ranging from economics to electronics, from mathematics to medicine, etc. However, you are less likely to have encountered UML in the strictly artistic domain of Theatre. Yes ... Right in the theatre.

In this article we illustrate a unique experiment: the application of the UML for analysis of dramatic words.

Daily interdisciplinary tests and hybridisations occur among the different contemporary arts; theatre meets new technologies more and more frequently and it turns out to be a fascinating and complex meeting. As a consequence, the problem of identifying new tools that the dramatist and/or director can use to analyse a text arises. Among these tools, it is important to select which is to be used in order to facilitate the sharing of the project among the different participants (director, dramatist, actors, etc.). The discipline of computer science can provide us with a valid solution to this problem, where comparable problems are, typically, solved by means of UML.

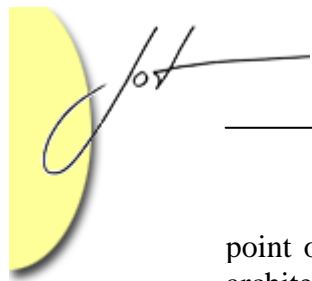
1 ANALYSIS OF DRAMATIC WORDS

Introduction

As a planning unit we chose the episode *Ecuador* by Aristides Vargas, an Ecuadorian, (see appendix A), taken from the play "*La Cruzada de los Niños de la Calle*" ([CRZ2001]), created by six Latin American dramatists coordinated by the Spanish dramatist José Sanchis Sinisterra. Its aim is to bring the problem of thousands of abandoned children wandering the streets of big cities in the Third World, forced to live a predestined life with no future, helpless victims of child prostitution, drug and organ dealing, to the attention of the world.

Being a dramatic text, it has to be analysed first using semiotics tools. This is necessary since, without determining and identifying the subject and main actions, prossing it using UML becomes increasingly challenging. The aim of this first analysis, which focus on the innermost structure of the text, is to identify all the elements of the drama considered important for analysing the surface and phrase structure of the text. This allows the production of the draft of the direction book, as a

Cite this article as follows: Luca Vetti Tagliati, Carlo Caloro: "UML and Object Oriented Drama", in *Journal of Object Technology*, vol. 7, no. 1, January – February 2008, pp. 85-101 http://www.jot.fm/issues/issue_2008_01/article2/



point of reference for directing the actors, and also for planning and modelling the architectural platform for the installation. Applying UML to words becomes, then, the study of the possibilities of a dramatic interplay which can occur between set and technological displays, between natural and not-natural language and between action on stage and systems.

The main problem during the planning stage is the logical distinction between words, understood as accounts of real life or fiction, and the process of planning the architectural platform of staging. So, first of all, it is necessary to distinguish between having recourse to a formal computer science language such as UML, applied only to the content of the text for analysing the connection between forces and interaction among the characters of the play by Vargas, and UML used for describing the architectural platform of staging in detail.

NOTE: To avoid limiting the discussion and consideration about the themes contained in the text to the area limited by the space of staging, Carlo Caloro carried out a one-of-a-kind artistic installation. He designed and implemented an installation where the text by Vargas interacts with the Internet. Starting from the themes covered in the text and expanding by following hyperlinks to other text files, images, sounds and videos existing in the net, a new representation of the word is created. Thus netsurfing is activated by people anytime they start walking on a treadmill, similar to those usually present in fitness centres.

A mention of requirements analysis

The following section was entirely drawn from the book [LVTUML]

Software engineering is a discipline of engineering that deals with analysis, design, development and management of systems, aiming at processing information automatically. In particular, by applying certain processes of developing software, it allows, starting from a complex set of initial specifications, the production of an automatic system (software) able to satisfy them. The more formal modern software development processes include a number of predefined disciplines – such as requirements analysis, analysis and design, implementation, test, deployment, etc.-executed in well-defined phases – such as: *inception*, *elaboration*, *construction* and *transition* -. Each discipline, as expected, involves different professional figures, requires well-defined input artefacts and produces others as output. For example, the design phase is typically assigned to a team of architects, who might be helped by some senior programmer. An essential artefact of entry is the requirements model, while some of the main artefacts produced are the architecture design, and the construction-phase planning, including its delivery (architecture is not only a mere document, but it includes the *mile stone* by which the infrastructure of the system is delivered... The famous 20% of requirements that permit an investigation of 80% of the architecture), etc.

One of the initial disciplines of each software development process is requirement analysis. [PKPK2003] The main activities of this phase include: analysing the real needs of the client/user, the regulations in force within the domain subject of study, possible business obligations, etc. The aim is to produce an initial model of the system, named requirement model, necessary to carry out the design and



implementation of the system. This model, in turn, is made up of other (sub) models, such as:

Description of requested services, typically designed through the UML notation of use cases [Jacobson1992];

Representation of the main business entities involved, together with the mutual relations (*domain object model*), designed through the UML notation of class diagrams;

Vocabulary of the domain subject of study;

Documentation of the non-functional requirements: performance requested, security levels, backup and restore procedures, etc. [IEEE1998]

The requirement model is, typically, produced by a professional figure, the business analyst, and used by the software architect for designing the system.

Requirement and words analysis

From a detailed analysis of the formal software development processes, it is possible to underline interesting similarities between the phase of user requirements analysis and word analysis: they both deal with turning a series of specifications, often communicated in an informal way (i.e. verbally), into a document/model which the architect/director can use in order to produce a system design, implemented by other people (developers/actors).

In the theatre the starting requirements are usually represented by the text of the author complete with a series of captions and dialogues. First the dramatist and then the director, at the very beginning, take the role of the business analyst; their task is to analyse the text in terms of function or from the perspective of staging (see figure 1).

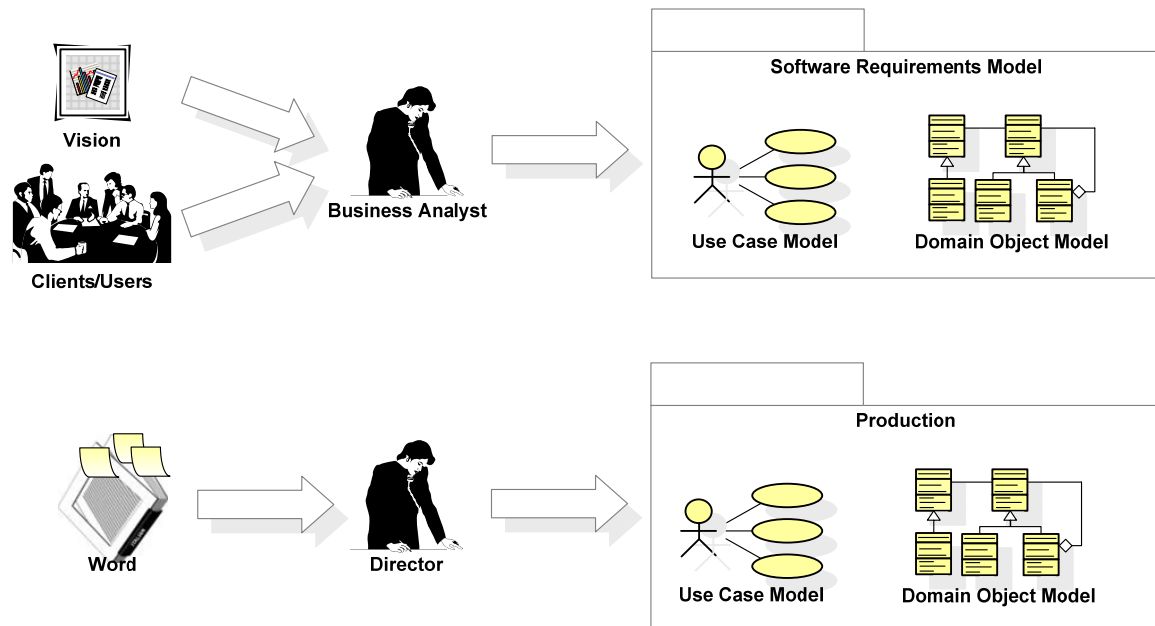


Figure 1. Similarities between requirement analysis and staging production

UML and software development processes

According to the specifications [OMG UML], UML is a language to specify, construct, present and document artefacts of both software systems and business processes and other systems that are not strictly software. UML allows practitioners to present, by means of a formal notation, engineering artefacts and also to illustrate ideas, decisions made and solutions taken. Such a language fosters the spread of information, being an international standard not linked to single companies. In theory, an IT practitioner coming from any country, with the knowledge of UML, should be able to read UML models related to projects and understand them in detail without too much effort and, above all, without incurring the ambiguities typical of any natural language. Although this is not always possible, the usage of UML still helps to minimise the need to write long documentation, inevitably, populated by imprecision, ambiguities, incompleteness and inconsistencies difficult to detect and correct.

However, UML is a modelling language and, as such, is “just” a part of the most general methods for developing software. UML is a formalism integrated into processes to produce, organise and document the artefacts created during the different process phases. A process, among the various principles, is made up of directives indicating who, what, when, where and why. A language, obviously, does not include these guidelines.

Processes have an absolutely essential role in carrying out successful projects, and not only in computer science. Sometimes, in spite of all the best aims, a project fails because the development was not underpinned by a well defined process or because it was not well managed: processes often make the difference between productive and unsuccessful projects.

Identifying the main actors

After studying the episode *Ecuador* by Aristides Vargas from Ecuador (see appendix A), we carried out a detailed study equivalent to the requirements analysis. In particular, the first step was to identify the main actors of the “system”. These are the entities that will use the services described by use cases identified afterwards. *In the use-case notation* “An actor specifies a role played by a user or any other system that interacts with the subject” ([OMG UML]). A “UML” actor is not necessarily a person; it can also be a system or any physical device. In our context, however, the interest is, as expected, limited only to people. In general, a (UML) actor is an entity external to the system that interacts with it. It is the idealisation of a person, a process, or anything else interacting with the system itself, sending and/or receiving messages; i.e.: exchanging information. Actors are graphically represented in UML by a standard notation of a *stick figure*, with the related name at the bottom, although it is possible to emphasise the role/behaviour of the actors by using appropriate *stereotypes* as shown in the use case diagram (see figure 2).

The actors identified during this analysis were catalogued by means of appropriate charts allowing the delineation of the main responsibilities.

Below are some modules used to identify the main characters of the episode subject of our study ([LVT2001]).



Actor: Doctor	Date: 12/02/2006	Version: 2.1
Responsibilities: He manages the orders of “goods” (i.e. organs) from the clinic and negotiates the corresponding payments. Having agreed a deal and captured the necessary victims, he proceeds with actions: equipping an operating room, preparing the victims and then removing their organs. These are then sent to the client clinic. In his lifetime he essentially plays the role of the hunter, targeting not just those are prey by definition, but also his aide (the male nurse) and his own wife. At the end of his lifetime, in according to the “contrappasso” law (also known as retaliation law) he becomes prey himself, being subjected to a trial by his victims.		
Use cases	Frequency	Primary actor
Place order of goods (from the clinic)	High	No
Negotiate the order (with the clinic)	High	Yes
Prepare the operating room for organ removal	High	Yes
Make the explants	High	Yes
Send organs to the clinic	High	Yes
Hunt aid nurse	Low	Yes
Hunt wife	Low	Yes
Hunt the hunter	Low	No

Module 1. Doctor’s actor card

Actor: (male) Nurse	Date: 12/03/2006	Version: 1.4
Responsibilities: He is the Doctor’s aide, but not as clever... He wanders through the depressed areas of the town looking for people, desperate parents, “willing” to sell their kids’ organs. He takes children/prey in custody and prepares them for the removal of their organs. He is not a sophisticated character, especially from an intellectual point of view, and, like the doctor, will end up being hunted under the terms of retaliation law.		
Use cases	Frequency	Primary actor
Look for preys (mainly children)	High	Yes
Negotiate the purchase of organs	High	Yes
Deliver the “prey” (mainly children)	High	Yes
Remove organs (he aids the doctor in operations)	High	No
Falls prey to remorse	Low	Yes
Confesses his misdeeds	Low	Yes
Hunt aide nurse (He is hunted)	Low	No

Module 2. (male) Nurse’s actor card

To properly model use cases, we have to define the system precisely (in terms of its “boundaries”), correctly identify the actors, define the functions of the various diagrams, determine the links between the use cases and validate the system. This activity, often complex, requires a careful study of the available material, a high interaction with the clients, and, often, also with people representing the actors of the system itself. In the very first stages of study of user requirements, ideas are typically in draft format and can be developed by the actors analysis.

NOTE. This note is aimed primarily at practitioners of the theatrical profession and therefore it is not strictly essential to the subsequent discussion. The requirements analysis, in this context, is very similar, taking into consideration Greimas generative semiotics, to an important stage of sense generation named “surface” fiction grammar. The surface fiction deals with actants: fiction entities of a syntactical kind, unlike the other actors, who are, instead, conversational entities,

whose semantic side is important. The attant in Greimas model plays different attantial roles: Subject, Object, Addresser, Addressee, Opposer, Aide (see [GREIMAS2001]). So there is a clear equivalence among the actors of UML Use Cases, whose attantial role is theorised by Greimas, in the same way that a use case is equivalent to an attantial box. They can be considered as logic functions. An actor plays a specific role for each use case he/she interacts and exchanges messages with, as an attant plays the specific attantial role for each attantial box he/she interacts with. They both belong to the deep structure of the text and both can be a convenient abstraction (for example, money, a collective character like children in the street, can be a natural person but also an object). Both the attant and the Use Case actors can be theatrically absent from the scenes (for example, the “Yankee” clinic or the children in the street or the baby’s mother). Afterwards they can also play different attantial roles (for example, the male nurse and the doctor are both addresser and opposer at the same time, as an actor of a use case modifies his/her/its own role and position interacting with another use case). In this stage of text analysis or of the business system described in it, both the attantial model and the classes and use case diagram focus their attention on the same point, that is that actions are what matter, not who does them..

The main use cases

The diagram in figure 2 depicts the main use cases, organised in packages, identified by analysing “The Crusade” section. In this context, packages are used in order to group “functional areas” together while the time element does not affect the use case arrangement. For example, although the `Doctor’s Wife` is “hunted” (by her husband himself) only after becoming aware of her husband’s awful business (The male nurse admits his misdeeds), figure 2 depicts the latter before the former.

It all begins with the actor `Clinic Manager`, manager of the clinic (referred to in the text as “*Yankee Clinic*”), who, wanting to satisfy the requests of wealthy clients, contacts the `Doctor` in order to receive the goods (the organs): he makes an order, which afterwards is modified, since more items are required.

The `Male Nurse`, the actual taskmaster in the street, looks for possible prey. He wanders through the ill-famed suburbs of the town, trying to contact desperate people. His tasks include the management, on behalf of the doctor, of the agreements related to the purchase of organs (`Deal with organs purchase`), the delivery of the agreed price and the custody of the preys (`Deliver the prey`). Once he has found his prey, the `Male Nurse` and the `Doctor` equip the operating room and then remove the organs (`Remove organs`) to be sent to the `Yankee clinic`, in order to satisfy the requests of wealthy people with no scruples.

The awful business starts tottering when the `Male Nurse` begins to be tortured by remorse (He falls prey to remorse). At this point, he confides in the `Doctor’s wife`. As a consequence, the `Doctor` hunts them both. It ends with a crusade: the children march happily (`Take part in the crusade`) and then catch the hunter: the `Doctor` is put on trial and is pronounced guilty (`Try “hunter/prey”`).

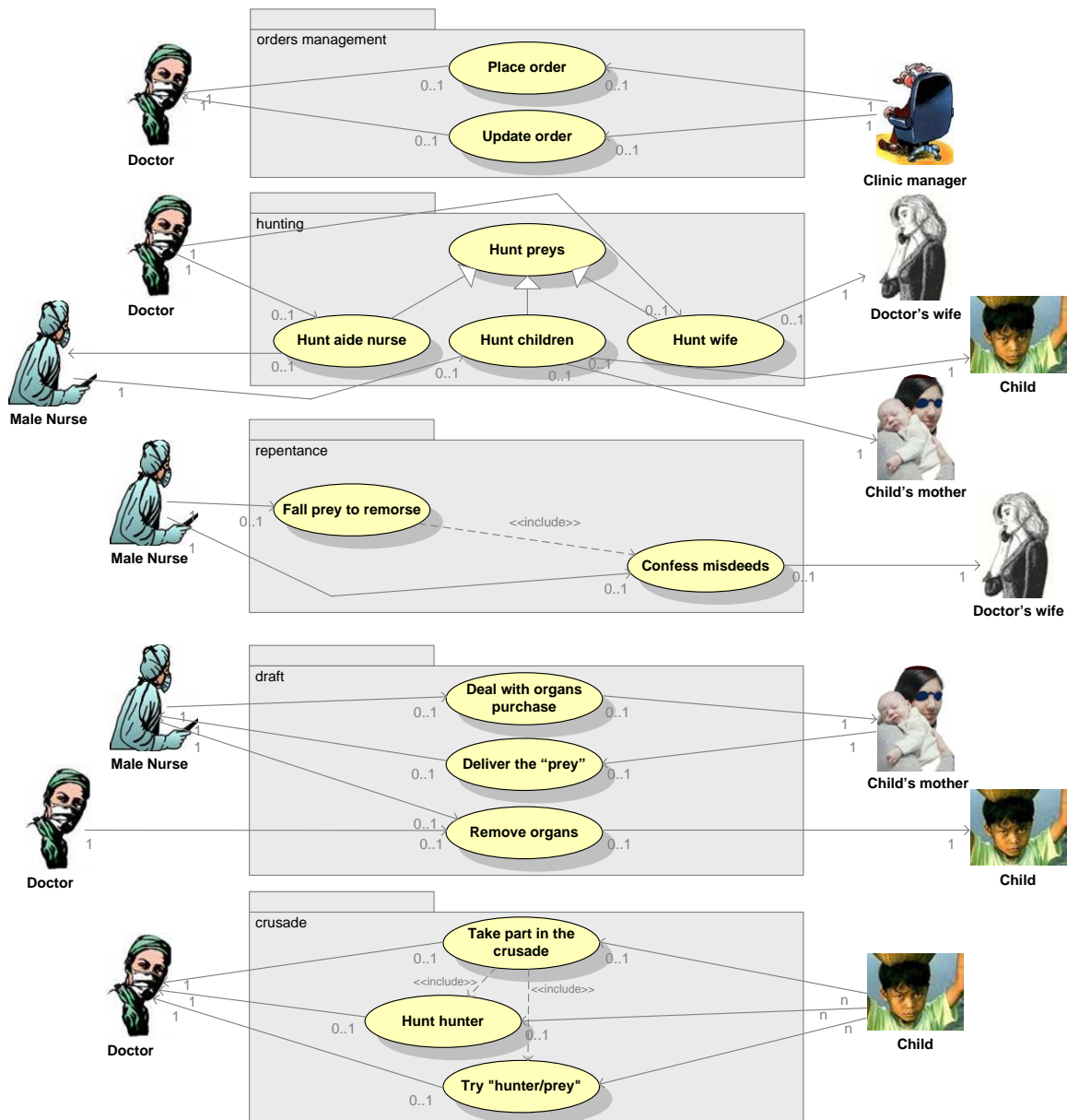
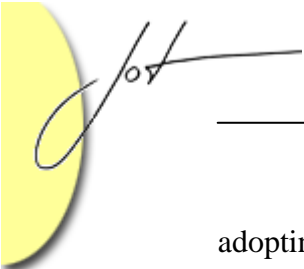


Figure 2. The main Use Cases.

Use cases specification

Limiting the requirements model, specifically the system functional requirements, only to Use Case diagrams (see figure 2) would be too risky a choice. In fact, single use cases (i.e. ellipses) are too “light” to provide architects and developers with precise guidelines about service implementation. For example they do not include descriptions of the list of steps (i.e. the flow of events), the pre-conditions and post-conditions, or the possible anomalies that can occur including the relative management procedures, etc. In order to compensate for this problem there are two possible solutions:

describing the above-mentioned information via other UML diagrams, such as activity and sequence diagrams;



adopting opportune templates (as shown below).

The first solution, although it presents a number of advantages typical of information represented in a graphical way, suffers from severe limitations; such as the following:

it requires a lot of maintenance. Use cases tend to include a number of scenarios (i.e. main, alternatives and exceptions) each composed of several flow forks and branches. Therefore, their diagrammatic representation tends to become confusing and difficult to manage. A technique often used to reduce the diagram’s complexity consists of drawing a number of separate diagrams: one for each scenario -one for the main scenario and others for each alternative and exception scenario. Unfortunately, this technique generates a huge maintenance effort due to the need to synchronise all the different diagrams whenever there is a change to one of the flow;

it is difficult to standardise the extra information, such as the pre and post-conditions, the trigger events, etc.

In order to address the above-mentioned problems, in commercial environments, the use case description is often defined using an appropriate template (for a detailed description see [LVTUML], [ARGR2001]), as follows:

USE CASE: UC_TRY_HP	Try “hunter/prey”	Date: 10/01/2006
		Version: 0.00.004
Description:	The whole episode “The Crusade” evolves around the desire of the collective subject (i.e. “ <i>Los Niños de La Calle</i> ”, Children from La Calle) to have their human rights recognised and to end the awful organ dealing business. This is achieved when they, finally, put their main hunter: the Doctor, on trial.	
Running time:	The duration of pictures 23 and 28 of the episode Ecuador should last minutes.	
Primary Actor:	Children (from the street). Their aim is put the doctor on trial, in order to condemn him, and then to continue the crusade.	
Secondary Actors:	Doctor He is the “prey” of the Court of children. His main aim is to save his own life. He vainly tries to use improbable disquisitions and philosophical-scientific speculations, to justify his deeds.	
Preconditions:	The children catch the doctor.	
Guarantees:	<u>Minimum:</u> the trial takes place <u>Success:</u> the doctor is found guilty and executed.	
Start:	The children in the street start the trial.	

Use case. Try “hunter/prey” use case specification



Main scenery of success.	
1.	Children: surround the doctor in the room where the trial takes place. <i>Caption picture 23 – sound of footsteps, laughter, children’s voices, [...] suddenly there is a harsh light illuminating the Doctor. [...]</i>
2.	Doctor: starts the monologue in his own defence: <i>He addresses an unspecified audience which seems to surround him. “The thing is: things had already been arranged before we came ...[...] Can a man refuse to do what God wants him to do?”</i>
3.	Children: reply <i>(The head of a doll is thrown and falls at the feet of the doctor who, visibly nervous, tries to keep on talking)</i>
4.	Doctor: continues his monologue in his own defence: <i>“The...The soldiers name it ‘due obedience’, but what I’m saying is...I mean that I’m a good professional... Why? Because I’m a good person. That’s what it is: goodness precedes us and for this reason ...”</i>
5.	Children: respond <i>A doll made of rope gets to the foot of the Doctor, who gets angrier</i>
6.	Doctor: continues the monologue in his own defence: <i>“I mean, and I hope you can understand me, that a heart is good independently of who’s got it, as well as a lung... or eyes, eyes...”</i>
7.	Children: reply. <i>(One of them throws a piece of a doll violently and hits the doctor)</i>
8.	Doctor: continues his monologue: <i>“I won’t let you assault me! I’m talking with a scientific and philosophical soul [...].I demand the right of silence.”</i>
9.	Children: respond <i>(Silence for some moments. Then the children, like in a puppet show, say in unison: “Talk, talk!”. Since the doctor keeps silent, they throw pieces of dolls at him while the light is turned off.)</i>
10.	Doctor: continues the monologue in his own defence: <i>Picture 28 Caption: (the voice of the Doctor is heard shouting in the dark, he’s very angry) “Can somebody tell me where you are taking me?”</i>
11.	Children: reply <i>(their chorus turns into laughter and murmuring)</i>
12.	Children: reply <i>A piece of a doll is thrown violently and hits the Doctor</i>
13.	Doctor: continues the monologue in his own defence: <i>“Are you listening to me?”</i>
13.	Children: reply <i>(A soft light illuminates the Doctor: He’s tied to a chair, his white gown is unbuttoned, his t-shirt is dirty. He looks shattered and neglected)</i>
14.	Doctor: continues his monologue: <i>“It is... too hot in here [...].You kidnapped me, organised this awful imitation of a trial and you are treating me as if I were...”</i>
15.	Children: reply <i>(A little storm of pieces of a doll falls on him and makes him quiet. The children say: “Talk! Talk!”)</i>
16.	Doctor: continues the monologue in his own defence: <i>“Well... as I was saying... and I hope you understand... the heart is the most expensive [...] the whole range, especially in the low season.”</i>
17.	Children: reply <i>(One of them throws a piece of a doll with violence and hits the doctor)</i>
18.	Doctor: continues his monologue: <i>“Stop it! I won’t talk anymore!... This way you prove you’re not able to kill, as I am. You’re unable to kill, don’t deny it. Because you aren’t our children, you don’t join our parties, you don’t... You only look until there, until darkness, with holes in your eyes... What are you accusing me of? You can buy only what somebody sells, can’t you? Then?... Somebody sold, I bought and then I sold... It’s a market, I hope you can understand me and I...”</i>
19.	Children: reply <i>The children’s give a rope to a toy soldier, which begins its march towards the Doctor</i>
20.	Doctor: continues his own defence: <i>“Are you listening to me? That deal has always been there...(he gets angry) God covered for every detail, including the possibility of selling your own soul, since it is nothing else than a piece of flesh on scales...”</i>
21.	Children: reply <i>Their voices, among laughter, murmur: “Pum, Pum, Pum!”</i>
22.	Doctor: collapses
Notes:	
1.	The authors of the text proposed to have no more than three characters for each story and asked that the children do not physically appear in the scenery, since working in the theatre with children is difficult, especially with sensitive themes. Therefore, in order to avoid the presence of the children, the above use case, should be “implemented” within the dialogue between the Doctor (his monologue) and the various noises offstage and the several objects thrown on stage.
2.	The previous treatment of the text by means of semiotics instruments and in particular of the attantial model allowed a less abstract and more concrete level of the process of generating sense and to develop oppositions and changes of sense identified and schematised through the semiotic picture. Moreover, they underlined the fact that the doctor cannot be considered as the attantial subject, though it seems so, since he has more lines than anyone else and his desire is not to lose what he has got.

Variations

Applying a template in use cases like these requires a series of variations: the most important are described hereafter.

In the analysis of these use cases there are no alternative or error scenarios. As a matter of fact, in a theatre script, unlike software systems, an actor cannot act in a different way (there are no business exceptions) nor can the Theatre be affected by anomalies (there can be no system exceptions). The dramatist and the director make cuts, and other modifications, but in the end, everything is fixed in the new version of the script. Scenes are predetermined by the author and they occupy a precise, prearranged time and place in the dramatic text. The running time of the dramatic event is fictitious and prearranged according to the text and, unlike real time and the real world, it develops as a linear and irreversible progression of events.

The description of the use cases was completed using the corresponding passages from the text. It was necessary both for setting the use case within its theatre context, and for facilitating its reading, stressing its dramatic features, set by the authors of the text himself.

Domain Object Model

Once the main services (use cases) of the “system” have been identified and described, the next task is to produce the Domain Object Model (although often the two models evolve together providing each other valuable feedback). This is a fundamental artefact in a software development process, not only because it is essential for carrying out other activities, such as the analysis and design of the entire system, the GUI prototype, the database schema, the design of messages, and so on, but also because it allows us to analyse the system, including the requested services, from a “data” point of view. It also allows us to review the use case model: the definition of the data navigation/relationships, typically requires some changes in use case scenarios.

The domain object model is produced during the requirements analysis stage and, as its name suggests, is a model dealing with “actually existing” entities in the area of the study that the software system will have to automate. One of the main aims of the model is to facilitate a full comprehension of the system context (the domain), allowing us to focus on the static structure of the data processed. It is a static model (as a projection independent on the time element) of the business domain, representing an abstraction in terms of the various existing entities and their relating interconnections. The entity classes identified at this point are in an embryonic shape, they will reach the implementation structure, used in the design model, through refinements taking place in subsequent phases of the software development process.

According to the conclusions of I. Jacobson, G. Booch, J. Rumbaugh ([JBR2000]), the entities in the domain model should belong to the following categories:

“business” entities, objects representing entities (more or less tangible), present in the business area of study or that can be derived from them. In an e-commerce business, these entities represent objects, such as: articles, shopping trolleys, catalogues, etc.;



objects and concepts belonging to the conceptual world the system needs to automate. In the above-mentioned example of an e-commerce business they may be discount regulations, events related to specific conditions required by the user (discounted prize, halved prize, etc.);

events which can occur, like notification of a new order, or notification of an offer relating to a specific product to a user, etc.

A typical feature of domain object models is the absence of specification of operations in the various classes. During this stage, the attention is focused on data and their interrelationships.

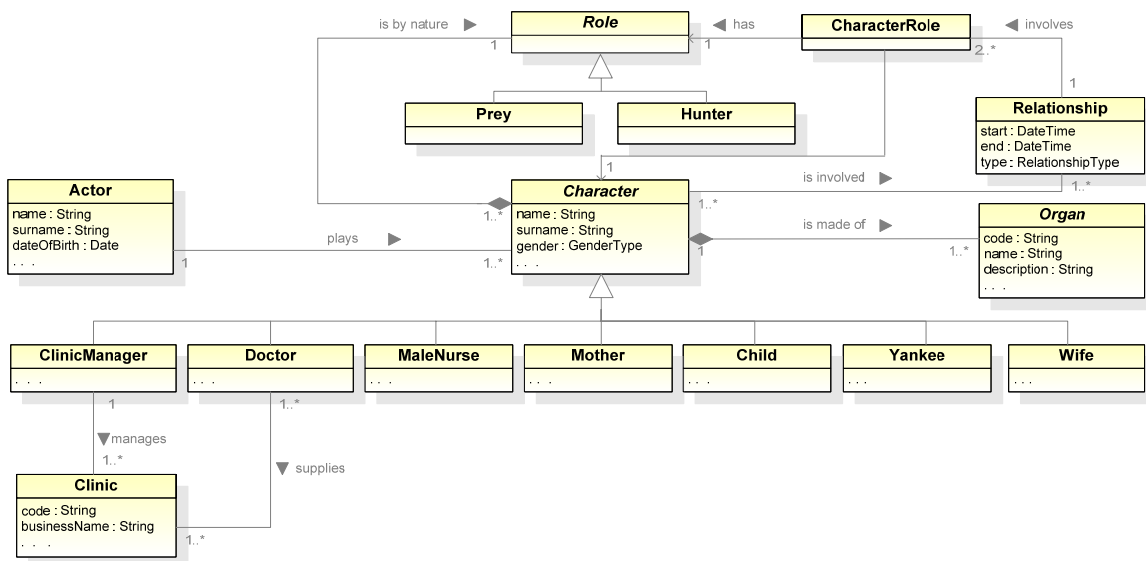
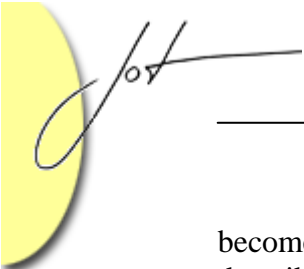


Figure 3. Segment of domain object model of to the main characters (actors), showing the main relationships

The diagram in the figure 3 depicts, using the class diagrams notation, “entities” present in the domain of study, including their main relationships.

Each Actor plays one or more Character instances. This entity is an abstract class (in italics in the diagram in accordance with UML notation), and as such it does not have its own life. Abstract classes need to be extended by specialising classes called “children”. They are, for example: Doctor, MaleNurse, Mother, etc., which, being concrete classes, can have their own life. Each Character, as expected, is made of a series of Organ elements (another abstract class), and, therefore, is a potential prey. In the class diagram, the Organ entity is specialised (inheritance relation) in type of organ, a fundamental part of the macabre price lists (see figure 4).

Every Character, has an intrinsic Role: Prey or Hunter. For example, children are intrinsically prey; the MaleNurse, on the other hand, is by nature a Hunter. In spite of this, each Character embodies an actual Role only when he/she relates with the others. In the diagram the Relation class models this aspect. In particular, each Character is involved in a series of relationships with others, and, in each relationship, it can embody a different role. (This part of the model is illustrated by an example in the next object diagram). Moreover, the role adopted by a Character towards another one is not fixed, but is an entity that can change over time. The most striking example is the relation Doctor/Child. The Hunter Doctor inevitably



becomes *Prey* of those children he has been hunting for a long time. This element is described within the *Relation* class, by specifying its length and the kind of relation attribute: parent, child, business, hunting, etc.

In order to describe in details the part of the class model relating to the characters involved in the various relationships, we used an object diagram. A particularly confusing practise in the software industry is to call “class diagrams” “object diagrams”. For example a domain object diagram, which contains classes, should be more appropriately, called business static model.

One of the much appreciated features of UML is the so-called type-instance dichotomy [ORGUML]. It is a mechanism that allows practitioners to represent both general aspects and concrete elements derived from them. The classic cases of type-instance dichotomy are provided by the following pairs: classes-objects, parameters-values, etc. The *object diagram* is a variation of the class diagram, and the notation used in the two diagrams is very similar. *Object diagrams* represent a snapshot of the system made in a precise moment in time of a hypothetical configuration. Unlike class diagrams which are made up of abstract elements like classes and interfaces, object diagrams are populated by objects (in UML names and type of objects are underlined) “suspended” in a particular state. This is a dynamic concept which, in a precise moment, is given by the value of all its attributes and the relations with other objects (which are, in the end, still particular attributes that store references to other related objects).

The diagram in figure 4 describes the relationships between two characters: *Doctor* and *Child*, who are, intrinsically, *Hunter* and *Prey*. The left side of the diagram depicts the first relationship involving both characters. It occurs before the crusade and therefore *Doctor* and *Child* obey their own temperament, so the *Doctor* is the *Hunter* and the *Child* is the *Prey*. The relationship on the right side of the diagram, on the other hand, exists after the children rebel, so the values are diametrically opposite: the children become hunters and the doctor ends up being the prey.

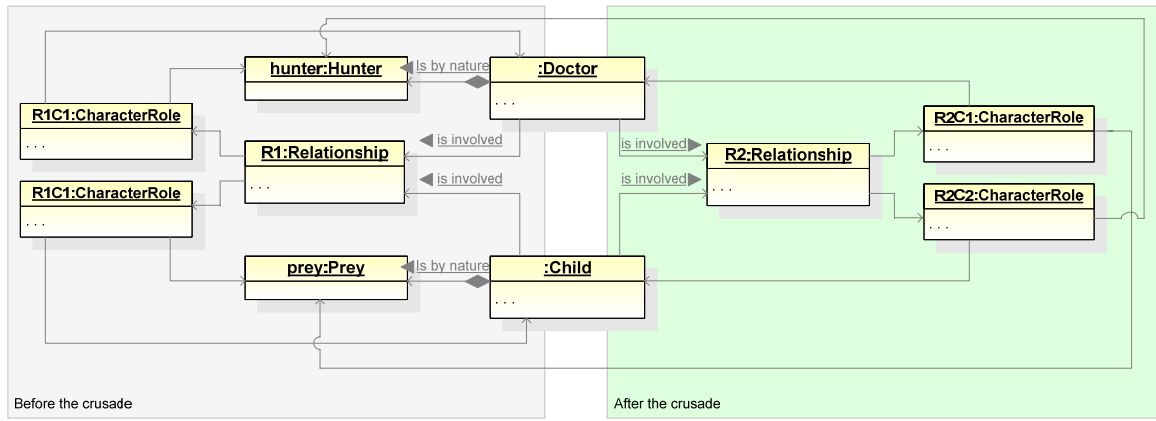


Figure 4. Object Diagram related to the part of the model inherent in the relation among the characters.

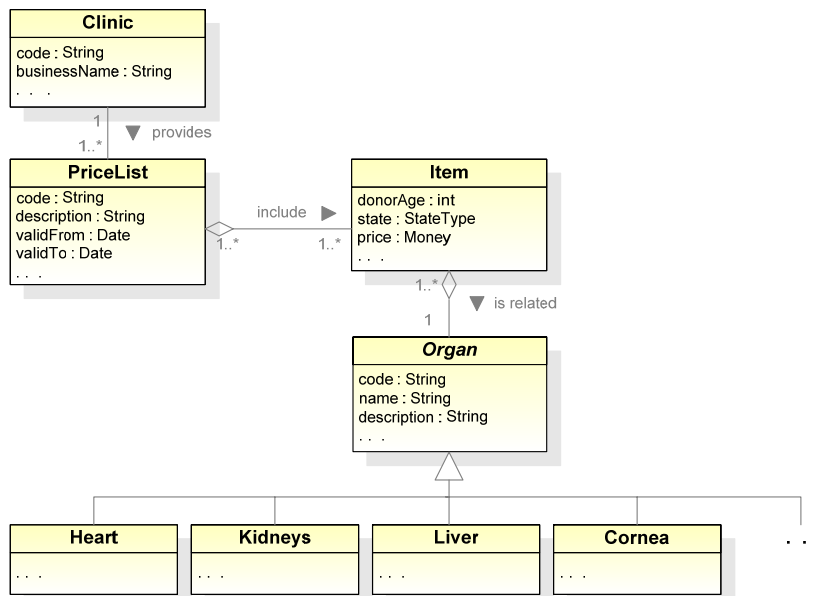


Figure 5. Representation of the Clinic's price lists

The diagram in figure 5 depicts the macabre price lists used by the Clinic. In particular, each `PriceList` object contains a description, the validity window and, obviously, the actual price of a series of items, each of them relating to a specific `Organ`.

Although the `Doctor` is a specialist in removing corneas, during his monologues he makes reference to the “mother piece”: the heart. Furthermore, we may consider the doctor as a piece of the big picture, a provider of the macabre price list patchwork of the clinic.

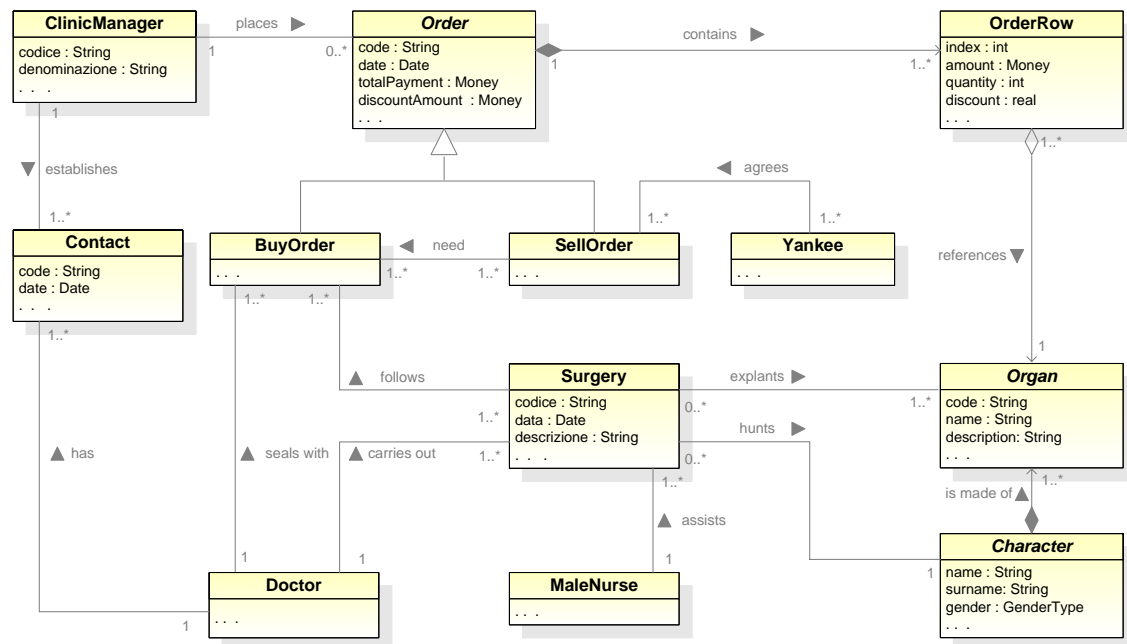


Figure 6. Representation of the order object graphs

The diagram in figure 6 depicts the entity Order. This is another abstract class, made up of a series of OrderRow elements. Each of them includes some information and the agreed price for each requested Organ. Order is represented by an abstract class, since there are two specialisations: BuyOrder and SellOrder. The former sanctions the agreements between the Clinic and Yankee entities (wealthy subjects with no scruples convinced that their money can buy anything... even their own health). The latter is, instead, the result of the agreements that the ClinicManager makes with other parties. In this context, he contacts the Doctor several times to get certain organs with well-defined requirements. Each purchase order is then followed by one or more operations, made by the doctor, aided by the MaleNurse. They are necessary to remove organs from their prey, typically children... But each person is a hunter or a prey!

State Machine Diagrams

During the requirements analysis stage, Business Analysts often need to fully specify the complete life cycle of well-defined entities. In order to achieve this result, they use the UML statechart diagram which allows them to show the different stages that an entity passes through during its existence and the events that trigger the passage from one state to the next. For example, a cash machine includes the following states: out of service, initialisation, card waiting, PIN acquisition, stand-by, etc.

In the context of a theatrical script, for example, modelling the life cycle of some characters could be useful, such as the doctor or the male nurse as shown in the following figures.

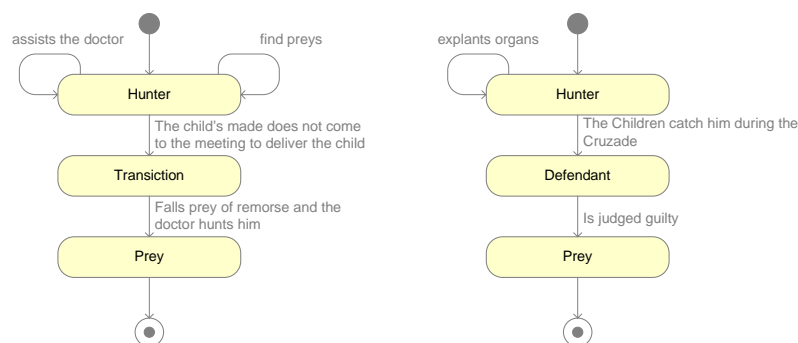


Figure 7. Life cycle of the Male Nurse and Doctor entities (characters)

Each character has its own role by nature, but when it builds relations with others, it might change its nature. For example, the male nurse, belonging to the circle of hunters, ends up being hunted by a stronger hunter: the doctor!

Conclusions

This experiment was a one-of-a-kind experience... Although UML, since its inception, was designed to be used in a number of different domains, probably even Grady Booch, James Rumbaugh e Ivar Jacobson, who designed the first version of UML, could never have imagined that it could have been applied to art, as this study proposes.



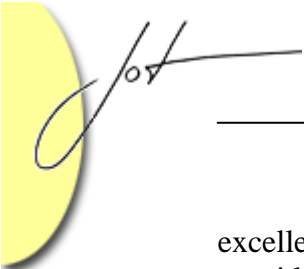
The experiment, though stirring, captivating and worthwhile, raises several issues, with implications and lessons for the classic software development project. It was a tough journey, beset with difficulties, mainly due to the fact of two different realities getting close to each other, given that the artistic tradition is intrinsically and deliberately uncomfortable with systems, schemes and rules, in their formality and precision more typical of the discipline of engineering. It was necessary to include the creativity in the classic UML elements and to develop the paradigm according to the Object Oriented laws.

However, the exuberant artistic stream led to creative diagrams, after some initially unsatisfactory experiments. They have the advantage of alluding to places, people and events mentioned in the text and at the same time to manipulate thousands of daily reality variants, inviting the viewers, each with his or her own personal knowledge about the theme that the text deals with, to think over and interpret the diagrams able to synthetically reveal the incomparable efficiency of the criminal system. The diagrams are able to describe certain situations as a whole or precise moments of the event, providing the spectator/beneficiary the necessary instruments to reconstruct on his or her own the dramatic progress of the fictitious or real text.

The formal analysis of the text using UML formalism, then, is not limited to mere productions of diagrams and modules as an end in itself, only able to raise unfruitful mental lucubration. Far from it... This experiment allowed us to show that many aspects of the use of UML, as in the case of theatre semiology, facilitate a better understanding of words, free from dogmatism, and away from any emotional involvement and, at the same time, flexible enough to leave room for creative expression of the people taking part in the development of the project. UML/O.O. and semiology allow us to analyse a dramatic script using different stages of abstractions, from the deep structure to the surface structure and/or vice versa. So identifying the discourse by which it is possible to set the system of textual signals necessary to the dramatist and to the director to make a meaningful system, is easier. Moreover, they are the basis for studying the existence of possible information systems used as integral part of staging. Carlo Caloro tested the hypothesis by carrying out an installation in which the spectator is let free to interact with the dramatic side of the text by using special information systems. The innovative scenery provided new stimuli and incentives for the traditional methods of analysing words... However, this is subject of another article.

Use cases, including their specification, produced during the first stage of text analysis are limited only by the structure of the text. These descriptions, once completed, provide an excellent reference to those who, afterwards, manage the direction of a play. This information can be used to analyse the surface and discourse structures of the text facilitating the writing of a script and providing a point of reference for directing actors. So, it contains all the instructions deduced from the static and dynamic modelling of the text architectural platform, producing a distinction between the list of related instructions necessary for actors interacting amongst themselves and the list of other codes composing the performance.

The formality of the various models, such as the domain object model, while on one hand is constraining because of the intrinsic formality of the O.O. notions, encapsulated in each graphic element; on the other hand, allows us to investigate in details entities existing in the text, characters, corresponding relations, etc. They are



excellent tool of investigation, though often not uncomplicated and requiring a considerable initial time investment... However, the time spent would likely be spent in just the same ways, maybe less formal and less easily reusable, in order to deepen the knowledge of the script. In the end it would probably require more time, as often happens in designing software systems, where wrong or late examinations mean more time is wasted. It takes time to produce the various models, but, once carried out, they help to clearly and unambiguously underline the different projections which can be used to analyse the same object.

Appendix A

The following brief paragraph is a short description of the text [CRZ2001].

A doctor, helped by his male nurse, trades in human organs, removed mainly from innocent children. These organs are sold to him by their desperate parents. The doctor, after striking the bargain, immediately takes the knife. He has his own philosophy of life, according to which the world is a huge market place crowded with sellers and buyers and nothing else. The world is, then, a huge hunting ground, the street "is" the hunting ground and children in the street are "prey" at the mercy of whoever catches them. He reduced his knowledge to hunting, merely a job to him, catching and dismembering his prey on at a much higher price. Children are "prey", "goods", a heart is the "big shot" and cornea are in demand in European and American markets. He feels similar to the Nazi accountants counting deportees' teeth and hair; his actions are based only on money and ideology: money makes the world go round and his knife is nothing else than a small cog in a wider and more complex mechanism... This self-justification would seem to save his conscience.

The male nurse, on the other hand, does not have the same ability of philosophical abstraction: he is poor and miserable, though with a grain of conscience left... So the inevitable happens: a sudden mixture of fear and uncurbed remorse end up reviving in his mind the sad look of the hare-children he immobilizes on the couch before being cruelly torn apart by the knife. He can hear the joyful sound of hares emerging from the drains and feels that something is changing. Soon, maybe, the chips will be down. The male nurse, then, tells everything to the doctor's wife, who, unaware until that moment and deliberately blinded by wellness, demands to know about her husband's awful job. He tells her that he has already taken measures against the male nurse, since you can be either "hunters" or "preys". If you stop being a hunter, you automatically become a prey.

At the end, this law of contrappasso (i.e. retaliation) is applied also to the doctor himself. He ends up being a prey in his turn. Before the Court of children, he uses disquisitions and philosophical-scientific speculations, in order to justify his deeds, but it is no use.



REFERENCES

- [CRZ2001] “La Cruzada de los Niños de la Calle” - Claudia Barrionuevo, Dolores Espinoza, Christiane Jatahy, Iván Nogales, José Sanchis Sinisterra (coord), Arístides Vargas, Víctor Viviescas - Sociedad General de Autores, Madrid 2001
- [PKPK2003] Philippe Krutchten, Per Kroll, Contributor Grady Booch:.”The Rational Unified Process Made Easy: a practitioner's guide to the RUP”, 2003
- [Jacobson1992] Ivar Jacobson –“Object Oriented Software Engineering. A Use Case Approach”, Addison Wesley
- [IEEE1998] IEEE Recommended Practise for Software Requirements Specifications IEEE Std 830-1998
- [OMGUML] <http://www.uml.org/>
- [LVT2001] Luca Vetti Tagliati: “UML e ingegneria del software. Dalla teoria alla pratica”, Tecniche Nuove, 2001
- [Greimas2001] Algirdas Julien Greimas : “Del Senso”, Bompiani, Milani 2001
- [JBR2000] Ivar Jacobson, Grady Booch, James Rumbaugh: “*The Unified Software development process*” – Addison Wesley, 2000
- [ARGR2001] Frank Armour, Granville Miller – “*Advanced Use Case Modelling. Software Systems*”, Addison Wesley

About the authors



Carlo Caloro collaborates, as an actor and director assistant with different theatre companies, and as a professor with different institutions in Italy and abroad. In 2001 he got a diploma in audiovisual medias with KHM, Medial Arts Academy in Cologne (Germany). In 2004, he got a diploma in direction with Accademia d’Arte Drammatica “Silvio D’Amico” in Rome, where now he is collaborating to the course of Direction given by the M.o Domenico Polidoro, as a researcher in the field of new technologies applied to plays.



Luca Vetti Tagliati works in the City of London as a Senior Technical Lead at Lehman Brothers UK and he is a part-time PhD student at the Birkbeck University of London. His professional career spans more than 15 years, the last 7 have been devoted to the financial domain. Over the last few years he had specialised in component based software systems and SOA.