Analysis of Interactive Information Systems Using Goals

Pedro Valente and Paulo N. M. Sampaio
University of Madeira
Portugal

1. Introduction

The high competitiveness in the world markets lead enterprises to provide their clients with the best services and products as possible in order to obtain important advantages over their opponents. These services and products are result of the enterprise's business processes (BPs) which, therefore, need to be improved.

The improvement of BPs can be achieved, both by, reorganizing their tasks, or by automating (completely or partially) these BPs by means of the development of interactive information systems (IISs) which articulate the work of every actor, thus improving speed and reliability of the goal(s) (of the BP) to be achieved.

ISSs are computer based software systems (Land, 2002) that have the ability to manage structured information which can be manipulated by humans by means of user interfaces in order to perform their tasks within the enterprises' BP. Since numerous BPs need automation, the development of IISs must be planned in order to best schedule the deployment of new and existing improved services and products, according to the needs of every stakeholder and available resources.

The successful development of IISs is usually a complex and demanding task that can only be achieved if a project is organized in such a way that every stakeholder is able to negotiate its intentions in terms of functional (and also non-functional) requirements. Once these requirements are implemented in an acceptable price, they will bring an added value to the enterprise, enhancing its overall business effectiveness and efficiency, and therefore contributing to ensure its wealth and survival in a demanding market.

The precise identification of functional requirements (FRs) is a crucial task for the fluent development of a project, and can be carried out during the organization of new or existing BPs if every stakeholder is able to express its point of view over the problem and if a final agreement is achieved. Following the identification of FRs, the implementation effort should be estimated, the requirements analysed and the system designed in such a way that future developers have no doubts on its implementation, improving the probability that the system is produced on schedule and with the fewer mistakes as possible.

The work presented in this chapter is based on Goals (Valente, 2009), a software engineering (SE) process proposed in order to provide the needed tools to define the precise conception of an IIS. Goals' first phase (requirements) defines the FRs of the IIS based on the design of the automated BP using Process Use Cases (PUC) (Valente & Sampaio 2007a). Goals' second phase (analysis) consists in the analysis of these requirements producing use cases design and a comprehensive architecture, using MultiGoals (Valente & Sampaio 2007b), which can be used to define implementation precedences and development tasks assignment. The artifacts produced by both requirements and analysis phases can be integrated with Interactive Use-Case Points (iUCP) (Nunes, 2010) to estimate project effort. MultiGoals can be further applied to complete the design (third phase) of the complete IIS including support for multimedia design.

1.1 Goals' requirements and analysis phases

Goals is a (business) process for the production of the correct IISs and can be applied for the resolution of a specific information problem. This process is defined into 6 different phases following a standard construction process: (i) requirements definition, (ii) analysis of the problem, (iii) design of the solution, (iv) development, (v) test and (vi) installation of the finished IIS. Goals also predicts that the software will need maintenance following two possibilities: (i) introduction of new requirements, in which situation the complete process will be followed again, and, (ii) corrective maintenance in which case the process is verified from the beginning in order to identify where the mistake took place during the conception.

The first three phases of the development process, which are the phases defined by Goals (requirements, analysis and design) are presented in Figure 1.

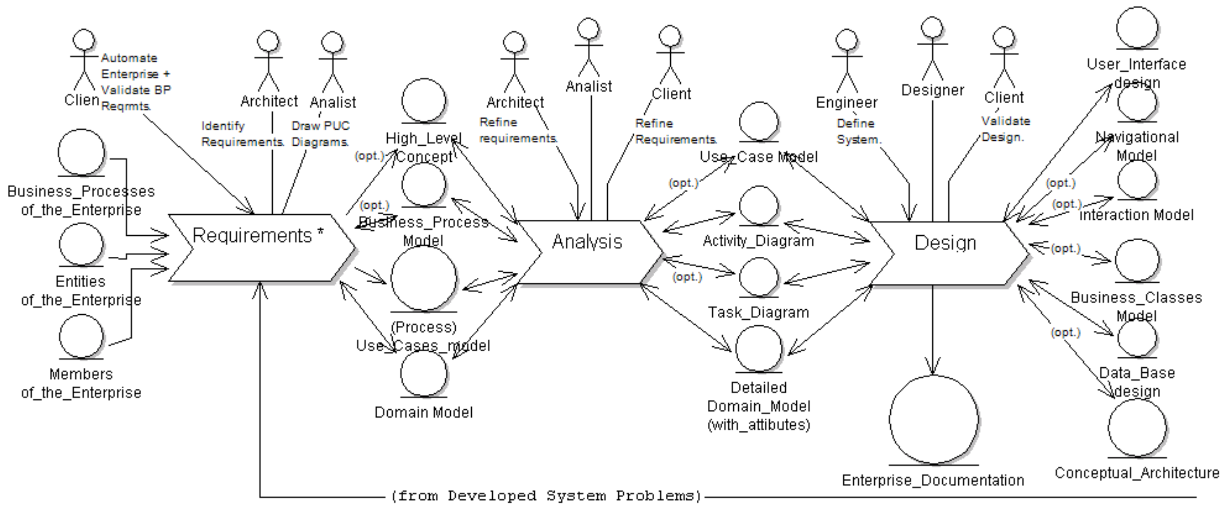

Fig. 1. Goals' phases of Requirements, Analysis and Design.

Each phase of Goals is a business process itself in which a different methodology should be applied to produce information for the construction of the IIS. Although the Goals process is independent from the methodologies used, some restrictions should be observed in order to achieve the minimal quality for the global process and assure that full advantage is taken from the available inputs and that the needed outputs are also produced. Also, each phase defines: the human intervenient and their objectives, the minimal set of information inputs, and the outputs for the next phase.

The next sections describe the development and integration of the first two phases: (i) requirements, in order to produce functional requirements, and (ii) analysis, in order to produce the system´s architecture and effort estimation.

1.1.1 Requirements phase

The methodology chosen for the phase of requirements should be: (i) use case-oriented, in order to identify (essential) use cases (Constantine, 2002), and (ii) information-oriented, in order to produce a Domain Model (Nunes, 2001).

This phase is triggered by the client with the intention of automating the enterprise regarding the resolution of some information problem, and can take advantage of artifacts that might already exist in the enterprise such as: business processes; information entities, or; organization of the enterprise.

The following artifacts are defined as the minimal set of information to achieve functional requirements definition: (i) Use Cases Model (or equivalent, identifying all use cases) - the use cases of the system, and; (ii) Domain Model - information entities of the enterprise. Optionally a High-Level Concept (Kreitzberg, 1999) and a Business Process Model (Eriksson & Pencker, 2001) identifying the enterprises' goals can be elaborated.

In Process Use Cases, the methodology suggested by Goals for the requirements phase, architect, analyst and client work in order to produce the needed output elements: High-Level Concept; Domain Model; Business Process Model, and; Process Use Cases Model.

1.1.2 Analysis phase

The methodology chosen for the analysis phase should be: (i) use case-driven, in order to detail the previously identified use cases, and; (ii) information-oriented, in order to detail the previously identified entities.

The following artifacts are defined as the minimal set of information to achieve comprehension of the problem: (i) an Activity Diagram (or equivalent) of the decomposition of the use cases into user intentions (or equivalent, representing user tasks) and system responsibilities (or equivalent, representing system behaviour), and; (ii) a Domain Model. Optionally a Use Cases Model, a Task Model (Paternò et al., 1997) can be elaborated, and an implementation effort estimation method applied.

In MultiGoals, the methodology suggested for the analysis phase, architect, designer and client work to produce: (i) a Use Cases Model; (ii) Activity Diagrams; (iii) an Interaction Model, (iv) an Application Domain Model, and (v) a System Architecture.

iUCP, the method presented in this chapter in order to estimate project effort, takes advantage mainly from the use cases and actors identified in the requirements phase and the System Architecture produced in the analysis phase to calculate the number of man-hours needed to finish the construction of the IIS.

This chapter presents a comprehensive illustrated example of the application of the previous methodologies to define, in a straight lined process: the project's concept, the BPs design, the information entities (in a domain model), the use cases design, the systems' architecture and the effort estimation (integrating with iUCP). To support the understanding of these contents the following definitions should be observed:

- activity – action performed by humans within an enterprises' organization, carried out or not, in interaction with a system. Activities are a generalization of the concept of use case (actions performed in interaction with a system).
- (essential) use case - specially structured form of a use case that represents a single, discrete and well defined task over a system that is complete and meaningful. Use cases should be described in abstract, simplified, and implementation-independent terms using the language of the domain understood by the users (Constantine & Lockwood, 2000).
- actors - humans that play a role in an enterprises' organization, performing at least one activity.
- interaction space - class that represents the space within the user interface of a system where the user interacts (...) to carry out some particular task or set of interrelated tasks (Nunes, 2001).
- task – class that models the structure of the dialogue between the user and the system in terms of meaningful and complete sets of actions required to achieve a goal (Nunes, 2001).
- entity - a class used to model perdurable information (often persistent). Entity classes structure domain (or business) classes and associate behavior often representing a logical data structure (Nunes, 2001).

The methods presented in this chapter are directed to software engineers and business process managers and have been developed based on its application in real projects mainly in the Software Applications Development Office at University of Madeira, a software development environment characterized by demanding terms, high information accuracy and usability user standards.

2. Requirements phase

The precise identification of functional requirements and its acceptance by all the stakeholders of a project is a key factor for the correct conception and success of an IIS, and user participation in the development life cycle can be seen as critical to achieve usable systems and has proven its efficacy in the improvement of systems appropriateness.

This section presents Process Use Cases (PUC) (Valente & Sampaio, 2007a), a methodology that defines the steps to achieve the identification of functional requirements in terms of use cases as a sequence of the organization of new or existing BPs and also identify the initial information entities and actors involved in an IIS.

2.1 Process use cases steps for requirements definition

PUC is a methodology defined within Goals, and is a Requirements Enginnering (RE) solution to bind the phases of requirements and analysis rapidly through the identification of use cases and information entities during the organization of BPs.

In order to achieve automation of the BP, PUC covers partially the lifecycle of Business Process Management (BPM) (Figure 2) (Webinterx, 2006) assuring that the BPs are analyzed, improved and modeled before they are automated (monitoring is out of the scope of PUC). The "analysis" is understood as the inspection of the current workflow of the BP, the "improvement" is the reorganization of the BP in a way that it becomes more efficient and/or more effective. After the "improvement", the BP is "modeled" and finally it is "automated" by a SE process that leads to the development of an IIS.

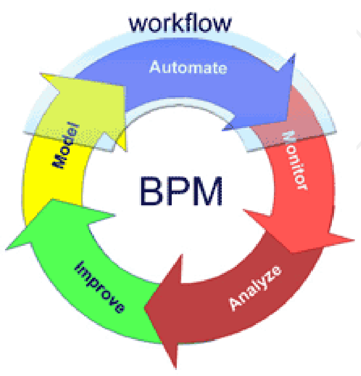

Fig. 2. Business Process Management lifecycle.

PUC describes the development of 4 artifacts: 1 statement and 3 models (respectively High-Level Concept, Domain Model, Business Process Model and Process Use Cases Model) using an information-oriented strategy for the identification and association of the components generated: business processes, information entities, actors and use cases.

Consider Table 1 which enumerates the steps of PUC. Each step has a name (Interiorize Project, Information Identification, etc...) and produces one artifact (High-Level Concept, Domain Model, etc…) that is manipulated by an intervenient (architect, analyst and/or client) towards components definition (entities, business processes, etc…).

Step	**Step Name**	**Model Name**	**Components Manipulated**	**Intervenient**
1	Interiorize Project	High-Level Concept	N/A	Architect, Client
2	Information Identification	Domain Model	entities	Analyst, Client
3	Business Processes Identification	Business Process Model	business process, entities, actors	Analyst, Client
4	Use Cases Identification	Process Use Cases Model	tasks, use cases	Architect, Analyst, Client

Table 1. Steps of Process Use Cases methodology

In order to illustrate PUC, a project ("Gastronomy Project") developed for a small enterprise is presented. This (non-profitable) enterprise related to a local governmental library (in Madeira, Portugal), is responsible for the bibliographic investigation on gastronomy. The idea of the director is to divulgate the gastronomic events promoted by the enterprise and the existing gastronomic recipes in a website. However, the budget for the project is reduced and the software development should be kept to its minimal.

The enterprise had three informal units, besides the business manager, responsible for the execution of its main activities: the secretary (responsible fot the contact with the clients), the investigation unit (responsible for aquiring and cataloging recipes), and the events unit (responsible for the organization of events). After a first approach, in which an attempt was made to understand the main activities of the enterprise, it was possible to understand that the enterprises' main products were: the identification and cataloging of gastronomic recipes and the organization of gastronomic events.

The application of the steps of Process Use Cases to this project and the more relevant artifacts produced are presented in the next sections.

2.1.1 PUC Step 1 – Interiorize project

The High-Level Concept (HLC) (Kreitzberg, 1999) is a technology independent paragraph that describes the part of the system (or full system) that is going to be implemented. The HLC must be understood by all the stakeholders (the community) of the project promoting a shared vision that will help the project community to keep focused on the product development. The Interiorize Project is the only unstructured part of PUC.

In this step client and architect agree on a HLC for the project. To do this, it is important to understand the scope of the project within the enterprise's global activity, so, it is necessary to understand how the enterprises' activities lead to the production of its main product(s) and what is the strategic reason that leads to the need of automation. Artifacts such as the enterprise's hierarchical organization and legislation may provide important information.

In the example project presented in this section the HLC agreed with the client was: "**Capture the attention of potential and actual clients for the gastronomic patrimony and events of the enterprise**.". The HLC expressed the intention of the enterprise to enlarge the number of clients and promote client fidelity by providing a quality service of information that combined the traditional historical recipes and the events that promoted those recipes.

2.1.2 PUC Step 2 – Information identification

Information is very stable within an enterprise. Mainly, information manipulated by core business processes is persistent from the birth of the enterprise until its closure and is independent from the technology used to manipulate it. Information is usually manipulated by several BPs of the enterprise, that once correctly identified (both information parts and BPs) will produce valuable artifacts for the Business Process Mangagement and Software Engineering activities.

In this step, the analyst identifies the main "concepts of information" defined in the HLC. These information concepts are transformed into entities that will be the first ones in the Domain Model, the output of this step. Entities represent information (not actions, actors, nor business processes; but the name may coincide) and relate to each other by the composition of a meaningful structure. This structure has relations of hierarchy (inheritance), dependency (composition) and possession (association) and is called Class Diagram as defined in UML (Object Management Group, 2003). In PUC, the entity stereotype is used instead of the class stereotype which at this stage is a more accurate concept of information.

The Domain Model can be used along all the Software Engineering process. At implementation stages, it is often used to generate database tables and (programmed) classes to manipulate these entities. The Domain Model must be updated at any stage in the process when new entities are revealed.

The Domain Model is defined based on the information entities identified in the HLC statement. These information entities are placed in the Domain Model relating to each other according to the natural relation between information entities using the relations of the UML's Class Diagram, being their cardinality also defined. Within PUC, and after this first step, the Domain Model will be updated whenever new information entities are identified, speccially during the modeling of the Business Process Model (Step 3).

In the example project presented in this book, the first entities derived from the High-Level Concept concepts of information were: "client"; "recipe" and "event". The entity "client" existence, although implicitly related to the events, was reinforced when we noticed that the business process for recipe capture also involved donation of recipes by "clients". The first entities identified were then combined with other entities ("Advertisement", "Producers" and "Recipe Submitted for Approval") identified in Step 3 (Business Processes Identification) to compose a single information structure as presented in Figure 3. It is suggested that the analyst describes the Class Diagram in natural language to the client in order to achieve diagram validation.

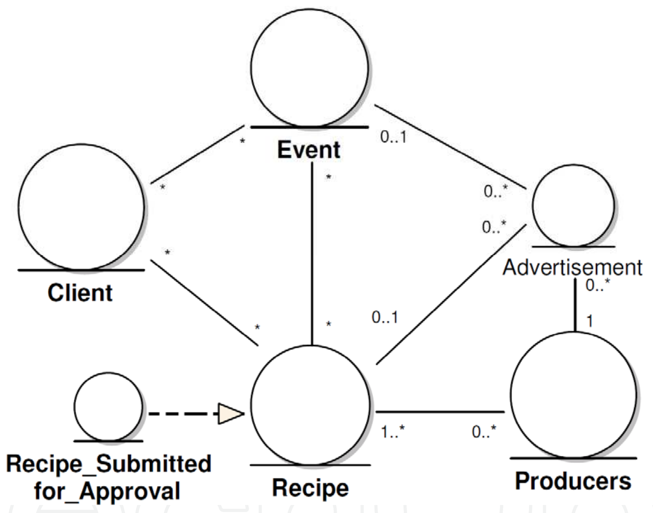

Fig. 3. PUC Step 2 - Domain Model.

2.1.3 PUC Step 3 – Business processes identification

The objective of this step is to identify business processes (BPs) for possible automation based on the information entities identified until this stage. At the same time, valuable information that can serve as documentation for future Business Process Management activities is being produced.

BPs exist in an enterprise to achieve a certain objective, a goal, a product, that can be described by information (associated with this product). BPs happen as many times as the need to give response to the needs of some enterprise member or third party (e.g. client) with some responsibility (active or passive, with some relation to the enterprise) within the activity of the enterprise. Many enterprise members can interact with these processes by carrying out some complete, unitary task, in which many different entities can be manipulated (consumed or produced). In order to be able to control (e.g. reorganize) these BPs, it is important for an enterprise to maintain complete and detailed information of relations among BPs, their inputs, outputs, actors and triggering events.

In this step, analyst and client will identify, relate and detail BPs. The identification of BPs should take place, at least, from the business unit (in an enterprise organization hierarchical perspective) "directly" responsible for the information being managed, i.e. unit(s) that consume or produce this information to achieve complete and meaningful tasks. BPs that relate "directly" to the information identified until this stage must be documented, if within the scope of the project defined in the High-Level Concept, in order to provide the understanding of all the manipulation made over the identified information.

BPs are named according to their goal, i.e., the product of the BP, whether it is a service, information or a material product (e.g. product "television", BP name "build TV"). The outputs and inputs (information, resource and output in the Business Process Model (Eriksson & Pencker, 2001)) are represented by entities. When the flow is towards the BPs it represents an input (and generates an event) and the contrary direction represents an output. Associations can be bi-directional representing event, input and output. Actors are associated with BPs using "associations" and their objectives are written in natural language (e.g. "approve recipe") separated by a plus signal (+) naming the association. When an actor triggers the BP, an event is generated and its relation with the BP is represented with a flow (arrow form). BPs can be related to each other, i.e., the outcome of a BP (which is an event) serves as the income to the next one.

In the example project presented in this section, four BPs that directly manipulated the entities "client"; "recipe" and "event" (Step 2) were identified: (i) the "Obtain Recipe" BP in which donators and investigators donate recipes that are evaluated by a gastronomy consultant; (ii) the "make event" BP that generates information for the entity "event", in which business manager and the event organizer interact to create a new event using the "producer" and "recipe" entities; (iii) the "advertise" BP which was created in order to produce information for the enterprises' future website, in which the business manager delivers advertisements to the advertiser about recipes, events or generalized news; and, (iv) the "obtain gastronomic information", which is a new BP that will exist as a consequence of the new website and represents the usage of that website by the clients of the enterprise. The relations between the identified BPs, the actors involved and information entities manipulated are illustrated on Figure 4.

2.1.4 PUC Step 4 – Use cases identification

The documentation of BPs in a language that every intervenient (stakeholders of a project) understands is important to enable correct dialogue over the actors, activities and goals of the BP. BPs can be partially or completely automated or not automated at all.

The identification of use cases is the purpose of this step. The BPs identified in the previous step will now be detailed using an Activity Diagram (Object Management Group, 2003) in which the activities that need automation will be transformed into use cases providing the projects' functional requirements.

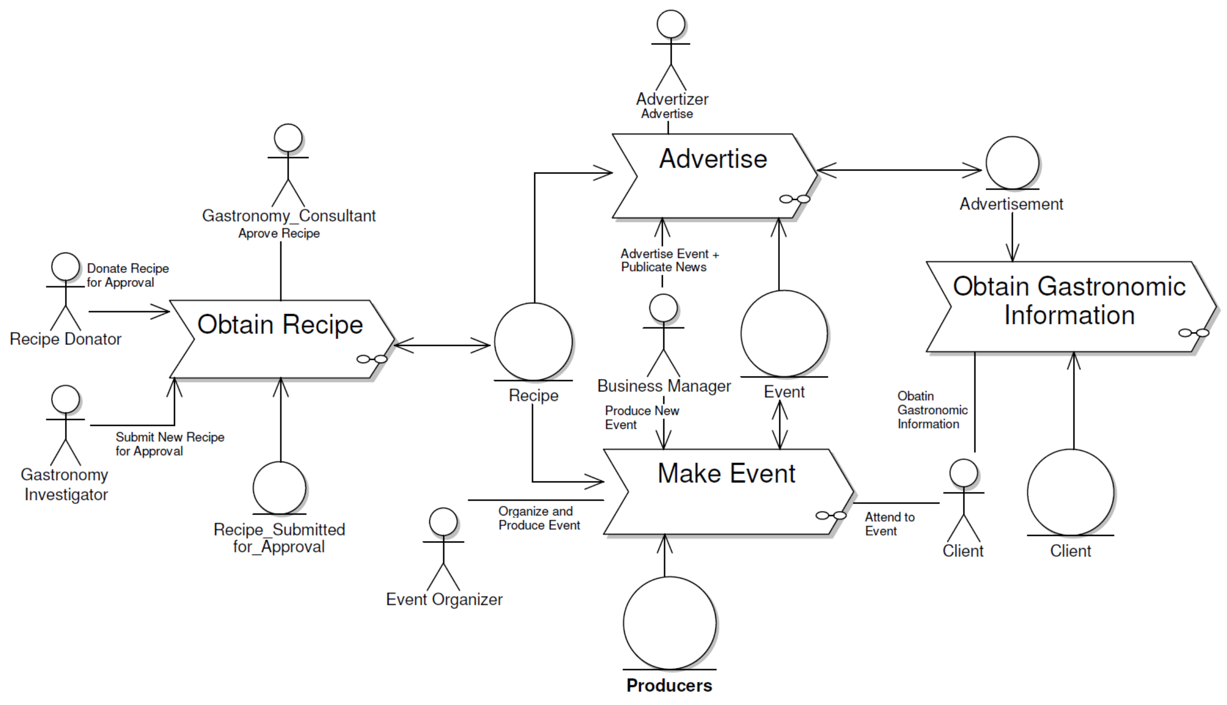

Fig. 4. PUC Step 3 - Business Process Model for the Gastronomy Project.

In this step, analyst and client model the activities and use cases of the BP which will be performed by the actors until achieving the targeted goal. The BP is designed with the Process Use Cases Model, through the use of an UMLs' Activity Diagram with swimlanes. The UML's activity stereotype is used to represent actions of the BP which are not use cases. Fork and Decision can be used to represent parallel activities and decision points, respectively.

Once all activities are identified, it is important that the architect (with the client) decides which activities should be automated. When this happens, a use case (stereotype change) takes the place of that activity.

In the example project presented in this section, four Process Use Case models were designed following the identification of the four BPs relevant for the project identified in Step 3, from which we chosen the "Obtain Recipe" to illustrate the Step 4 of PUC in Figure 5.

2.2 Process use cases basis and related works

Different abstractions provided by different techniques are used to represent the information acquired within PUC. These techniques are: UML (Object Management Group, 2003) that provides the modeling notation; Wisdom (Nunes, 2001) that provides the basic concepts for the RE process, by means of the "Requiremens Workflow"; the High-Level Concept (Kreitzberg, 1999) a concept used in PUC without changes; the Business Process Model (Eriksson & Pencker, 2001) that provides the (adapted) notation used in PUC for modeling BPs and their interaction with users and information, and Usage-centered design (Constantine, 2006) that provides the definition of (essential) use case and the basis for the definition of actor.

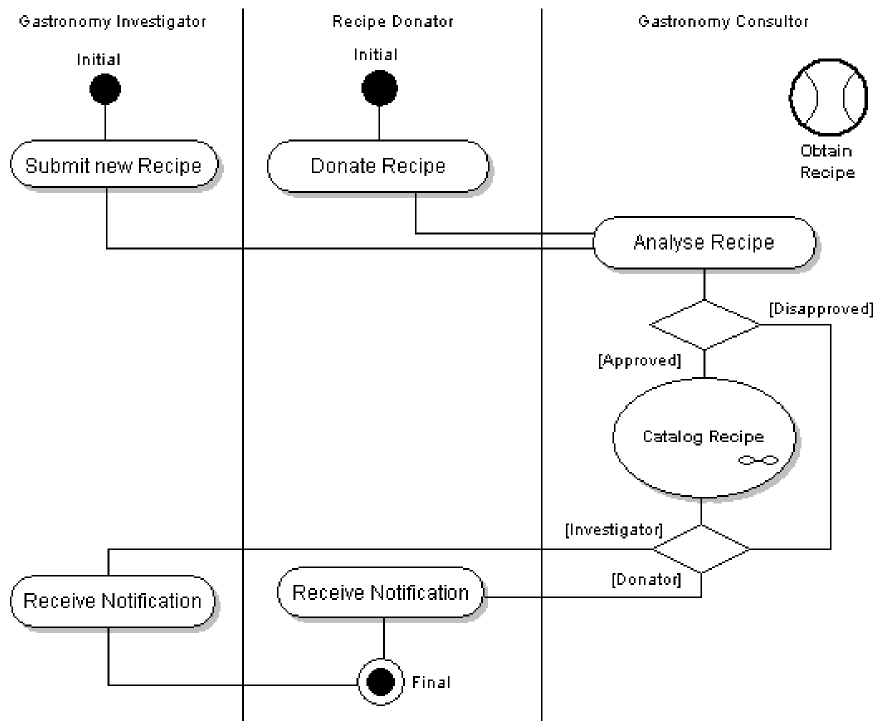

Fig. 5. PUC Step 4 - Process Use Cases Model for the "Obtain Recipe" BP.

The following RE related works also identify use cases as a result of the analysis of business processes: (González & Díaz, 2007) proposes an extensive approach which defines business strategy, business and IT infrastructure to produce a Business Process Goal Tree from which the goals that need automation (use cases) are derived; (Shishkov & Dietz, 2005) that derives use cases from the description of business processes based on the construction of norm sentences; Usage-centered design (Constantive, 2006) that represents the relevant activities (BPs) and interrelationships among them (Activity Map), caracterizes each activity (Activity Profiles) and identifies each participant (Participation Map) and their relationships with each other and with the various artifacts involved in the activity, and then extracts (Activity-Task Map) the task cases (essential use cases) from the activities previously identified; (Dijkman & Joosten, 2002) that proposes a detailed procedure to transform business process models into use case diagrams by mapping roles to actors, steps to use cases, and tasks to interactions; Wisdom (Nunes, 2001) that comprehends the steps of "Interiorize Project" producing an High-Level Concept, "Understand System Context" producing a Domain Model and/or a Business Model, "User Profiling" producing a Role Model and "Requirements Discovery" that encompasses finding actors and essential use cases; and (Štolfa & Vondrák, 2006) that proposes that business processes are designed using Activity Diagrams and that a mapping is made between the activities of the business process and use cases, which can be "one-to-one" or "mapping several actions to use cases" by applying the Sequential pattern or the Optional pattern respectively. For a more comprehensive analysis and comparision of the related works on RE please refer to (Valente, 2009).

3. Analysis phase and effort estimation

Following the identification of the functional requirements, it is important that these, which represent development problems, are further analysed in order to better understand user needs and the components that will be needed to support the construction of the IIS (for these users), and how much time will be needed to develop it.

MultiGoals (Valente & Sampaio, 2007b) is a User-Centered Design (UCD) based methodology that inherits from Usage-Centered Software Engineering and Interactive Multimedia Documents (IMDs) design the concepts and techniques needed in order to understand the user, simplify conception of the usage, and conceptually conceive the user interfaces, system functions and information entities that will compose the IIS with support for Multimedia, covering the phases of analysis and design. Interactive Use Case Points (iUCP) (Nunes et al., 2010) is a method that estimates the effort needed to develop an IIS based on the information produced from the phases of requirements and analysis.

This section presents the analysis phase of MultiGoals and how it uses the artifacts produced in the requirements phase, in order to detail the usage of the system, identify system functions and information entities and their dependencies, and how these objects can be complementarily integrated with iUCP to estimate the effort of the implementation of the IIS, therefore allowing the scheduling of the project within the enterprises' activities.

3.1 MultiGoals for system analysis

MultiGoals is a methodology that defines 11 steps for both analysis and complete detailed design of an IIS. Although it was conceived in order to be comprehensive and cover all the aspects of the conception of an IIS, it also provides the flexibility to be partially applied.

This section will illustrate the application of the simplified analysis steps (Steps 1 and 2, highlighted using and *) of MultiGoals (by means of the example "Gastronomy Project" previously used to illustrate PUC), plus Step 3 – Interaction Model, and Step 11 - System (Conceptual) Architecture that defines the dependencies between the objects identified in the previous steps, i.e. User Interfaces, System Functions and Information Entities.

Consider Table 2 that presents the steps of MultiGoals. Each step adopts a different modeling technique (Use Case, Activity Diagram, Interaction Model, etc…) to produce the appropriate component (actor, task, system responsibility, etc…), within a standard Software Engineering (SE) phase and IMD design level, that will lead to the design of the application.

3.1.1 MultiGoals Step 1 – Use cases model

The Use Cases Model in MultiGoals follows the classical semantics and notation for the UMLs' Use Case Diagram (Object Management Group, 2003). In order to specify the Use Case Model, it is necessary to associate actor(s) to the use cases that they perform.

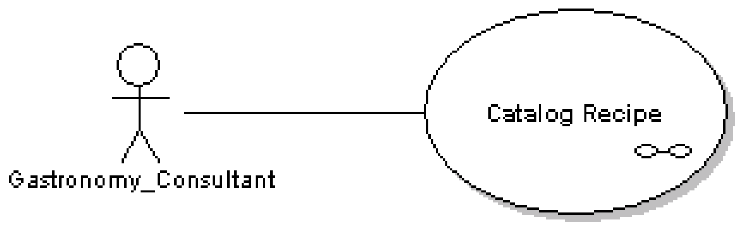

Fig. 6. MultiGoals Step 1 – Use Cases Model (partial) for the "Gastronomy Project".

Step	Step/Model Name	Components Manipulated	SE Phase	IMD Design Level
1*	Use Cases Model	actor, use case	Analysis	Requirements
2*	Activity Diagram	interaction space, task, system responsibility	Analysis	Requirements
3	Interaction Model	task, system responsibility	Analysis, Design	User Interaction
4	Navigation Model	interaction space	Design	Presentation
5	Presentation Model	interaction space, task	Design	Presentation, User Interaction
6	Application Domain Model	entity	Design	Presentation, Content
7	Application Object Model	entity object	Detailed Design	Presentation, Content
8	Conceptual Model	interaction space, task, system responsibility, entity, entity object	Detailed Design	Conceptual, Content
9	System Behavior Model	system responsibility	Detailed Design (Multimedia)	Conceptual
10	Temporal Model	task, system responsibility	Detailed Design (Multimedia)	Conceptual
11	System Architecture	interaction space, task, system responsibility, entity, entity object	Analysis, Detailed Design	Conceptual

Table 2. Steps of MultiGoals methodology

Following the application of the PUC methodology, the Use Cases Model can be directly deduced from the Process Use Cases Model by connecting the actor and use cases that are on the same swimlane. In the example presented in Figure 6, the "Gastronomy Consultant" actor and the "Catalog Recipe" use case were directly derived from the Process Use Cases Model presented in Figure 5 of section 2.1.4 PUC Step 4 – Use Cases Identification.

3.1.2 MultiGoals Step 2 – Activity Diagram

The Activity Diagram will specify how the interaction between the user and the system will lead to the accomplishment of the use case by means of its decomposition into user intentions and system responsibilities, described in abstract, technology-free, implementation independent terms using the language of the application domain and of external users (Constantine & Lockwood, 2000).

The user intentions are represented by the task stereotype and the system responsibilities are represented by the control stereotype. User intentions are tasks that the user wants to accomplish on the system, being most of the times a high level task representing what the user is doing at this step in order to complete his goal (e.g. "Reserve Room"). System responsibilities are the response of the system to the task carried out by the user (e.g. "Confirm Room Reservation").

The Activity Diagram can begin in either side, system or user. Usually, in common cases, 2 to 6 tasks are enough to the user accomplish what he needs. Of course, the number of tasks depends on the complexity of the overall use case purpose. After the Activity Diagram is completed with tasks and system responsibilities, the interaction spaces (user interfaces) in which the tasks will be performed, ant the entities on which the system responsibilities depend must be identified. Notice that one interaction space can support one or more tasks.

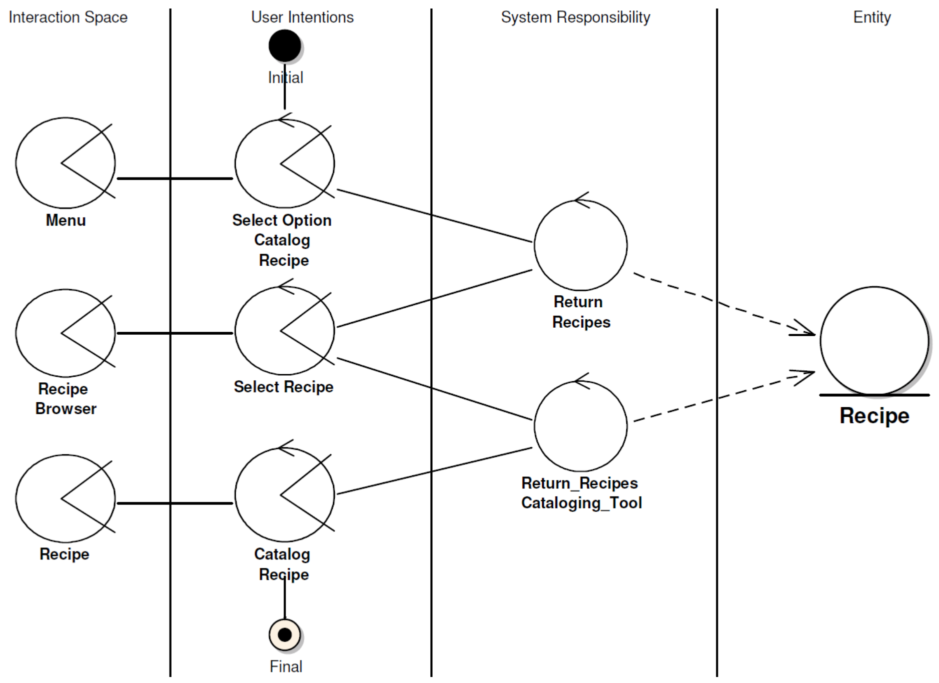

Fig. 7. MultiGoals Step 2 – Activity Diagram for the "Catalog Recipe" use case.

For instance, consider the Activity Diagram depicted in Figure 7. In this diagram, the user intentions initially describe the intention of the user to "catalog a recipe" which is expressed in the "menu" interaction space and immediately carried out by the system returning the recipes in the "Recipe Browser" interaction space so that the user can select the recipe to edit or select a new recipe. After selecting the recipe, the system will return the recipe cataloging tool ("Recipe" interaction space) to the user where the edition of the recipe will be made. The "Return Recipes" and "Return Recipes Cataloging Tool" system responsibilities depend on the "Recipe" entity.

3.1.3 MultiGoals Step 3 – Interaction Model

The Interaction Model details the user interaction necessary in order to perform a user task and specifies which will be the response of the system to each one of the user (sub) tasks, relating these tasks to the interaction spaces where they occur.

The Interaction Model details (decomposes) tasks into sub-tasks, and the corresponding system responsibilities into sub-system responsibilities. The higher level of an Interaction Model is a combination task -> system responsibility taken from the Activity Diagram. Thus, a task is decomposed by means of a ConcurTaskTrees' Model (Fábio Paternò et al., 1997) up to the representation of a physical interaction on the user interface. Similarly, corresponding system responsibilities (which are controls, system functions) are decomposed into lower level controls, which are executed whenever that user task takes place. The sub- tasks are then associated with the corresponding sub-system responsibilities, interaction spaces where the tasks occur are also associated with these tasks, and entities on which the system responsibilities depend are also identified.

The decomposition of tasks into sub-tasks is carried out using aggregation, e.g. "Reserve Room" decomposes into "Select Room", "Select Customer" and "Select Duration". Moreover, an operator also must be specified among the sub-tasks in order to determine their order. These operators can be: Choice (T1 [] T2); Independent concurrency (T1 ||| T2); Disabling (T1 [> T2); Enabling (T1 > T2); Suspend/Resume (T1 |> T2); Order independent (T1 |=| T2). For further information on these operators see (Fábio Paternò et al., 1997).

3.1.4 MultiGoals Step 11 – System Architecture

The System Architecture is the representation of all the relevant components for implementation of the system, and the relations of dependency among them. These components are: interaction spaces (User Interfaces), system responsibilities (System Functions) and entities (Information Entities).

From Steps 2 and 3 of MultiGoals it is possible to extract the interaction spaces, controls and entities of the system, and from the Step 2 of PUC is possible to extract the entities that are directly manipulated by the controls. A relation of dependency is used to relate the components, meaning that an component only works correctly if the other component exists and also works correctly.

From Step 2 of MultiGoals (Activity Diagram), illustrated in Figure 7, it is possible to deduce that one interaction space depends on a control when this control provides valid information for that interaction space, in this case "Recipe Browser" depends on "Return Recipes", and "Recipe" depends on "Return Recipes Cataloging Tool" system responsibility. From Step 3 of MultiGoals (Interaction Model) it is possible to extract directly the relation of dependency from the interaction space to system responsibility to entity.

In the "Catalog Recipe" use case the "Return Recipes" and "Return Recipes Cataloging Tool" system responsibilities will depend on the entities "recipe" and "recipe submitted for approval". These dependecies are represented in Figure 8.

Moreover, it is possible to define precedences of implementation based on the existing dependencies. The components on wich more components are dependent on must be developed first since they will be evoqued by the dependent components. It is also possible to isolate the components that are relevant for a specific use case by means of the existing dependency relations, therefore allowing the possibility to choose which use case to develop first.

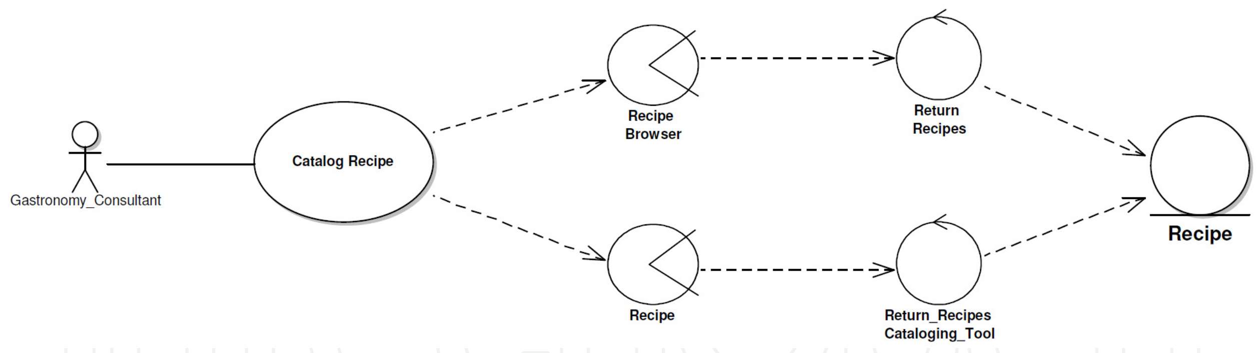

Fig. 8. System Architecture for the "Catalog Recipe" use case.

3.2 MultiGoals basis and related works

Different abstractions provided by different techniques are used to represent the information acquired within MultiGoals. These techniques are: UML (Object Management Group, 2003) that provides the modeling notation, Wisdom (Nuno Nunes, 2001) that provides the main SE process, Usage-centered design (Constantine, 2006) that provides the definition of (essential) use case and the Canonical Abstract Prototypes (Constantine, 2003) for user interface design, and ConcurTaskTrees (Paternò et al., 1997) that provides the technique for user-task modeling.

The following SE related works are also UML based and use case-driven, and support the design of Interactive Information Systems with support for Multimedia:

- UML-based Web Engineering (UWE) (Koch et al., 2008) which is a methodology for the analysis and design of internet application, that produces in five steps the following artifacts: (i) Requirements Specification – (that produces the) Use Cases Model and for each use case an Activity Diagram; (ii) Content – Content Model; (iii) Navigation Structure – Navigation Model and Process Flow Model (for each identified navigation class); (iv) Presentation – Presentation Class and Presentation Model, and; (v) Aspect Modeling – Model Aspect, Link Traversal Aspect and Flow Aspect.
- W2000 (Baresi et al., 2001) is an Hypermedia methodology recognized as the ancestor of a family of several design methodologies, composed of the following steps: "Requirements Analysis", that produces a Use Cases Model, and the "navigation" capabilities associated with each user; the "State Evolution Design" that analyses the state of the information usign an UML StateChart diagram; the "Hyperbase Information Design" that models the domain classes and their attributes; and, the "Hyperbase Navigation Design" that defines the "navigational nodes" and "navigational links" which are derived based a set of rules and design decisions.
- OMMMA (Sauer & Engels, 2001) is a methodology for the development of IMDs that models: the domain (both information and media) using a Class Diagram; interaction using a Collaboration Diagram, temporal and logical system behavior (including navigation) using Statechart and Sequence diagrams, and presentation using the stereotypes of of the identified classes to represent their spatial location in the use interface. OMMMA covers all the design aspects of IMDs (modeling presentation, content and conceptual levels).

For a more comprehensive analysis and comparision of the related works on SE methods for analysis and design with support for Multimedia please refer to (Valente, 2009).

3.3 Interactive use case points for effort estimation

Interactive Use Case Points (iUCP) (Nunes et al., 2010) is a method for effort estimation at early stages of a project based on the weighting of the functional requirements (as use cases) and identified actors, following the phases of requirements and analysis.

iUCP takes advantage of the additional information that can be withdrawn from Usage-centered design (Constantine & Lockwood, 1999) techniques to produce an improved weighting of: actors, specially by means of the concept of user role (an abstraction that represents a relationship between an user and a system that can be described by the context in which is performed); and (essential) use cases by means of the simplification introduced by this concept when compared to the notion provided by the traditional use case as provided by UML (Object Management Group, 2003), and the accurate definition of programmable classes and entities manipulated by each use case that can be retrieved from a system architecture, in order to produce an enhanced calculation of traditional UCPs (Karner, 1993).

Users are weighted in iUCP according to the following criteria in order to calculate the unadjusted actor weight (UAW):

- Simple system actors (a factor of 1) communicate through an API (Application Programming Interface).
- Average system actors (a factor of 2) communicate through a protocol or data store.
- Simple human actors (a factor of 3) are supported by one user role.
- Complex system actors (also a factor of 3) communicate through a complex protocol or data store.
- Average human actors (a factor of 4) are supported by two or three user roles.
- Complex human actors (a factor of 5) are supported by more than three user roles.

Use cases are weighted in iUCP according to the following criteria in order to calculate the unadjusted use case weight (UUCW):

- Simple use cases (a factor of 5) involve a simple UI or simple processing and only one database entity.
- Average use cases (a factor of 10) involve moderately complex UIs and two or three database entities.
- Complex use cases (a factor of 15) involve complex UIs or processing and three or more database entities.

The Unadjusted Use-Case Points (UUCP) is the sum of the previous variables, i.e., UUCP = UAW + UUCW.

The UUCP is then further modified trough the weighting of the technical (Technical Complexity Factor, TCF) factor that reflects how difficult will it be to construct the system, and environmental (Environment Complexity Factor, ECF) factor that estimates how efficient the project is. Both TCF and ECF are the result of similar formulas that include two constants (C_1 and C_2), and a set of 13 and 8 factors (F_1 to F_{13} and F_8), each multiplied by its own weight (W_1 to W_{13} and W_8) respectively and classified (each F_i) from 0 to 5.

The TCF is computed according to the following formula:

$$TCF = C_1 + C_2 \sum_{i=1}^{13} W_i \times F_i$$, where C_1 = 0,6 and C_2 = 0,01.

The complexity factors and each corresponding weight are depicted in Table 3.

F_i	**Factors Contributing to Complexity**	W_i
F_1	Distributed systems.	2
F_2	Application performance objectives, in either response or throughput.	1
F_3	End user efficiency (on-line).	1
F_4	Complex internal processing.	1
F_5	Reusability, the code must be able to reuse in other applications.	1
F_6	Installation ease.	0,5
F_7	Operational ease, usability.	0,5
F_8	Portability.	2
F_9	Changeability.	1
F_{10}	Concurrency.	1
F_{11}	Special security features.	1
F_{12}	Provide direct access for third parties.	1
F_{13}	Special user training facilities	1

Table 3. Technical complexity factores and corresponding weights.

The ECF is computed according to the following formula:

$$ECF = C_1 + C_2 \sum_{i=1}^{8} W_i \times F_i$$, where C_1 = 1,4 and C_2 = -0,03.

The environmental factors and each corresponding weight are depicted in Table 4.

F_i	**Factors Contributing to Efficiency**	W_i
F_1	Familiar with Objectory.	1,5
F_2	Part time workers.	-1
F_3	Analyst capability.	0,5
F_4	Application experience.	0,5
F_5	Object oriented experience.	1
F_6	Motivation.	1
F_7	Difficult programming language.	-1
F_8	Stable requirements.	2

Table 4. Environmental complexity factores and corresponding weights.

The final calculation of UCP follows the formula UCP = UUCP x TCF x ECF

In order to estimate the number of man.hours of a project the UCP should be multiplied by a productivity factor (PF) that reflects the number of man.hours needed to implement one UCP (the lower in cardinality is the PF, the higher is the productivity), i.e., the total development man.hours of a project would be UCP x PF. For further information on the PF and application of the original UCP consult (Clemmons, 2006).

4. Case study

This section illustrates with a case study based on a project ("Creditations") developed at a professional level in the Software Applications Development Office of the University of Madeira (UMa), the application of the methods described in the previous sections.

In a simplified description, the "Creditations" project consisted in providing the UMa's IIS with the automation of the creditations business process, that involved a request from students of creditations for their actual degree based on courses previously approved in degrees of other universities or UMa. For each course the degree's director would then give a creditation in a given quantity of ECTS (European Credit Transfer and Accumulation System) that would contribute for the conclusion of the student's current degree.

The requirements phase for the "Creditations" project consisted in the application of the PUC methodology, and the analysis phase consisted in the application of analysis steps (1, 2, 3 and 11) of the MultiGoals methodology. After the analysis phase, the effort estimation method iUCP was applied. The diagrams and calculations produced by PUC, MultiGoals, and iUCP will now be presented and described.

4.1 Requirements definition: PUC Step 1 – Interiorize project

The elaboration of the High-Level Concept (HLC) was based on the project description provided by the client, the Rectory of UMa, and the available documentation on the creditations business process. The HLC for the project was:

"Provide students the possibility of requesting creditations for the current degree based on past courses, and the degree's director with the possibility of crediting the request. The student will be able to appeal, in which case the process will be reviewed by the rectory."

4.2 Requirements definition: PUC Step 2 – Information identification

In order to accomplish this step the initial main concepts of information, transformed into entities, derived from the HLC (high-lighted in light-green) were: Student; Creditation; Course; Degree; and Director. The remaining entities (in dark-green): Registration; Degree Plan; Conclusion Plan; Institution; and Files, were ellicited later during the analysis process and then added to the diagram and associated with the previous entities. The final Domain Model is depicted in Figure 9.

4.3 Requirements definition: PUC Step 4 – Process use cases model

The third step of PUC, Step 3 - Business Processes Identification, identifies BPs that relate to each entity if within the scope of the project defined in the HLC. Although many BPs existed that relate to each identified entity such as: registration process; degree publication; director election, among many others, only the creditation BP was within the scope of the HLC, and therefore, was the only one documented, yet, not in Step 3 for schedule reasons, but in Step 4, what would be sufficient to attain the primary goal of the requirements phase, the identification of the use cases for the project.

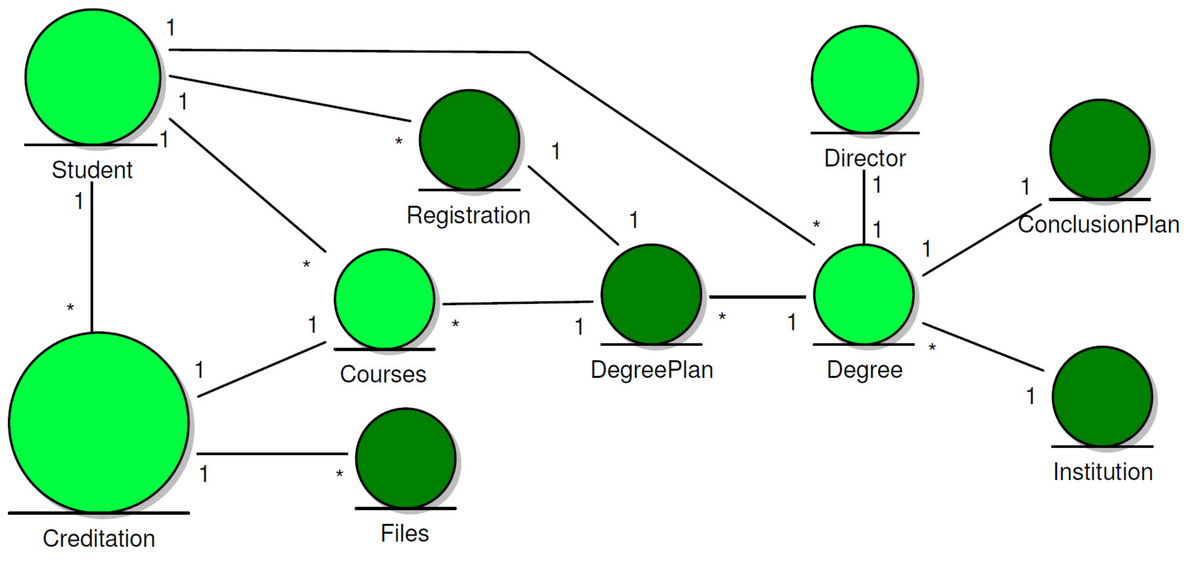

Fig. 9. PUC Step 2 - Domain Model for the "Creditations" project.

The elaboration of Step 4 – Use Cases Identification was made with the client of the project, the Rectory, that provided all the documentation and information necessary to elaborate the sequence of steps that would lead to finishing the creditation BP, and with the Academic Issues Unit (UAA), that accumulated the experience of hundreds of processes.

The previous creditations BP, based in excel files, was first analyzed, and then improved, introducing new activities, for inspection of the initial requests from the students, and appeal to the rectory. The BP was then modeled and decided which activities would be automated (transformed into use cases). The final result of the revized and translated (from portuguese to english) BP is illustrated in Figure 10.

4.4 Analysis: MultiGoals Step 2 – Activity Diagram

Following the identification of the use cases in Step 4 of PUC, the next step would be gathering all the use cases in a single diagram (the Use Cases Model) and relate them to the identified actors. However, since the Process Use Cases Model already establishes those relations, and the "Creditations" BP is the only BP for the project, there is no need to repeat the same information in another diagram, therefore Step 1 of MultiGoals was omitted.

Following the identification of the use cases for the project, the next step consists in analyzing the tasks that each actor has to carry out in the system in order to accomplish its activity. Each use case was decomposed into user tasks and system responsibilities in a sequence of steps, and each task was related to the interaction space in which it would be performed, and each system responsibility to the entities that would provide the information necessary to its functioning.

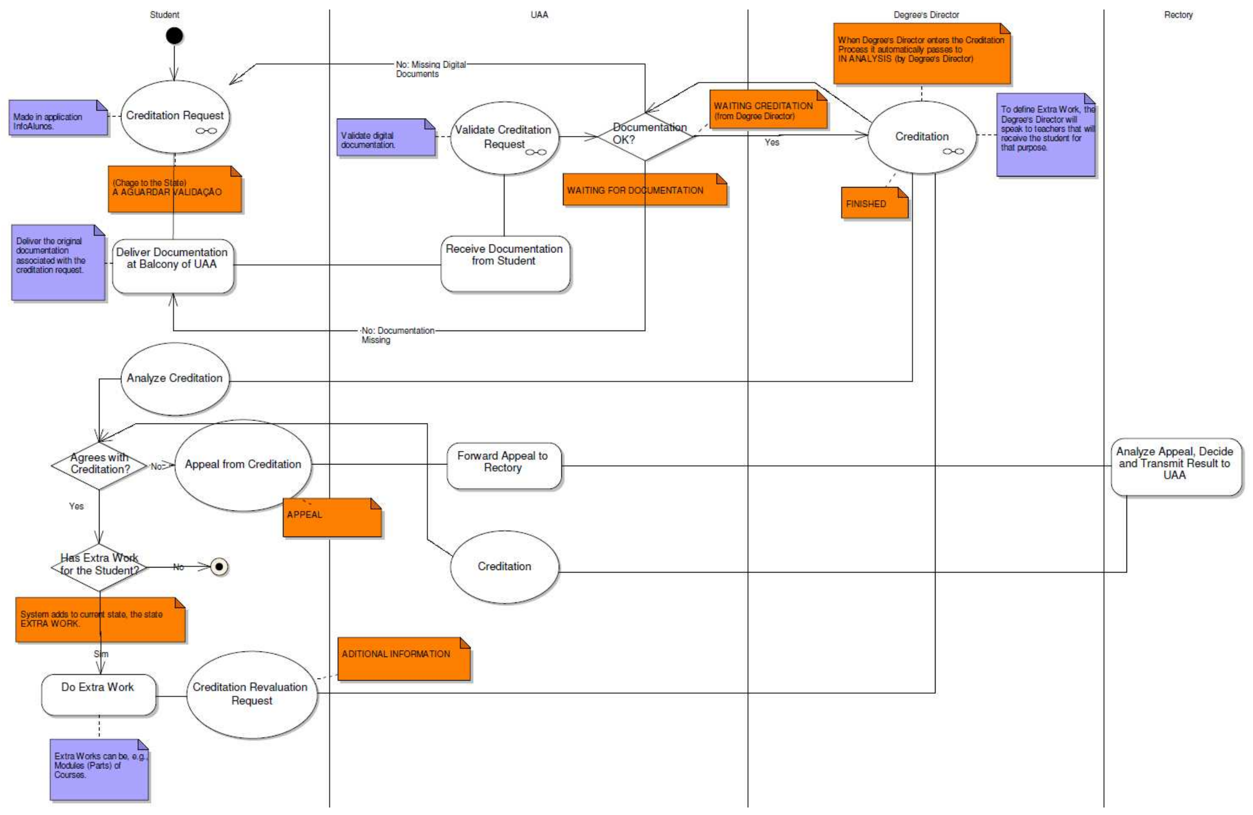
Student
UAA
Degree's Director
Rectory
Made in application InfoAlunos.
Creditation Request
No: Missing Digital Documents
Validate digital documentation.
Validate Creditation Request
Documentation OK?
WAITING CREDITATION (from Degree Director)
When Degree's Director enters the Creditation Process it automatically passes to IN ANALYSIS (by Degree's Director)
Yes
Creditation
To define Extra Work, the Degree's Director will speak to teachers that will receive the student for that purpose.
(Chage to the State) A AGUARDAR VALIDAÇÃO
WAITING FOR DOCUMENTATION
FINISHED
Deliver the original documentation associated with the creditation request.
Deliver Documentation at Balcony of UAA
Receive Documentation from Student
No: Documentation Missing
Analyze Creditation
Agrees with Creditation?
No
Appeal from Creditation
Forward Appeal to Rectory
Analyze Appeal, Decide and Transmit Result to UAA
Yes
APPEAL
Has Extra Work for the Student?
No
Creditation
System adds to current state, the state EXTRA WORK.
Sim
ADITIONAL INFORMATION
Do Extra Work
Creditation Revaluation Request
Extra Works can be, e.g., Modules (Parts) of Courses.

Fig. 10. PUC Step 4 – Process Use Cases Model for the "Creditations" project.

From the six use cases identified, we chose to present one of the most representative, "Creditation". In this diagram all the tasks that the user needs to carry out in order to accomplish the use case were identified and the appropriate responses from the system were defined. Complementarily, the interaction spaces where the user interactions occur, and the entities on which the system responsibilities depended were identified. The Activity Diagram for the "Creditation" use case is depicted in Figure 11.

4.5 Analysis: MultiGoals Step 3 – Interaction Model

The Interaction Model is used to specify interaction between the user and the system that was not already specified in the Activity Diagram. Following the example given in Step 2, Figure 11, the "Specify Scientific Area and Level", "Specify Classification" and "Specify State of Creditation Request" tasks, specific to a single course (a creditation request can gather several past courses), were associated with the system responsibilities that supported the information needed to the task, as well as the association between the tasks and the interaction spaces where they occur and the entities on which the system responsibilities depended.

The Interaction Model for the tasks derived from the "Creditation" use case are depicted in Figure 12.

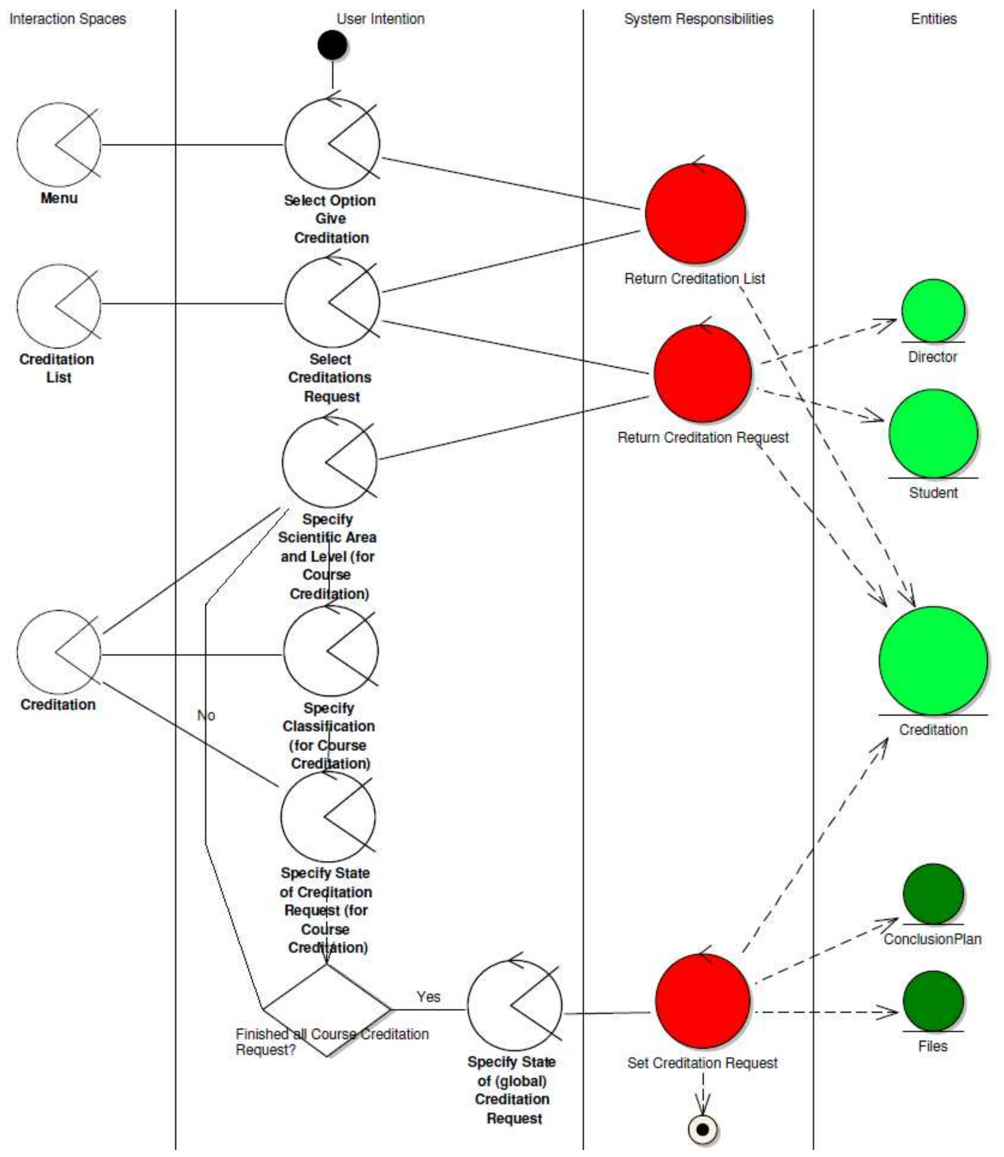
Interaction Spaces
User Intention
System Responsibilities
Entities
Menu
Select Option Give Creditation
Return Creditation List
Creditation List
Select Creditations Request
Return Creditation Request
Director
Student
Specify Scientific Area and Level (for Course Creditation)
Creditation
Specify Classification (for Course Creditation)
Creditation
No
Specify State of Creditation Request (for Course Creditation)
ConclusionPlan
Yes
Finished all Course Creditation Request?
Specify State of (global) Creditation Request
Set Creditation Request
Files

Fig. 11. MultiGoals Step 2 – Activity Diagram for the "Creditation" use case.

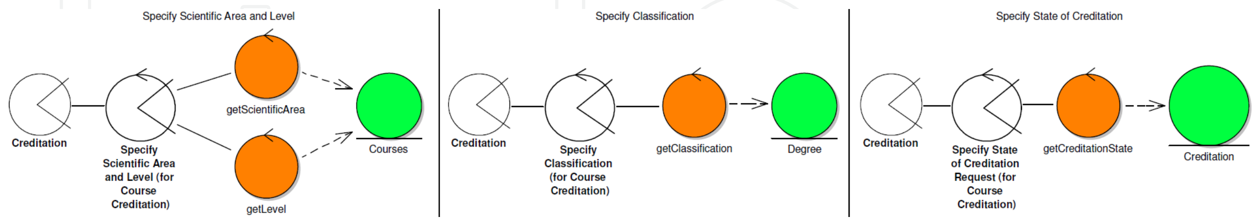
Specify Scientific Area and Level
Creditation
Specify Scientific Area and Level (for Course Creditation)
getScientificArea
getLevel
Courses
Specify Classification
Creditation
Specify Classification (for Course Creditation)
getClassification
Degree
Specify State of Creditation
Creditation
Specify State of Creditation Request (for Course Creditation)
getCreditationState
Creditation

Fig. 12. MultiGoals Step 3 – Interaction Model.

4.6 Analysis: MultiGoals Step 11 – System Architecture

The elaboration of System Architecture is based on the definition of the existing dependencies between the identified components. The composition of the diagram follows the sequence: placing of the interaction spaces, placing of the system responsibilities on which the interaction spaces depend; placing of the entities on which the system responsibilities depend on. The System Architecture is presented in Figure 13.

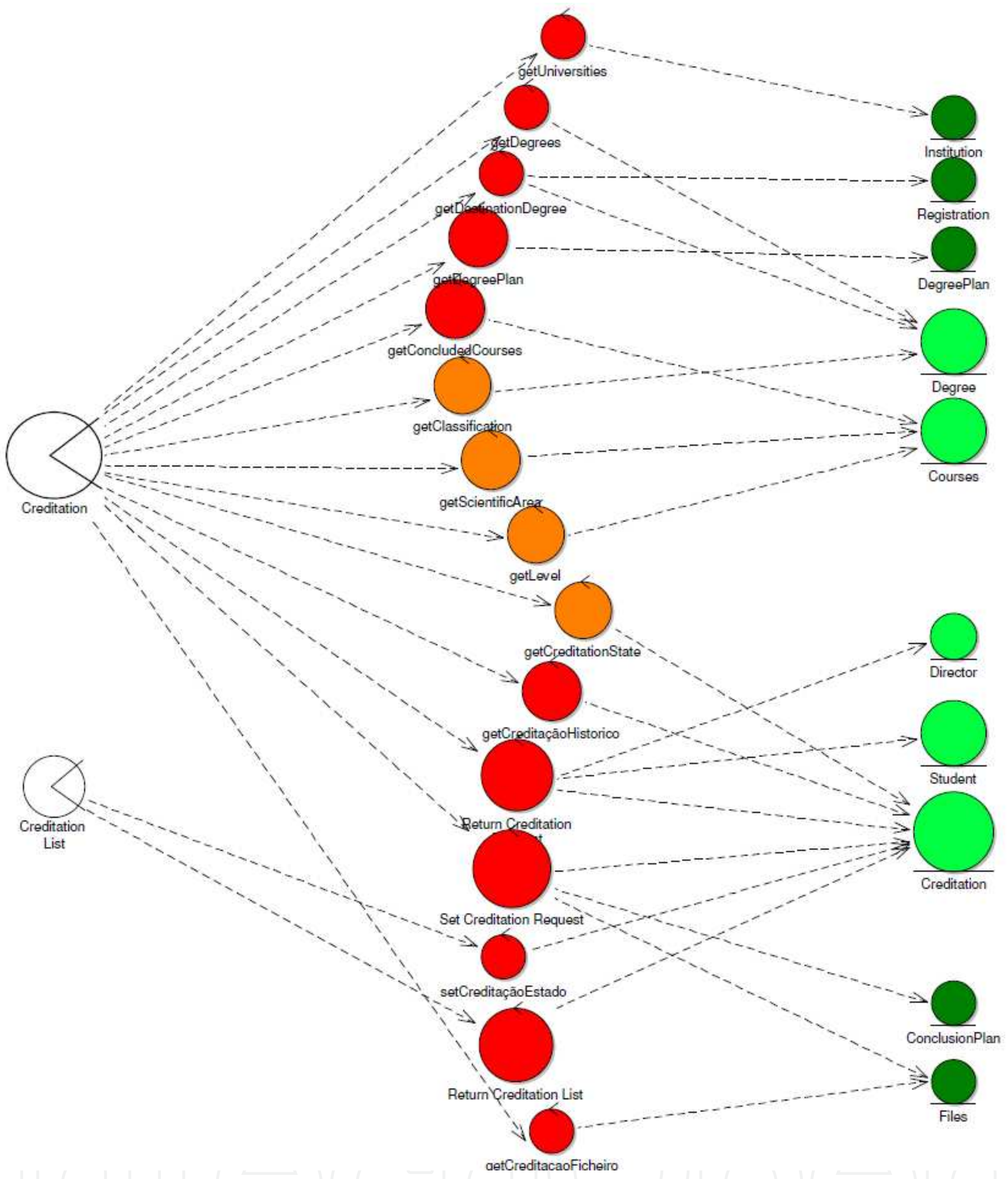

Fig. 13. MultiGoals Step 11 – System Architecture Creditations project.

4.7 Effort estimation: Interactive use case points

The calculation of the effort estimation for the "Creditations" project was based on the information included in the diagrams presented in this section.

The first step for the application of the iUCP method was the definition of the Unadjusted Actor Weight (UAW). Since Goals does not distinguish actor from role, i.e., every actor is actually a role, and since there was no possible generalization regarding a common actor considering the identified roles: Student; UAA; and Degree's Director (the Rectory role did not interact with the system), every actor was considered a simple human actor. Thus, 3 simple human actors multiplied by a factor of 3 resulted in a UAW of 9.

The next step was the calculation of the Unadjusted Use Case Weight (UUCW). In this case, since all the use cases interacted with 10 database entities all the use cases were considered complex once all of them interacted with the "Creditation" and "Creditation List" interaction spaces. Thus 6 use cases multiplied by a factor of 15 resulted in and UUCW of 90.

The calculation of the technical complexity factor (TCF) consisted in assigning a classification between 0 and 5 to the technical factors (F_1 trough F_{13}) and multiply each factor for each weight (W_1 trough W_{13}) and sum each result, situation that is depicted in Table 5.

Factor (F_i)	**F_1**	**F_2**	**F_3**	**F_4**	**F_5**	**F_6**	**F_7**	**F_8**	**F_9**	**F_{10}**	**F_{11}**	**F_{12}**	**F_{13}**	**Sum**
Weight (W_i)	2	1	1	1	1	0,5	0,5	2	1	1	1	1	1	
Classification	0	4	3	5	5	3	4	0	4	3	3	0	3	
Classification*W_i	0	4	3	5	5	1,5	2	0	4	3	3	0	3	33,5

Table 5. Base parameters for the calculation of the TCF.

The result was then multiplied by the constant C_2 = 0,01 and summed to the constant C_1 = 0,6 resulting in a TCF of 0,935.

The calculation of the environment complexity factor (ECF) consisted in assigning a classification between 0 and 5 to the technical factors (F_1 trough F_8) and multiply each factor for each weight (W_1 trough W_8) and sum each result, situation that is depicted in Table 6.

Factor (F_i)	**F_1**	**F_2**	**F_3**	**F_4**	**F_5**	**F_6**	**F_7**	**F_8**	**Sum**
Weight (W_i)	1,5	-1	0,5	0,5	1	1	-1	2	
Classification	3	0	4	3	4	4	0	3	
Classification*W_i	4,5	0	2	1,5	4	4	0	6	22

Table 6. Base parameters for the calculation of the TCF.

The result was then multiplied by the constant C_2 = -0,03 and summed to the constant C_1 = 1,4 resulting in a TCF of 0,74.

For the calculation of the UCP for the project the formula UCP = UUCP x TCF x ECF (in which UUCP = UAW + UUCW) was applied with the result of 68,4981. Applying a productivity factor (PF) of 21,57 (a PF moderated by the previous application of the iUCP method), a final estimation of 1476,82 man.hours was obtained.

5. Conclusions

Goals defines a set of restrictions and main guidelines regarding the requirements, analysis and design phases that are crucial for the correct conception of an IIS, resulting in artifacts that can be used in the remaining phases of the construction to minimize development errors. The application of the phases of requirements and analysis is sufficient to produce valuable results, such as a system conceptual architecture and an effort estimation, that can be crucial for the success of the complete project.

The requirements phase, completed by means of the application of the PUC methodology, resulted in the correct identification of the functional requirements, entities and actors for the project based on the reorganization of the base BP for the project, in which the Process Use Cases Model played a central role in the discussion between the software development team, the client and other stakeholders of the project.

The analysis phase, completed by means of the application of the MultiGoals methodology, complemented the results of the previous phase with the identification of the user interfaces, the system functions and entities for the project, therefore producing sufficient information so that the system could be design in detail in the following design phase. Moreover, the system architecture provides an overview of the dependencies between the objects of the project allowing development tasks assignment.

The application of the iUCP method for effort estimation, integrated by means of the compatibility of the definition of actor (MultiGoals) and role (iUCP) and by the identification of the entities manipulated by each use case, proved to be a consistent tool, since the deviation from the 1476,82 man.hours was only of (plus) 9%.

In spite of the need to accomplish 7 steps in order finish the complete analysis of the problem behind the project and estimate its effort in terms of man.hours, they provide the analyst with enough information to easily bridge its efforts towards the correct design of the solution (only steps 4, 5 and 6 of MultiGoals for non-Multimedia projects), therefore increasing implementation efficiency and minimizing future need for maintenance of the completed IIS.

6. References

Baresi, L., Garzotto, F. & Paolini, P. (2001). Extending UML for Modeling Web Applications. In: Proceedings of 34th Hawaii International Conference on System Sciences, Maui, Hawaii. Vol. 3.

Clemmons, R. (2006). Project estimation with Use Case Points, In: CrossTalk - The journal of Defence Software Engineering, Diversified Technical Services Inc., Vol. 19, Issue 2, pp. 18-22.

Constantine, L. (2002). Usage-Centered Engineering for Web Applications, In: IEEE Software, Vol. 19(2), pp. 42-50.

Constantine, L. (2006) Human Activity Modeling: Toward a Pragmatic Integration of Activity Theory and Usage-Centered Design. In: Human-Centered Software Engineering II, Seffah, A., Vanderdonckt, J., and Desmarais, M. (eds.), NY: Springer-Verlag, 2009.

Constantine, L. & Lockwood, L. (1999). Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design, Addison-Wesley Longman. ISBN: 978-0201924787.

Constantine, L. & Lockwood, L. (2000). Structure and Style in Use Cases for User Interface Design. In Object Modeling and User Interface Design. Boston: Addison Wesley. ISBN: 978-0201657890.

Dijkman, R. & Joosten, S. (2002). An Algorithm to Derive Use Cases from Business Processes. In: Proceedings of the 6th IASTED International Conference on Software Engineering and Applications (SEA), M.H. Hamza (ed.), Boston, MA, USA, pp. 679-684.

Eriksson, H. & Pencker, M. (2001). Business Modeling With UML: Business Patterns at Work (1st ed.), John Wiley & Sons, ISBN: 0471295515.

González, J. & Díaz, J. (2007). Business process-driven requirements engineering: a goal-based approach. In: Proceedings of 8th Workshop on Business Process Modeling, Development, and Support (BPMDS'07), Trondheim, Norway.

Karner, G. (1993). Resource Estimation for Objectory Projects. Objective Systems SFAB.

Koch, N., Knapp, A., Zhang, G. & Baumeister, H. (2008) UML-based Web Engineering: An Approach based on Standards. In: Web Engineering: Modelling and Implementing Web Applications. pp. 157-191, chpt. 7. Springer, HCI, Vol 12, 2008. ISBN: 978-1-84628-922-4.

Kreitzberg, C. (1999). The LUCID Framework (Logical User Centered Interaction Design) (Pre-Release Version 0.4). Retrieved December 19th 2007: http://www.cognetics.com

Land, R. (2002). A Brief Survey of Software Architecture, In: Mälardalen Real-Time Research Centre (MRTC) Technical Report, ISSN 1404-3041.

Nunes, N. (2001). Object Modeling for User-Centered Development and User Interface Design: The Wisdom Approach, Phd Thesis, Universidade da Madeira, Madeira, Portugal.

Nunes, N., Constantine, L. & Kazman, R. (2010). iUCP - Estimating interactive software project size with enhanced use-case points, In: IEEE Software, Vol. 28, pp. 64–73, ISSN: 0740-7459.

Object Management Group. (2003). Unified modeling language superstructure, version 2.0. final adopted specification.

Paternò, F., Mancini, C. & Meniconi, S. (1997). ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, In: Proceedings of INTERACT '97, IFIP TC13 International Conference on HCI, pp. 362-369.

Sauer, S. & Engels, G. (2001). UML-based Behavior Specification of Interactive Multimedia Applications. In: Proceedings of IEEE Int'l Symposium on Human-Centric Computing Languages and Environments (HCC), pp. 248—255, Stresa, Italy.

Shishkov, B. & Dietz, J. (2005). Deriving Use Cases from Business Processes. In: Enterprise Information Systems V, S. Netherlands (Ed.), Vol. Computer Science, pp. 249-257. ISBN: 978-1-4020-1726-1 (Print) 978-1-4020-2673-7 (Online).

Štolfa, S. & Vondrák, I. (2006). Mapping from Business Processes to Requirements Specification. Universitat Trier.

Valente, P. (2009). Goals Software Construction Process, VDM Verlag Dr. Müller, ISBN: 978-3639212426.

Valente, P. & Sampaio, P. (2007a). Process Use Cases: Use Cases Identification, In: Proceedings of ICEIS´2007 - 9th International Conference on Enterprise Information Systems, Vol. Information Systems Analysis and Specification, pp. 301-307, Funchal, Madeira, Portugal.

Valente, P. & Sampaio, P. (2007b). Goals: Interactive Multimedia Documents Modeling. In: Lecture Notes in Computer Science, K. Luyten (Ed.), Vol. 4385/2007, pp. 169-185, Springer Berlin/Heidelberg, ISBN: 978-3-540-70815-5, Hasselt, Belgium.

Webinterx. (2006). Services. Retrieved December 19th 2007: http://www.webinterx.com/services/