CLEI ELECTRONIC JOURNAL, VOLUME 17, NUMBER 2, PAPER 1, AUGUST 2014

Unified Process for Domain Analysis integrating Quality, Aspects and Goals

Francisca Losavio
Laboratorio MoST, Centro ISYS, Escuela de Computación,
Facultad de Ciencias, Universidad Central de Venezuela
Caracas, Venezuela
francisca.losavio@ciens.ucv.ve

Alfredo Matteo
Laboratorio MoST, Centro ISYS, Escuela de Computación,
Facultad de Ciencias, Universidad Central de Venezuela
Caracas, Venezuela
alfredo.matteo@ciens.ucv.ve

and

Irma Pacilli Camejo
Departamento de Informática
UPT del Norte de Monagas "Ludovico Silva"
Caripito-Monagas, Venezuela
pacilli_irma@hotmail.com

Abstract- The Requirements Engineering (RE) discipline is where the software system needs or requirements are captured; these are then "translated" into software components. At present, functional requirements are treated, but non-functional requirements (NFR) are neglected, causing problems at later stages of development. In an industrial software production context, product quality must be considered, and the Domain Analysis discipline within RE, proposes different approaches to treat NFR for building a Reference Architecture (RA) from which all products of a domain family can be generated. Consequently, the same process is adapted to different contexts and abstraction levels. This paper proposes a *Unified Process for Domain Analysis (UPDA)*, based on Aspect and Goal orientations to deal with NFR, specified by quality standards to enhance communication. UPDA integrates techniques that are separately used: - the Chung and others extended process of Losavio and others, based on the NFR Framework with treatment of crosscutting concerns, and – the ISO/IEC 25010 quality standard to specify NFR. Three sub-processes constitute UPDA: - *Construction of the quality model*, - *Identification of crosscutting concerns and* - *RA design*. The main artifact obtained is the RA, which can be reused as an asset in the context of software product lines.

Keywords - domain analysis, Chung extended process, NFR Framework, aspect-orientation, quality standards, ISO/IEC 25010, reference architecture, UPDA

I. Introduction

The process of software development has shown great strides since the inception of Software Engineering to the present. Modern approaches have been defined to improve in general this process; each phase of the development is becoming more specialized and the application of a variety of techniques and tools have been introduced to help improving the quality of the final product, understood as the set of desired properties or characteristics that must be present in the product, from the user point of view and the product itself [1]. In the same way that the phases of the development process have been specialized, approaches have also been presented to raise the abstraction level: applications consider now including needs and goals of organizations and even their contextual environment.

A crucial discipline within the software development process is *Requirements Engineering* (RE). Different definitions are found in the literature, such as the process of discovering the purpose of software systems by identifying stakeholders and their needs, while documenting these needs to favor analysis and communication. These needs, called requirements, will be "translated" later on into software components [2]. On the other hand [3] defines RE as a process including elicitation, evaluation, specification, consolidation and evolution of the goals, functionalities, qualities and constraints that a software system must accomplish in an organizational or physic context. It can then be generalized that RE allows determining the services that a system must satisfy, as well as the constraints that must be considered. In this context, we use the term *domain* and the definition given in [4], as the minimum set of properties that describe accurately a family of problems for which computer applications are required. The term *application domain* is defined by [5] as a specific business area where certain categories of software systems are used, helping to generalize functional (FR) and non-functional requirements (NFR) for these systems. The *Domain Engineering* (DE) discipline, within RE, analyzes the domain knowledge, to produce reusable assets for similar systems or family of products. The first phase of DE is *Domain Analysis* (DA), where the elicited requirements are specified by common and variable characteristics for a family of products, modeled in general by a generic framework called reference architecture, with focus on the reuse of domain assets. DA is widely used in the development of complex systems in the context of software product lines [5]. From these points of view, it is important to ensure that the final product satisfies all the stated FR and NFR; in order to achieve this, the quality of the software product and intermediate artifacts obtained during the development process, must be assured; FR are directly perceived by the user; however NFR are inherent software properties [6] which can be captured from the description of the domain or required by some functionality as an implicit functionality; they are not directly perceived by the user and they are directly associated with Quality Requirements (QR). However, in the literature the terms NFR, QR, and quality attributes are frequently used indistinctly. In this work we differentiate NFR, QR and quality attributes, according to the quality model specification of the standard ISO/IEC 25010 [1]; a *quality model* is a hierarchical structure of quality characteristics used to decompose QR into measurable elements called *quality attributes*. It should be noted that the quality model describing a software product is considered a bridge between FR, NFR and their respective QR; it must be customized or adapted to specify the quality of a particular domain family and it represents a set of *scenarios* in the sense of classical methods of architectural design. In this way a clear traceability is established between the requirement (FR or NFR) and its required degree of quality. This traceability, using different techniques and approaches, briefly described in what follows, helps to define precisely all the components of the architecture, which is the main structure on which the whole software system is built.

The *architecture* of a software system is defined in [7] as a set of components and connectors, with precise behavior. Documented architecture [8] [9] must meet both FR and NFR requirements of the system. Architecture design considers that every architectural decision (for example, add and/or remove components and/or connectors) is justified on the basis of having verified that the quality requirements are met. A *Reference Architecture* (RA) is a generic architecture from which all products (systems), family members of the domain, can be generated [10] from a reusable common core of assets. The term RA is widely used in the context of software product lines and we use this term because in our context, RA is derived from the domain knowledge, taking into account the basic functionality and overall NFR of the domain, for which RA is generally responsible for. Our proposition is based on establishing RA from DA process using one of the approaches of *Goal-Oriented Requirements Engineering* (GORE) [11]. The GORE approach involves the intentional point of view, that is to say, the interest of all participants or stakeholders of the software project to obtain, through a refinement step, the system's requirements. A *goal* is the interest that the system, artifact, process or activity, must achieve [11]. This approach proposes a more natural language to analyze requirements, decomposing them into a logical structure expressing the whole system's needs. Within the GORE context, the *NFR framework* [12] introduces the *operationalization* concept, which states the way to implement a NFR as an architectural component; in this context a NFR is called "softgoal" and it is in general a goal difficult to express and not directly perceived by the user [8]; a FR instead is called "hardgoal". A graph structure, called *Softgoals Interdependency Graph* (SIG) is used to decompose and relate softgoals [12]. GORE proposes other approaches for requirements analysis, such as KAOS (Knowledge Acquisition in Automated Specification of Software) in [14] and the i* Framework in [15], however we will discuss here only the approaches that have been used in our research. We consider in this paper explicitly the *NFR Framework* of the GORE approach, as it is used by for analysis and design by Supakkul and Chung [16]. Aspect orientation is considered by the *integration of crosscutting concerns* to the *use case model* of Moreira, Brito and Araujo [17], following the approach called "early aspects" of the *Aspect-Oriented Requirements Engineering* (AORE) [18]; they state that crosscutting concerns, which are properties of interest that are used by or crosscut other properties, are taken into account at early stages of the software development process in order to avoid problems of entangled or scattered code later on. Losavio, Matteo and Pacilli [19] have recently incorporated this approach to the *Chung Extended Process* (CEP), also defined by Losavio, Matteo and Pacilli in [8]. Finally, standards for product quality specification are included into the

Domain Analysis Process with Quality Standards (DAP-QS), with the proposal of Losavio, Matteo and Rahamut [20], where the ISO/IEC 25010 standard quality model is used to specify the domain QR.

The main goal of this paper is to present a *Unified Process for Domain Analysis* (UPDA) to obtain a reference architecture for a family of systems or products in a domain, integrating some of the approaches that have proven successful in their individual proposals: aspect and goal orientations, the NFR Framework and the NFR specification by standard quality models. The integration of these approaches can be appreciated in Figure 1.

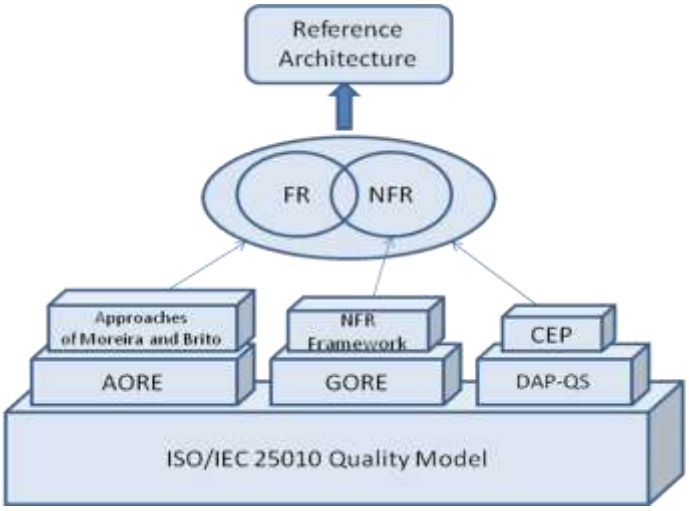

Figure 1: Approaches integrated in the UPDA proposition

A. UPDA Overview

The proposed UPDA process is modeled with SPEM 2.0 (Software & Systems Process Engineering Metamodel) of the OMG, which is an industrial standard for the representation of process models in software engineering and systems engineering [21]; UPDA offers three main disciplines for generating the RA as the final artifact (see Fig. 2).

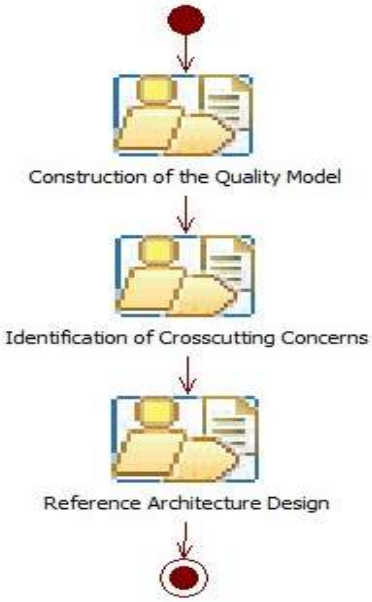

Figure 2: UPDA

In the first discipline, *construction of the quality model*, the product quality model [1] is built through the implementation of six activities and the DAP-QS process [20], to obtain the *functional* and *non-functional "cores"* of the domain as intermediate artifacts. The functional core contains the basic functionality that is common to the domain family of products and its associated QR, called in this case, *implicit functionalities*; the non-functional core contains the NFR expressed informally as text taken from the domain description, and their correspondence with the QR specification. Based on this knowledge of the domain, the quality model is built.

In the second discipline, *identification of crosscutting concerns*, the existing crosscutting concerns are identified, according to AORE [18]. In this discipline the approaches of Moreira, Brito and Araujo [17], and Supakkul and Chung [16] are used to obtain an extended use case model with FR and NFR.

The third and final discipline, *reference architecture design*, determines the global NFR, of which the architecture is generally responsible for; in the GORE context, these global requirements are decomposed using the NFR Framework for their operationalization and ultimately the RA is built from these components.

The structure of this article, besides this introduction and the conclusion, is as follows: a second section which describes broadly techniques and tools used in UPDA: the quality model of the domain as used within the DAP-QS process, the approach of Moreira, Araujo and Brito [17] and the Chung Extended Process [19] based on the NFR Framework for the goal-oriented analysis and design [16]. In the third section, UPDA is described in details. In the fourth section, the validation of the proposal is illustrated with a case study in the e-banking domain. Finally, in the fifth section, a comparative analysis with similar processes and techniques is presented.

II.Techniques and Tools used by the Unified Process for Domain Analysis (UPDA)

A. Quality Model of the domain with the ISO/IEC 25010 standard

The quality of a system is the degree to which the system meets the specified and implied needs of stakeholders. In this sense, the establishment of quality characteristics that the software product must satisfy takes on significant importance [1].

To specify the quality of the product in this proposal, the ISO/IEC 25010 standard [1] will be used, since it is a recognized standard, accepted and much used in the software development process by the software engineering community. It should be noted that other quality models could be used (standard or not), besides the ISO/IEC 25010; however all of them need to be adapted or customized to the particular domain. However, dealing with the adaptation standards is not straightforward because they do not provide guidelines for the customization, which make their use difficult and based on the expert knowledge on the standard and on the domain. We use the ISO/IEC 25010 because, on one hand it is well known within the software community and it has been recently updated, and on the other hand, we have experience is using and customizing it.

The ISO/IEC 25010 standard [1] consists of eight (8) main quality characteristics, which are hierarchically represented, from features of high abstraction level, which are refined into sub-characteristics of lower level, until the so-called *attributes*, which are measurable properties. Table 1 shows the hierarchy, until the sub-characteristics level. These qualities after the respective analysis, translate into architectural components, derived from global system or organizational constraints, which are not directly perceived by the user, they represent "implicit functionalities" derived from the functionality required by the end user. We will speak indistinctly of quality characteristics or QR.

Table 1: Characteristics and Sub-characteristics of the ISO/IEC 25010 quality model

Functional Suitability	Performance efficiency	Compatibility	Usability	Reliability	Security	Maintainability	Portability
Functional completeness	Time-behavior	Co-existence	Appropriateness recognisability	Maturity	Confidentiality	Modularity	Adaptability
Functional correctness	Resource utilization	Interoperability	Learnability	Availability	Integrity	Reusability	Installability
Functional appropriateness	Capacity		Operability	Fault tolerance	Non-repudiation	Analyzability	Replaceability
			User error protection	Recoverability	Accountability	Modifiability	
			User interface aesthetics		Authenticity	Testability	
			Accessibility				

The establishment of the quality model for the domain family of products helps to define the overall NFR, valid for the entire domain, as well as some quality requirements for common functionalities and represent a *quality view of the domain knowledge* [22]. In this case, the *domain quality model* is expressed using the process defined by Losavio and others [20], DAP-QS, where it is developed for a family of products. The steps of this process are presented in what follows:

DAP-QS Process

Input: the textual description of the problem, for which a computational solution (system or application) is required; identify the high-level architectural solution (s) or style (s) for the family.

a) Define a taxonomy of the main functionalities for the family: the functionality list

b) Define the domain quality model, using the ISO/IEC 25010 standard.

b.1) The *domain architectural quality* is specified using the quality properties of the architectural solution or a style for the domain family.

b.2) The *domain functional quality* is specified, where each functionality of step "a" is associated with its quality requirements or quality characteristics specified by ISO/IEC 25010, as goals to be met to achieve the respective functionality.

Output: quality model of the domain family.

Notice that QR derived from business and organizational rules, such as "portability" to other platforms, or the system "interoperability" are global QR that should be considered, and they were included as architectural quality in the DAP-QS process; in the UPDA complete process (see Section III) all these QR are included in the non-functional core.

B. Extended use case model

In the field of RE, several research trends are found that seek the integration and modeling of FR and NFR at design level. Note that in all these works, the term "quality attribute" is equivalent to the term of ISO/IEC 25010 "quality characteristic" or QR already discussed; recall that *quality attribute* for this standard means a low level measurable quality characteristic. Moreira, Araujo and Brito [17] consider quality attributes (QR) of crosscutting FR in the use case model, and they propose a process that consists of three main activities: a) *identification*, where system requirements are identified and these quality attributes relevant for the application and stakeholders are selected, b) *specification*, first FR are specified using an approach based on use cases and then, quality attributes are described using special templates and the crosscutting FR are identified, and c) *requirements integration*, where a set of models are proposed to represent the mainstreaming of quality attributes and FR. According to the SQuaRE standard [6], a quality attribute in [17] corresponds to a QR. The first two activities use tables to display relevant information, but only in the third activity, quality is considered.

Quality attributes are associated with FR, for the entire system, through an "extended use case model"; they are represented in a UML (Unified Modeling Language) [13] extended diagram to include new use cases stereotyped for each quality attribute, in such a way that the initial use cases include new crosscutting attributes; they are represented in the diagrams with the use case oval and their names are indicated with the quality attribute symbol stereotype of UML this can be appreciated in Fig. 3, where such <<include>> use cases are stereotyped as <<Security>> and <<Response Time>> representing the NFR within the extended use case model, proposed, in a case study of an automatic toll payment system for vehicles registered through an electronic device installed in each vehicle [17].

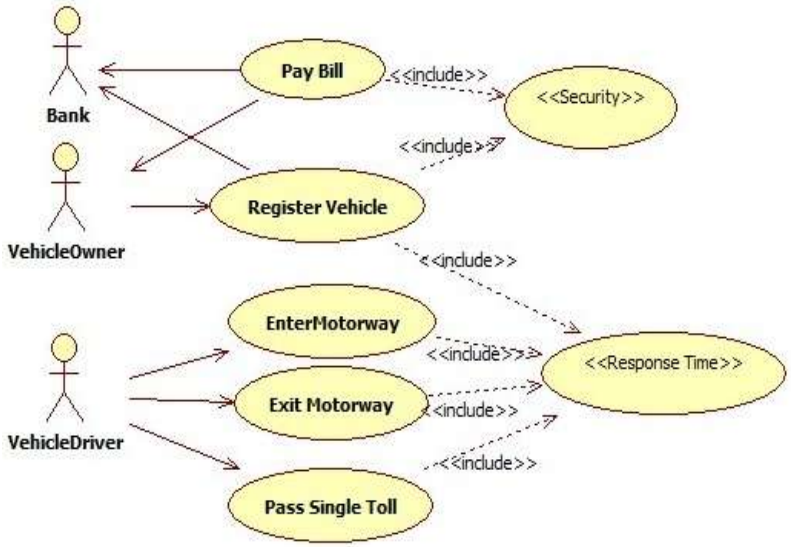

Figure 3: Integrated diagram of FR and NFR [17]

C. Chung Extended Process (CEP)

The Chung Extended Process (CEP) presented in [19] proposes the extension of the original Chung process [16] integrating the DAP-QS process described above, plus the analysis of crosscutting concerns, while identifying global NFR as in [17].

Recall that the basic premises set forth in this process are based on the NFR Framework, initially defined in [23] and the classic scenario-based or use case approach. These premises provide:

- NFR integrated to the use case model, that correspond to the main users FR through the so-called association points, used to classify NFR
- Establish the extent of propagation rules for partnership proposals related to the use case model.

Both association points as propagation rules allow to pinpoint global NFR and the inclusion of specific NFR as included use cases, stereotyped as <<include>> in UML [13] within the use case model [17], as it has been previously explained.

Then, potential crosscutting concerns are identified from composition tables [17], in the context of an early-aspects design. The activity diagram in Fig. 4 illustrates the incremental and iterative process with the integrated FR and NFR; Figures 5 and 6 illustrate this process with an example taken from [19] for a pricing system for airlines, where providers are requesting a service sending their proposals, which may or may not be approved by the manager.

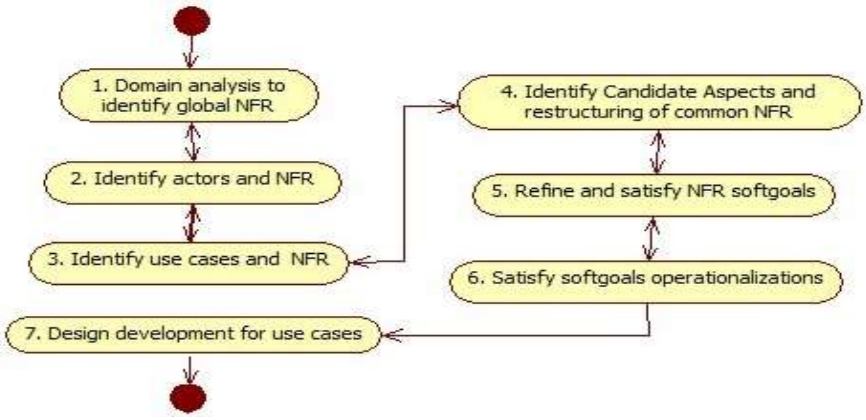

Figure 4: CEP Activity Diagram integrating FR and NFR with the DAP-QS process and identification of candidate aspects [19]

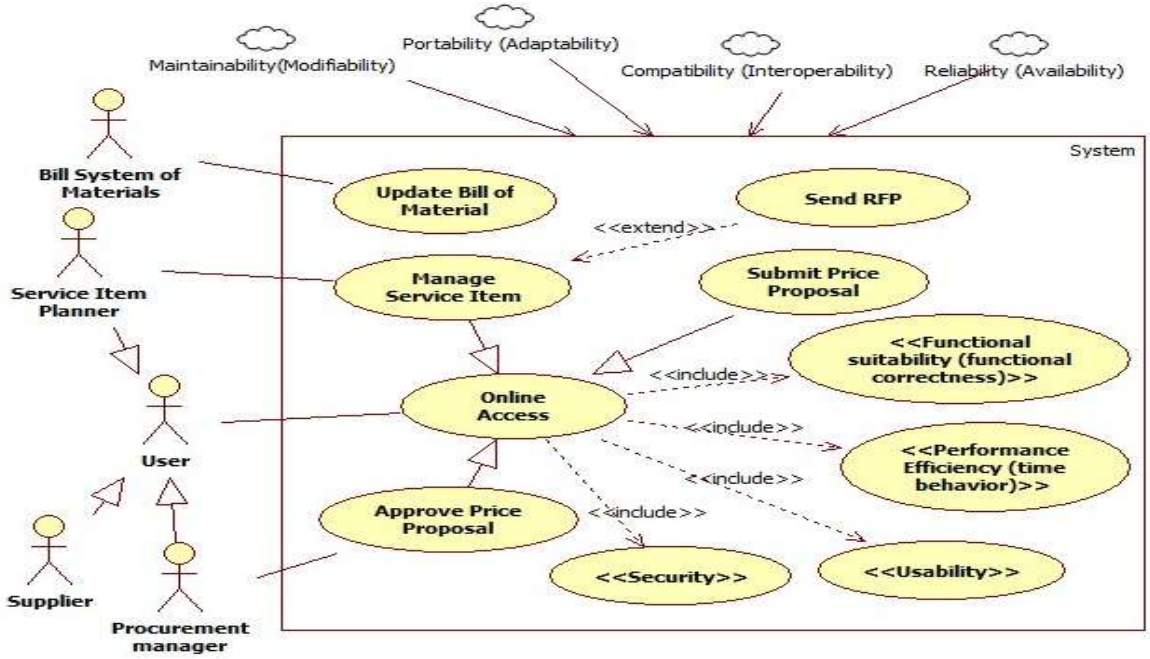

Figure 5: Restructuring of common NFR in the use case model according to [16] and [17], as a result of step 4 of CEP

In this case, the NFR framework is used to represent NFR as softgoals, associated with a certain context of action (usually a system component), using the SIG structure [12]. Fig 6 shows a SIG example adapted from [16] to the SOA (Service Oriented Architecture) [24] for an Airline Pricing System context in [19], representing the decomposition of all NFR associated with the architecture of the system context of action, into more reachable softgoals through which the goal of the superior level is attained. Each symbol (a cloud) representing the softgoal is identified by a *name* and the *subject* or *context of action* of the named softgoal and its is denoted by *name [context of action]* [16], for example Security [Account]; it can be decomposed into two or more softgoals that must be satisfied to reach the "father" goal; in the example of Fig. 6, it is represented by a single arch indicating "AND" decomposition, meaning that, for example, that *Portability* and *Compatibility* are needed for the *Customer* context of action; softgoals could also be selectively satisfied and represented by a double arch, indicating "OR" decomposition. SOA for the Pricing System is the context of action for all the NFR that the architecture must satisfy specified as softgoals (*Portability, Compatibility, Reliability,* and *Maintainability*; in our context these QR constitute the SOA quality model); it is decomposed into two softgoals, specified as follows: *Portability, Compatibility [Customer]* and *Reliability, Maintainability [Server]*. The *Portability* softgoal is decomposed reaching immediately an operationalization, meaning that the *Customer*, which is the context of action of *Portability*, will be "solved" or "implemented" by a *Multi-Language Platform*. Instead, to satisfy the *Compatibility* softgoal we need to decompose it farther into the *Interoperability* softgoal, which is operationalized into a *Communication Protocol* as solution. This decomposition process is continued for all softgoals specified for the system, until each one reaches an operationalization for a solution [11].

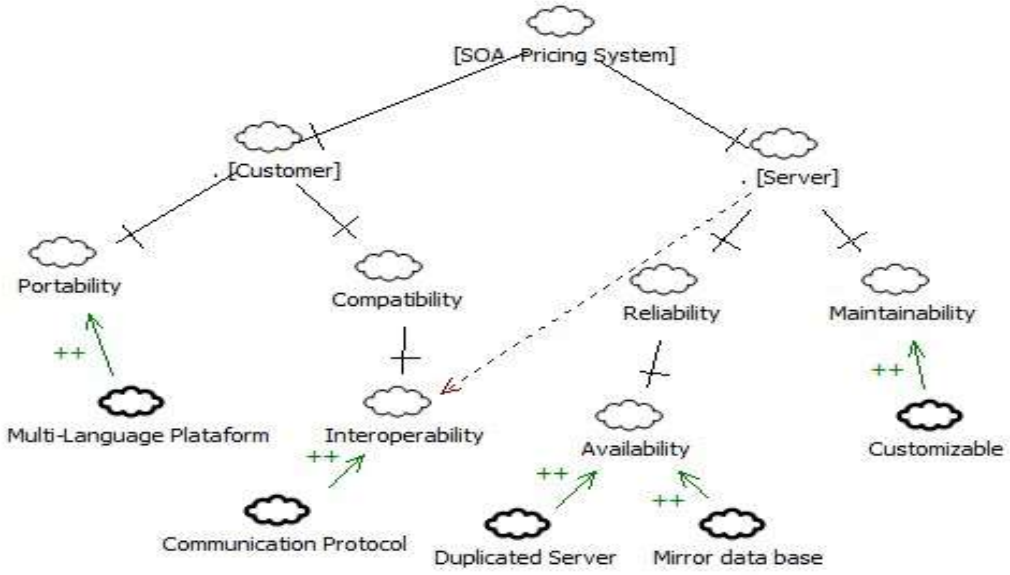

Figure 6: Softgoals Interdependence Graphics (SIG) for SOA (Service Oriented Architecture) [19]

III. Unified Process for Domain Analysis (UPDA)

UPDA integrates the domain quality model according to ISO/IEC 25010 to specify NFR, Aspect and Goal orientations of the AORE approach and the NFR Framework of the GORE approach; the process seeks the integration of these different well-known approaches for the early treatment of NFR into a unique process, to obtain, as the main artifact, the reference architecture for a specific domain. A process can be defined as a series of steps including activities, constraints and resources to produce a particular desired result [25]. In this sense, the UPDA process is presented (see Fig. 2) as a proposed solution for the integration of these approaches. In what follows, the three disciplines of UPDA are detailed:

A. Construction of the domain quality model

This first discipline aims to determine the *Functional and Non-functional Cores of the domain*, to build the *domain quality model* (Fig. 7). Six activities are considered, namely

a) Identify the main functionalities of the domain (*Functional Core*): in this activity all the functionalities must be captured from a detailed domain description or from existing domain knowledge. QR can be associated with each functionality to determine the implicit functionalities.
b) Identify organizational constraints and business rules: it means to identify mandatory regulations, organization rules or laws currently used in the domain. The constraints identified can be associated with the QR.
c) Identify architectural styles of the domain: this activity involves the software architect expertise for the correct use of existing architectural patterns and styles/patterns catalogues to evaluate the "best" architectural solution (s) for the system with respect to the domain QR; this step may involve an architectural evaluation process. Organizational constraints and business rules must be considered. QR must be assigned for each style or solution; they are in general present in the catalogues. The commonality and variability of the domain family of products can be also identified. A selection of styles or architectural solutions that will be used later on in the construction of the RA is produced.
d) Determine the Candidate Architecture: from the precedent functional core and choice of architectural styles and solutions, a first initial or Candidate Architecture is built. The software architect expertise is also crucial in this step.
e) Determine the Non-Functional Core; this activity concerns the assembly of all the QR for the domain, obtained in activities a), b) and c). It is used to determine the "global" requirements of the system which in general correspond to organizational constraints obtained in b) and architectural requirements obtained in c).
f) Specify the Domain Quality Model; in this activity, a table is constructed with all QR obtained in a), b) and c), eliminating repetitions. QR associated with the functional and non-functional core are placed in the table; QR associated with the styles or architectural solutions of step c) represent in general the global QR of the system.

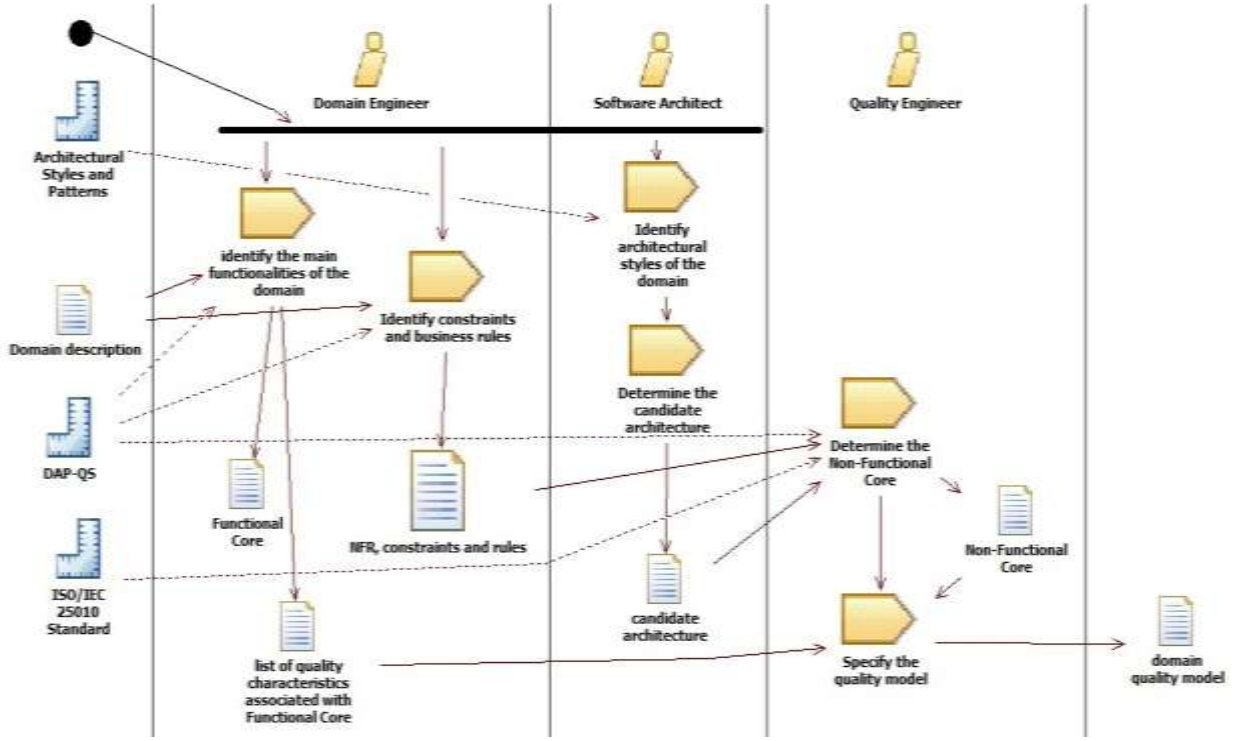

Figure 7: Construction of the Domain Quality Model

Note that all intermediate artifacts obtained in the activities for the construction of the Domain Quality Model will be used in the construction of the RA for the domain, from the techniques described in detail in previous sections (DAP-QS), ISO/IEC 25010, NFR Framework and the Chung Extended Process (CEP), in addition to a catalog of architectural Styles and Patterns taken from the domain assets. As shown in Fig. 7, domain engineers, quality engineers and software architects are involved as actors in this discipline and several artifacts are generated: "Functional Core", "list of QR associated with Functional Core", "list of NFR", "constraints and business rules with corresponding QR", and the "Candidate Architecture", which is constructed from the catalog of styles and patterns that matches the QR of the domain under study; the "Non-Functional Core" that provides the NFR derived from constraints, business rules, architectural style (s) QR. Finally the last artifact produced is the "domain quality model".

B. Identification of crosscutting concerns

The second discipline of UPDA aims to *generate the Extended Use Case Model of the domain*. This discipline, as can be seen in Figure 8, is composed of five (5) activities that allow first the generation of the *Use case Model* and then the NFR identification, global or not, to determine the *crosscutting concerns* for inclusion in the extended use case model. Note that activities a), b) and c) are basically taken from the CEP process, that activity d) involves the AORE approach, and also that NFR are now associated with the QR of the ISO/IEC 25010 Quality Model:

a) Identify actors and associated NFR: the requirements engineer must identify external entities and their associations (specialization/generalization) from the domain description and functional core.
b) Identify use cases: from the domain description and functional core, the software architect must specify the system functionality, expressing it in UML, in terms of the use case model
c) Associate RNF to the Use Case Model: NFR are associated with the Use case Model, using the <<include>> stereotype.
d) Identify candidate aspects: the requirements engineer takes from the non-functional core the global QR that apply to the whole system, and represents them in a new use case diagram; for this step, the *association points* of Supakkul and Chung [16] are used. QR associated with actors are also represented in this use case diagram. The association point establishes a relationship between the actor and the use case, indication the QR involved. Finally, the QR of the functional core are represented as association points directly on the use case.
e) Restructuring common NFR: in this activity, NFR that can be candidate aspects are identified; they are represented in the use case diagram applying the generalization/specialization principles of UML and the propagation rules of Supakkul and Chung [16].

In the development of this discipline, the actors involved are software architects and requirements engineers. These actors take as input the artifacts "domain description", "quality model", "policies and guidelines" issued by the CEP process [19], Brito, Araujo and Moreira [17] for both the extended use case model and the composition tables are followed, generating the following intermediate artifacts: *list of actors and associated NFR, use case model, use case model with NFR and crosscutting concerns*; they contribute to construct the *Extended use case model* that includes domain global NFR and all the identified crosscutting concerns.

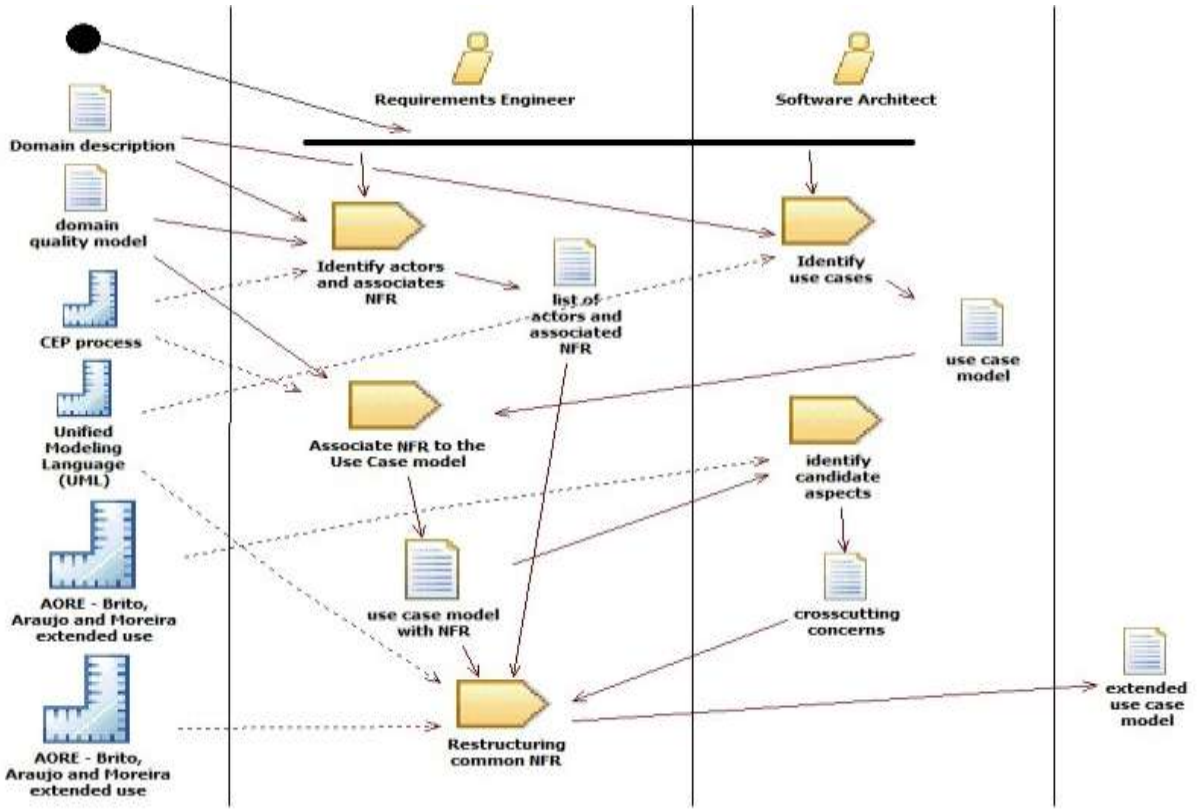

Figure 8: Identification of crosscutting concerns

C. Reference Architecture Design

The third and last discipline of UPDA aims to *design the domain reference architecture* (Fig. 9). The three (3) activities take as input the main artifact generated in the previous discipline, the *extended use case model*; each global NFR is decomposed through the SIG [12]; a SIG for every global non-functional requirement is obtained; the operationalizations of each softgoal are then identified, as the final step to get the domain RA. The activities of this discipline are the following:

a) Refine NFR: the NFR obtained in step A. f) and used in step B to construct the Extended Use Case Model, using the NFR Framework, to generate the corresponding SIG. It must be noticed that global NFR are satisfied by the reference architecture; on the other hand NFR represented as <<inclued>> use cases are specified with the composition tables proposed in [17]; they will be later on satisfied by modules or methods during the implementation stage.
b) Identify operationalizations: the software architect must analyze in the SIG positive or negative contributions (trade-offs) of the QR context of action (component) by a bottom-up process, to relate operationalizations with the global NFR of the components in the superior level.
c) Define Reference Architecture: each operationalization is listed in a table and associated with its possible architectural solutions, the candidate architecture obtained in step A. d) is modified adding these components.

The development of these activities involves two actors, the quality engineer, who must ensure the application of the ISO/IEC 25010 standard for refinement of the NFR and the software architect.

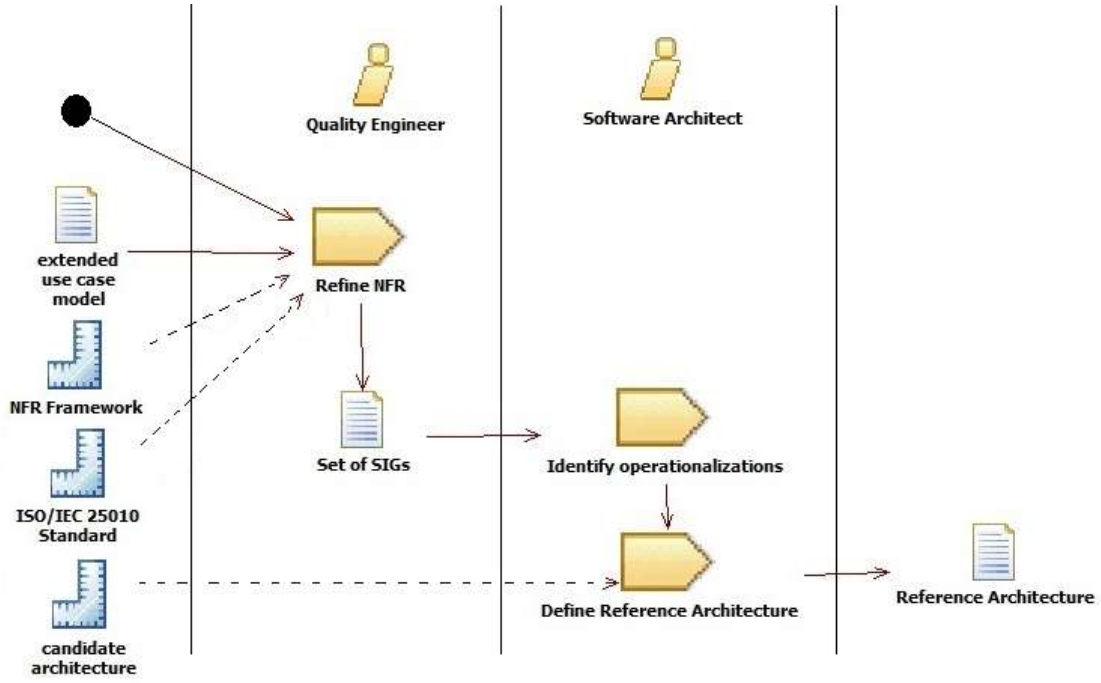

Figure 9: Reference Architecture Design

IV. Case Study: Customer Service Online Banking

In what follows, the UPDA process is validated applying it to a case study, adapted from [26] for the e-banking domain; each activity of the UPDA disciplines is applied for the development of the RA for the e-banking domain.

A. Construction of the quality model

Input: Description of the domain. The online banking, as it is commonly called, is part of the domain of electronic banking. Online banking provides to its clients the majority of bank services through the use of a computer with internet access; electronic banking is the term used to generalize the use of any electronic device that is used to communicate with the bank, including ATMs and mobile phones [27]. The domain of online banking aims at allowing its customers to make ubiquitous transactions (from any place and at any time), through any personal computer with internet access, offering a maximum of security and confidence in transactions. Additionally, it is

important for the online banking, to promote its entire investment portfolios at the largest number of customers, offering security, speed of responses, as well as reliability in the managed data.

a)Identify functionalities of the domain

From the description of the online banking domain, main functionalities can be clearly identified: display of banking movements, transfers between own accounts and third parties, cancellation and payment of services, visualization and acquisition of investments, application and payment of loans; all these functionalities are considered transactions, which are defined in [24] as a set of activities grouped into a unit; a transaction is accomplished as a whole, if any activity fails, the entire transaction fails.

Applying the principle of generalization of processes, we can group some of the functionalities in three main branches, thereby generating the domain Functional Core: a) Internal movements, meaning any transaction with the same bank products; b) movements between banks, which involve transactions between two or more banks and ultimately, implicit functionality such as access control by the type of data exchange occurring in the domain (see table 2).

Table 2: Functional Core

Functional Core	
Access control	•User ID •User password •Reset time
Internal movements	•Account activation •Payment of credit cards •Check transactions •Transfers between own accounts •Transfers to other customers •Loans •Enabling services •Payment for services •Investment advice
Movements between banks	•Payment of credit cards from other banks •Transfer to own account •Transfers to other customers

After determining the main functionalities of the domain, they are with quality characteristics that must be fulfilled for each one of them. The functionality "internal movements" are transactions involving both consultations and transfers and payments within the same organization, so that it requires security, efficiency, usability, reliability and functional suitability. In the case of "movements between banks" in addition to the above-mentioned quality characteristics, support is required for the interoperability of the systems involved. Finally the functionality of Access Control involves the QR of functional suitability, usability, security, and efficiency (see table 3).

b)Identify business rules & organizational constraints

In the online banking domain, there are some basic rules that should not be neglected; a) provide customers with the highest security to perform their transactions via internet; b) generate sufficient service reliability to attract lots of customers; c) availability of service 24 hours a day, 365 days a year and, d) all customer information must be concentrated to carry out basic operations on the clients accounts.

The listed QR are traceable from the functionality requiring them (see Tables 3 and 4) and the precise quality goals to be reached can be measured, if the quality attribute level is specified according to ISO/IEC 25010. Here they are presented as quality characteristics and sub-characteristics; quality attributes are not shown..

The communication between the bank and customers must support different operating systems as well as different access platforms. On the other hand, the implementation of the online banking domain implies indirectly the increase in the amount of transactions, therefore the scalability of the hardware and software involved should be evaluated (see table 4).

As for legal constraints, it is necessary to review the country laws and regulations.

Table 3: List of quality characteristics (QR) associated with the Functional Core

Functional Core (FR)	Associated Quality Characteristics (QR)
Access Control	• Functional suitability (functional correctness) • Usability • Security (confidentiality, authenticity, integrity) • Performance efficiency (time behavior, capacity)
Internal movements	• Functional suitability (functional correctness, completeness) • Usability (operability, learnability, user error protection) • Reliability • Security • Performance efficiency (time behavior)
Movements between banks	• Functional suitability (functional correctness, completeness) • Usability (operability, learnability, user error protection) • Reliability (availability, fault tolerance) • Security • Performance efficiency • Compatibility (interoperability)

c) Identify Architectural Styles

For the online banking domain several architectural solutions are available. In our case, on the basis of previous studies [28] [19], we prefer to use the Service Oriented Architecture (SOA), which follows a layered style, with a client/server communication layer.

The SOA architecture is defined by [29] as a set of patterns, principles and practices to build pieces of software that can interoperate, regardless of the technology used in their implementation; It is normal to apply a design of three-layer which includes the presentation of the client-side, the processing layer, also known as application server, and the data management or database server layer.

Table 4: Online banking domain constraints

Non-Functional Requirements	Associated quality characteristics (QR) (ISO/IEC 25010)
Provide customers with the highest security to perform their transactions online	• Security (confidentiality, authenticity, integrity)
Generate sufficient service reliability to attract lots of customers	• Security (confidentiality, integrity) • Functional suitability (functional correctness)
Availability of service 24 hours a day, 365 days a year	• Reliability (availability)
all customers information must be centralized to perform the basic operations on the clients accounts (centralized data)	• Security (integrity), • Reliability (availability)
Communication between the bank and customers will support different operating systems, as well as different access platforms (Interoperable and portable)	• Security (confidentiality, integrity) • Portability (adaptability) • Compatibility (interoperability)
Increase in the number of transactions must assess scalability of both the hardware and software involved	• Maintainability (modifiability) • Performance efficiency (resource utilization, time behavior, capacity)

d)Determine Candidate Architecture

Candidate Architecture is determined on the basis of the main features of the domain as well as its constraints and business rules. The quality characteristics originated from the architectural styles considered for the domain are identified. In our case, the candidate architecture can be represented graphically as shown in Fig. 10, revealing the layered design with service selection and communication (Internet and firewall) layers.

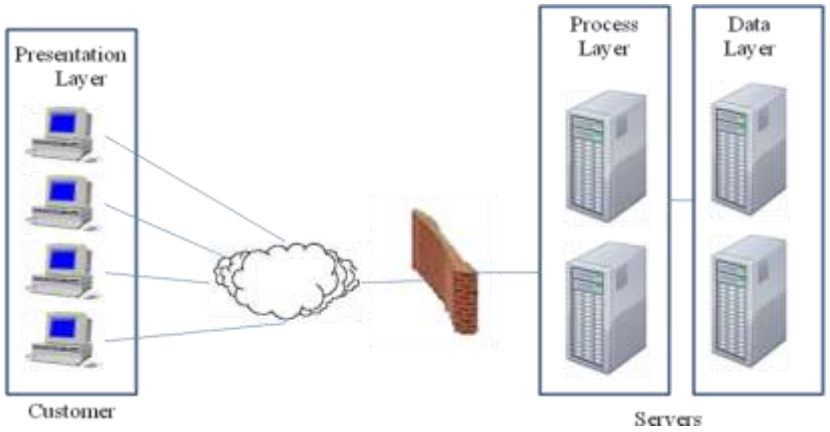

Figure 10: Basic components of Candidate Architecture

The SOA architecture quality characteristics, according to previous studies [20] [19] are *compatibility (interoperability), maintainability (ability to be modified), reliability (availability) and portability (adaptability)*. On the other hand, the domain of online banking belongs to the family of applications based on web services (WS) of transaction type [20] [19], which implies that two general functionalities are clearly identified: data exchange and access control. Table 5 shows an overview of quality characteristics associated with the SOA architecture [19].

Table 5: Quality Characteristics (QR) associated with the candidate architecture

Candidate Architecture		**Associated Quality Characteristics (QR) (ISO/IEC 25010)**
Architectural style	SOA	• Compatibility (interoperability) • Reliability (availability) • Maintainability (modifiability) • Portability (adaptability)
Functionalities of the Transactional WS Family	Data Exchange	• Security (confidentiality, integrity) • Functional suitability (functional correctness) • Performance efficiency (time behavior)
	Access Control	• Security (authenticity)

e)Determine Non-functional Core

The non-functional core is the union of the quality characteristics determined in previous activities (constraints and business rules and quality of candidate architecture); table 6 them in details the online banking non-functional core.

f) Specify the domain quality model

The quality model consists of the set of quality characteristic and sub-characteristic determined in the previous steps, which are specified by the ISO/IEC 25010 standard. The domain quality model helps to define global non-functional requirements (based on the non-functional core) and represent the quality view of the domain knowledge [19]. In this case, the quality model is represented in table 7 with the quality characteristic associated with the originating functional core (*functional suitability, usability, reliability, safety, efficiency and compatibility)*, and of

the quality characteristics of the non-functional core *(security, functional suitability, reliability, portability, compatibility, maintainability, efficiency)*.

Table 6: Non-functional Core of online banking domain

Concept	Associated Quality Characteristics (QR) (ISO/IEC 25010)
Business Constraints	• Security (authenticity) • Functional suitability (functional correctness) • Reliability (availability) • Portability (adaptability) • Compatibility (interoperability) • Maintainability (modifiability) • Performance efficiency (resource utilization, time behavior)
Candidate Architecture	• Compatibility (interoperability) • Reliability (availability) • Maintainability (modifiability) • Portability (adaptability) • Security(authenticity, integrity, confidentiality) • Functional suitability (functional correctness)

B. Identification of crosscutting concerns

The aim of this discipline is to determine which NFR specified in the quality model are candidates to be considered an aspect, by applying the basics concepts of AORE and specifically the approach of Brito, Moreira and Araujo [17], resulting in the extended use case model

a)Identify actors and associated RNF

From the description of the domain and considering the functionalities identified in the previous phase, it can be appreciated that for this domain, the main human actor is the *customer* of the bank. However, two external systems can be considered as actors: the domestic banking system and the external banking system, which we will call *external customer* and *bank respectively*.

The RNF's *security* and *usability* are associated with the customer actor; the external customer actor is associated with *security*, *efficiency* (*behavior in time*), *functional suitability* (*correctness*), *compatibility (interoperability*) and the bank actor is associated with *reliability*

Table 7: The Domain Quality Model

Concept	Associated Quality Characteristics (QR) (ISO/IEC 25010)
List of quality characteristics associated with functional core	• Functional suitability (functional correctness, completeness) • Usability (operability, learnability, user error protection) • Security (confidentiality, authenticity, integrity) • Performance efficiency (time behavior, capacity) • Reliability (availability, fault tolerance) • Compatibility (interoperability)
Non-functional Core	• Security (authenticity) • Functional suitability (functional correctness) • Reliability (availability) • Portability (adaptability) • Compatibility (interoperability) • Maintainability (modifiability) • Performance efficiency (resource utilization, time behavior)

b) Identify use cases

With the information in table 2 (Functional Core) and description of the domain the main use cases for the domain of banking online can be identified on the basic use case model (see Fig. 11).

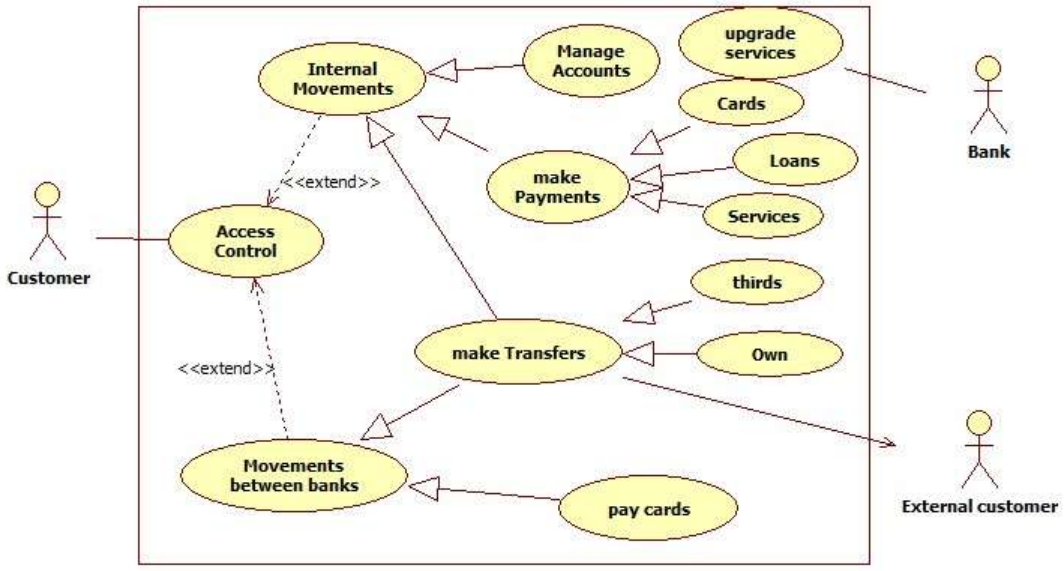

Figure 11: Use case diagram

c) NFR associated with the use case model

Based on the quality model in table 7, Fig. 12 shows the quality characteristics of the Non-functional Core as global NFR (with Association Point on the border of the use case diagram), the quality characteristics associated with actors as NFR with Association Points on the communication between actor and use case, and finally the quality characteristics of the functional core, which are directly associated with use cases.

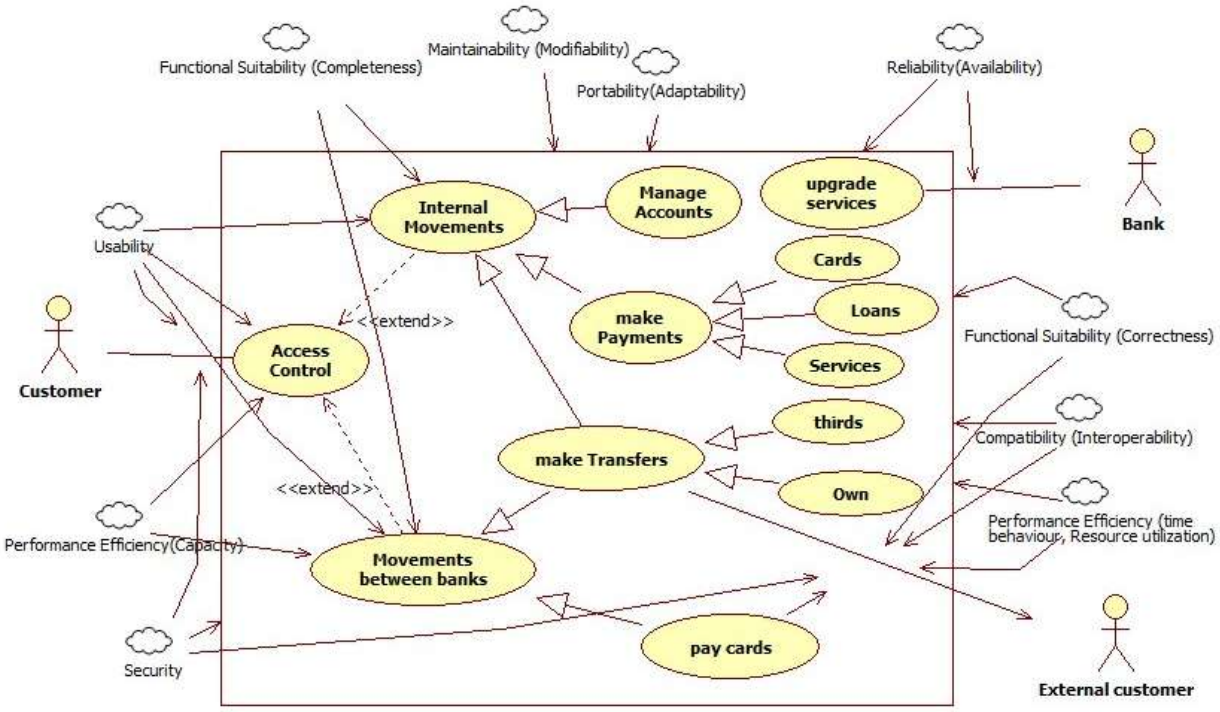

Figure 12: Use case model with associated NFR

d) Identify aspects candidates

To identify candidate aspects it is necessary to remind that according to the aspects-oriented approach, those NFR affecting more than one functionality should be considered as a candidate aspect or crosscutting concern [17]. Table 8 summarizes the association of the functionalities and the crosscutting NFR.

Table 8: Crosscutting concerns

Common QR	Affected functionality
Functional Suitability (Functional correctness)	• Internal movements • Movements between banks
Performance Efficiency	• Access control • Movements between banks
Usability	• Access control • Internal movements • Movements between banks

e) Restructuring of crosscutting NFR

Verify the NFR which affect more than one functionality and apply the rules of propagation [19] within the use case diagram, they are relocated as use cases with a relation of "include" based on the combination of approaches [16] and [17], getting the graph of Fig. 13, as the final artifact of this discipline, which we have called Extended use case model.

C. Reference Architecture Design

Continuing with the main disciplines of UPDA, at this point part of the extended use case model obtained in the previous stage and the graphics of interdependence of objectives (SIG) are generated and each softgoal corresponding to a NFR is refined into an operationalization, to construct the reference architecture of the domain.

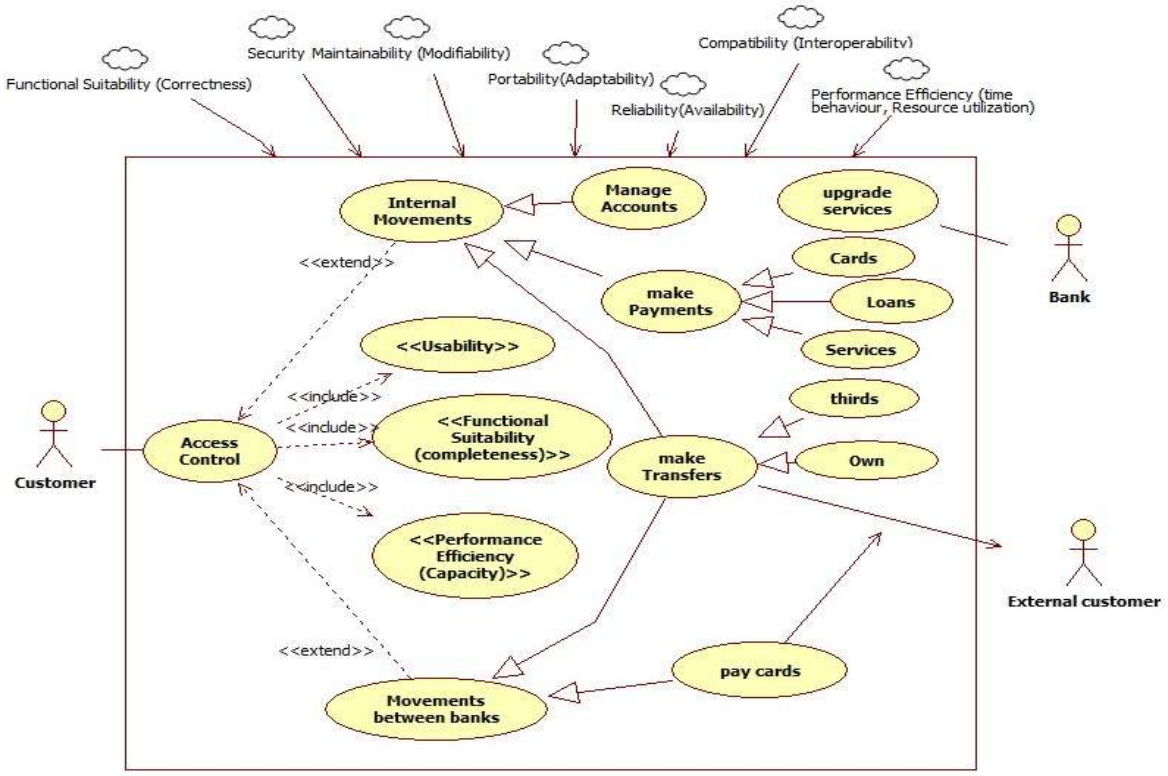

Figure 13: Extended use case model

a)Refine NFR

On the basis of the Extended Use case Model (Fig. 13) each NFR is considered, identified as <<include>> use case, and decomposed using the tree structure known as SIG [12], in order to capture in this structure the sub-goals that satisfy the NFR.

In the same way, the global NFRs expressed in Fig. 13 are considered within the SIG for the selected architecture, since all of them must be satisfied.

b)Identify Operationalizations

The refinement of NFR produces a graph of softgoals, where the last sub-goal can be satisfied with a software component, this is known as operationalization within a top-down approach. In our case, the operationalization of global NFR (fulfilled by SOA) is expressed in Fig. 14.

c)Define Reference Architecture

After operationalizing each NFR to obtain the software components satisfying it, the contribution of each one is evaluated on a bottom-up approach, taking feasible solutions as those who are the "most" positive within the branches of the tree, creating thus the reference architecture of the domain (see table 9). The logic view of this architecture is represented in a UML [30] diagram of components in Fig. 15.

Table 9: Online Banking Domain Reference Architecture

LAYERS	SATISFIED NFR	COMPONENTS
Customer	Portability (adaptability) Functional suitability (correctness)	• Multi-language platform • Browser
Communication	Security (integrity) Performance efficiency (time behavior, recourse utilization) Compatibility (interoperability)	• Communication channel • Proxy server • Communication Protocol
Server	Security (authenticity, confidentiality) Reliability (availability) Maintainability (modifiability)	• Encryption engine • Duplicate server, mirror data base • Customizable (mechanisms to add flexibility to the DB)

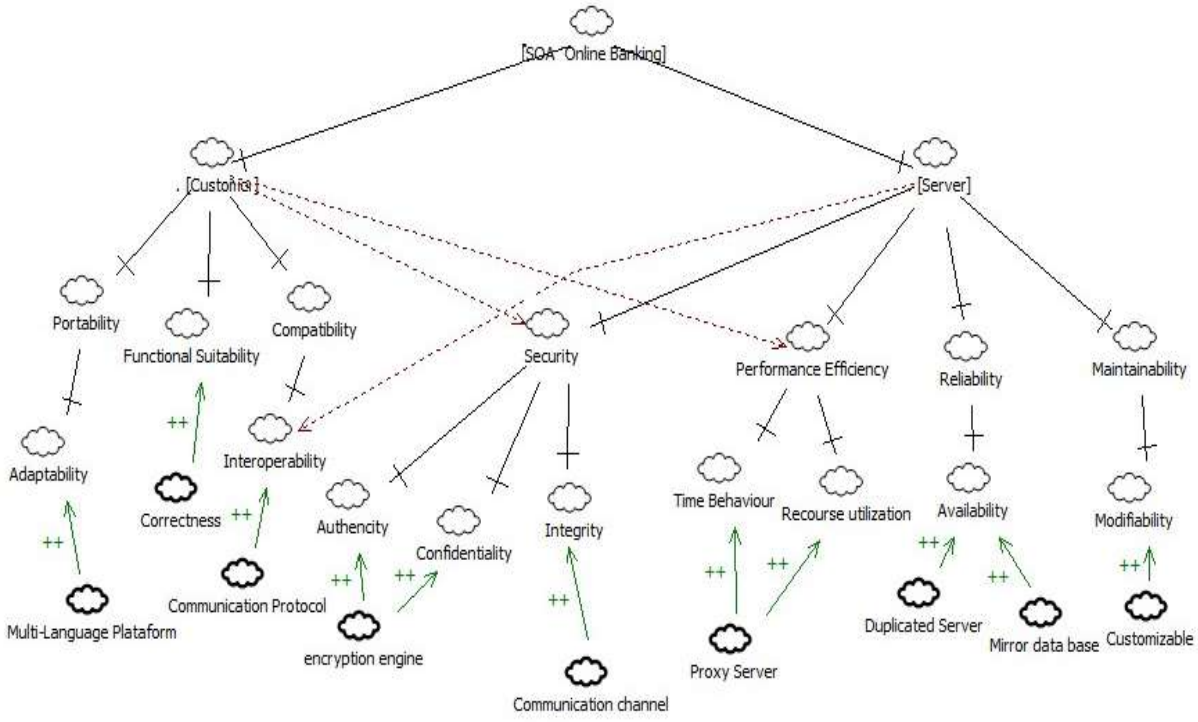

Figure 14: SIG of SOA Architecture for Online Banking

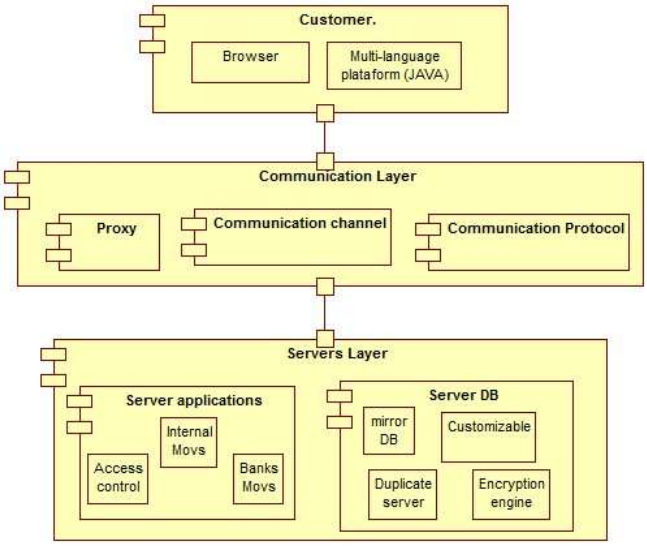

Figure 15: Reference Architecture for Online Banking

V. Related Works

In the review concerning the topics of this research, various proposals can be found in the literature seeking the combination, within the processes for Requirements Engineering, of aspects and goals techniques to relate functional and non-functional requirements.

Among the proposals considered, we can mention the works of De Sousa [31] and Navarro [32].

In the de Sousa methodology, Goal-Oriented Requirements founded on the Separation of Concerns, arises the (GREMSoC) methodology, which aims to produce improvements in requirements specifications, with focus on the separation of concerns. This methodology addresses the crosscutting requirements in the phases of Analysis and Documentation of Requirements Engineering, as shown in Fig. 16 [31]. The process is divided into four main activities: FR analysis and specification, NFR analysis and specification, crosscutting requirements identification and crosscutting requirements specification. For the second activity, they use an adaptation of the NFR Framework proposing the specification of the NFR in specific composition tables, indicating when the requirements are affected and when the crosscutting requirement can be activated by the operators "overlap" and "override", eliminating invasive or direct relationships of crosscutting requirements with the affected requirements.

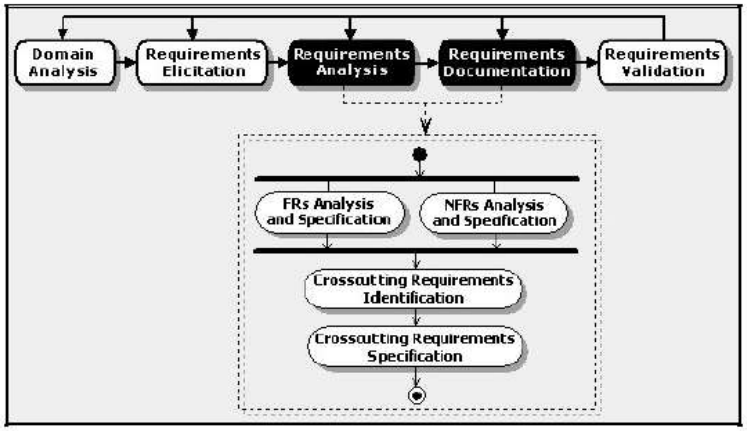

Figure 16: Outline diagram of GREMSoc Methodology [31]

Navarro also integrates aspect and goal oriented approaches, with focus on a proper and organized document of system requirements, taking into consideration the ISO/IEC 9126-1 standard, which is the older version of ISO/IEC 25010, to specify NFR. The proposed methodology [32], called ATRIUM defines software architectures from requirements. It is oriented to the concurrent definition of requirements and architecture through five (5) general activities (see Fig. 17). The first activity of integration of FR and NFR is based on the NFR Framework and the KAOS technique [11][32], obtaining a goals model of the product, defined in [32] as the set of graphs that are interrelated to each other, which is obtained determining stakeholders needs in addition to the review of all existing documentation of the current system.

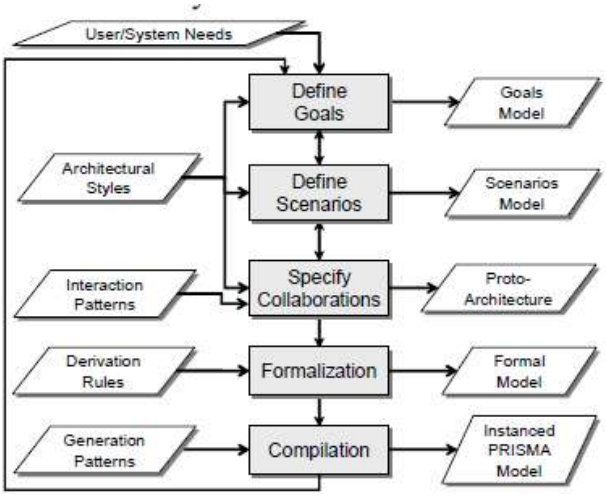

Figure 17: ATRIUM Activities and Artifacts [32]

Both methodologies described above integrate FR and NFR in the RE discipline. The first, proposes improvement to the requirements document based on the proper handling of crosscutting concerns through its specification tables in the phases of analysis and documentation of requirements unlike our proposal that seeks the early integration of FR and NFR and at a higher abstraction level, such as the domain analysis phase. With respect to the second proposal considered, it can be observed that it aims at generating formal derivation rules and patterns by instantiating the goals model through a formal architecture description language, taken from KAOS. In our proposal, a domain analysis process, which integrates well-known requirements engineering techniques enhancing the treatment and modeling of NFR, is used to generate a reference architecture. However, in general one of the main limitations of goal and aspect-oriented approaches is the absence of standards in specification and notations; we used standards as much as possible, responding to best practices of software engineering.

VI. Conclusion

The unified process for domain analysis is proposed as an early stage of a software development process that allows integrating FR and NFR.

The UPDA process allows firstly, to ensure quality throughout the development process, guaranteeing communication among the software development work team, since it is based on the ISO/IEC 25010 standard, and secondly, identify NFR as candidate aspects so that they can be integrated into the process from early stages of analysis and design, facilitating their implementation from the RA, at the end of the development process.

The integration of different known requirements engineering approaches into a unified process was performed smoothly and did not generate major problems, since only essential activities for the process were used; activities and components were reused as assets, focusing on a future work in the context of architecture-centric processes or software product lines. Actually, UPDA can be easily used in a requirements engineering discipline or in a software product line context. The integration of different requirements engineering techniques and approaches at early stages of development into a unified process has the advantages of offering a complete set of a smoothly integrated portfolio of tools and techniques, which have been used separately up to know. The integration process, from a theoretical point of view, was not difficult; models and concepts could be easily integrated. However, it has to be

pointed out that aspect and goal oriented techniques in general, do not offer standard conceptual models or notations, being this a great limitation of these approaches. A future work will be to design a computational tool, integrating existing systems, or designing from scratch new tools, supporting the techniques and practices used by UPDA.

ACKNOWLEDGMENT

This research has been partially sponsored by the PEII-Fonacit DISoft project N° 2011001343, the CDCH DesCCaP project N° PG-03-8014-2011, the CDCH DARGRAF project N° PG-03-730-2013/1, Universidad Central de Venezuela, and the Postgrado en Ciencias de la Computación.

REFERENCES

[1] ISO/IEC 25010. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Systems and software quality models, ISO/IEC JTC1/SC7/WG6, 2011

[2] B. Nuseibeh, S. Easterbrook, Requirements Engineering: A Roadmap. Proc. Conference on the Future of Software Engineering. Limerick, Ireland. 2000.

[3] A. Lamsweerde, Requirements Engineering: From Craft to Discipline.Proc. FSE'2008: 16th ACM Sigsoft Intl. Symposium on the Foundations of Software Engineering, Atlanta, Noviembre. 2008.

[4] E. Berard, *Essays in Object-Oriented Software Engineering*, Prentice Hall, 1992.

[5] I. Sommerville, Software Engineering. 9th Edition. Pearson Education. 2011.

[6] ISO/IEC. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) 2006

[7] M. Shaw, D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. 2d Ed Pearson Us Imports & PHIPEs, 1996

[8] F. Losavio, A. Matteo, I. Pacilli, "Proceso dirigido por objetivos para análisis de dominio bajo estándares de calidad". Enl@ce Revista Venezolana de Información, Tecnología y Conocimiento, 6(3), 11-28, 2009

[9] P. Clements, F. Bauchmann, L. Bass, D. Varlan, J. Ivers, R. Little, P. Merson, R. Nord, J. Stafford, *Documenting Software Architecture, Views and Beyond*. 2nd Edition, Addison Wesley, SEI Series in Software Engineering, NJ, 2010

[10] E. Nakagawa, P. Antonio and M. Becker, *Reference architecture and product line architecture: a subtle but critical difference*, Crancovic V., Grhun, and M. Book (Eds.): ECSA 2011, LNCS 6903, pp. 207-211, Springer-Verlag Berlin Heidelberg 2011

[11] A. Lamsweerde, "A Goal-Oriented Requirements Engineering: A Guided Tour", 5th Int'l Symp. On RE, IEEE CS Press. 2001

[12] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Kluwer Acad. Publishers. 2000

[13] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*. Addison-Wesley, U.S.A. 1999

[14] A. Dardenne, A. Lamsweerde, "Goal-Directed Requirements Acquisition", Science of Computer Programming. vol. 20, pp 179-190. 1993

[15] E. Yu, Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. Proc. RE-97 – 3rd Int. Symp. On Requirements Engineering, Annapolis, 226-235. 1997.

[16] S. Supakkul, L. Chung, "Integrating FRs and NFRs: A Use Case and Goal Driven Approach", 2nd ICSE, pp 30-37, 2004

[17] A. Moreira, I. Brito, J. Araújo, "Crosscutting quality attributes for requirements engineering", The 14th ICSE and SEKE'02, Ischia, Italy, ACM Press, pp 167-174, 2002

[18] G. De Sousa, J. Silva, J. Castro, "Adapting the NFR framework to aspect-oriented requirement engineering", In The XVII Brazilian Symposium on Software Engineering, Manaus, Brazil, October, 2003.

[19] F. Losavio, A. Matteo, I. Pacilli, "Proceso Extendido de Chung con Análisis de Dominio identificación de Aspectos y Estándares de Calidad", II Simposio Científico y Tecnológico en Computación, pp 148-155, 7-9 de mayo, Escuela de Computación, Caracas, Venezuela, 2012

[20] F. Losavio, A. Matteo, R. Rahamut, "Web Services Domain Analysis Based on Quality Standards", 2nd ECSA, Cyprus, 2008, R. Morrison, D. Balasubramaniam, and K. Falkner (Eds.): ECSA 2008, LNCS Vol. 5292, 354–358 © Springer-Verlag Berlin 2008

[21] F. Ruiz, J. Verdugo, *Guía de Uso de SPEM 2 con EPF Composer*. V 3.0. 2008

[22] F. Losavio, A. Matteo, N. Levy, "Web services domain knowledge with an ontology on software quality standards", ITA'09, CAIR, Glyndwr University, UK, Sept. 2011

[23] E. Yu, J. Mylopoulos, "Why goal-oriented requirements engineering". Proceedings of the Fourth International Workshop on Requirements Engineering: Foundations of Software Quality, Pisa, Italy, pp. 15-

22,1998

[24] World Wide Web Consortium, Web Services Architecture Requirements. W3C Working Group Note, 11 February (2004). Copyright © 2004 W3C® (MIT, ERCIM, Keio). Available: http://www.w3.org/TR/wsa-reqs

[25] S. Pfleeger, "Ingeniería del Software. Teoría y Práctica". Prentice Hall, 2002

[26] F. Losavio, A. Matteo, "Reference Architecture Design Using Domain Quality View". Journal of Software Engineering & Methodology, Vol. 3 (1) pp 47-61, March 2013

[27] F. Muñoz, *La adopción de una innovación basada en la Web. Análisis y modelización de los mecanismos generadores de confianza*. Tesis doctoral, departamento de Comercialización e Investigación de Mercados, Universidad de Granada. 2008

[28] F. Losavio, L. Chirinos, A. Matteo, N. Lévy, A. Ramdane-Cherif, "Designing Quality Architecture: Incorporating ISO Standards into the Unified Process". Journal of ISYM, Vol. 21(1), 27-44. 2004

[29] J. Vázquez "¿Por qué SOA?". 2010 http://blogs.tecsisa.com/articulos-tecnicos/por-que-soa/

[30] OMG. (2005). Unified Modeling Language™ (UML®) 2.0. http://www.omg.org/spec/UML/

[31] G. De Sousa, J. Castro, "Towards a Goal-Oriented Requirements Methodology Based on the Separation of Concerns Principle". In: Anais do WER 2003 - Workshop em Engenharia de Requisitos, Piracicaba-SP, Brasil, November 27-28, 2003, pp. 223–239, 2003, http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER03/georgia_souza.pdf

[32] E. Navarro, P. Letelier, I. Ramos, "Goals and Quality Characteristics: Separating Concerns". Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design Workshop, collocated to OOPSLA 2004, Vancouver, Canada: October 25, 2004