Computer Science's Digest

Volume 2

Jesús Insuasti

Oscar Revelo

Alexander Barón

Authors: Jesús Insuasti. Oscar Revelo. Alexander Barón.

University of Nariño. Systems Department. Galeras.NET Research Group

Computer Science's Digest. Volume 2.

ISBN (volume): 978-958-8958-34-7

Edition Number: 1

Cover design: Cover image "Mandelbrot fractal frame" By Firkin, Created 2015-11-15.

Description: A frame based on a portion of a Mandelbrot fractal (Open Clipart).

Proof reading: Katrin Riddle

Layout, printing and finishing: Editorial Universitaria - University of Nariño

Printed and made in Pasto, Colombia.

Insuasti, Jesús
Computer science's digest / Jesús Insuasti, Oscar Revelo, Alexander Barón. – 1ª. ed. – San Juan de Pasto : Editorial Universitaria : Universidad de Nariño, 2017.
v. : il. gráficos. tablas

Incluye bibliografía
Contiene : v.2
Índice de contenidos: v.2. Internet and multimedia technology ; System analysis and design ; Software engineering

ISBN: 978-958-8958-34-7

1. Computadores 2. Informática 3. Programación (computadores electrónicos) 4. Sistemas de almacenamiento y recuperación de información.
I. Revelo, Oscar II. Barón, Alexander

004 I599 - CDD- Ed. 21 Biblioteca Alberto Quijano Guerrero

About the Authors:

Jesus Insuasti, MSc.

Jesus Insuasti is an Assistant Professor in the Systems Department at University of Nariño. His specialties include software design and implementation, embedded systems and game development, amongst others. He especially enjoys working with Microsoft technology. His academic career is as follows: B.S. Systems Engineer (University of Nariño - 1999), MSc. in Higher Education Teaching (University of Nariño - 2010), English Proficiency (San Jose State University - Silicon Valley - 2010), MSc. in Internet Systems (The University of Liverpool - 2014). Currently, Insuasti is finishing his

Oscar Revelo, MSc.

Oscar Revelo is an Assistant Professor in the Systems Department, at the University of Nariño. His areas of interest include Operations Research, Multimedia, and Software Development. His academic career can be summarized as follows: Bachelor of Commerce and Accountancy (Mariana University - 1996), Educational Multimedia Specialist (Antonio Nariño University - 1998), Systems Engineer (Universidad Mariana - 2005) and Masters in Operations Research (Galileo University (Guatemala) - 2008).

Alexander Barón, MSc.

Alexander Barón is an Assistant Professor in the Systems Department at the University of Nariño. His main area of interest is software engineering. His academic record can be summarized as follows: B.S. Systems Engineer (Incca University - 1994), Specialization in Software Engineering (Industrial University of Santander - 1998), Specialization in Higher Education Teaching (University of Nariño - 2002), Specialization in Software Development (Eafit University - 2012), MSc. in Computer Engineering (Eafit University - 2012). Currently, Barón is completing his PhD in Engineering Systems and Computer Engineering at the National University of Colombia

Acknowledgment

The authors want to thank the Systems Department at the University of Nariño for their support in the creation of this series of textbooks. This series is dedicated to the students of the Systems Department, to give them reading material related to computer science in a second language. This book covers Internet and Multimedia Technology, System Analysis and Design, and Software Engineering.

Preface

This series of textbooks was created for the students of the Systems Engineering Program at the University of Nariño. They have been intentionally written in English to promote reading in a foreign language. The textbooks are a collection of reflections and workshops on specific situations in the field of computer science, based on the authors' experiences.

The main purpose of these textbooks is essentially academic. The way in which the reflections and workshops were constructed follows a didactic structure, to facilitate teaching and learning, making use of English as a second language.

The Authors.

CONTENTS

1. INTERNET AND MULTIMEDIA TECHNOLOGY

1.1. REFLECTIONS

1.1.1. Audio Formats

In the following example, Sound Forge Audio Studio 10.0, a professional program, was used; two audio samples were captured: the first, an individual monologue lasting 1 minute and 4 seconds, and the second, an extract of famous song lasting 1 minute.

For the monologue, the frequency used was 44,100 Hz, 16 bits of rate, in stereo mode, and PCM technique to save the information. The file format was Microsoft Wave (*.wav) and the quality of the audio was very good; in this format, no information was lost. The sampling rate used in this first exercise was equivalent to CD quality; however, the size of the file could potentially be a problem, as it was 10Mb.

In the second run of the first exercise, the same monologue was recorded with a frequency of 8,000 Hz, 8 bits of rate, in mono mode, and with the same duration. The difference in size was striking, the second recording generated a wave file of only 470Kb, that is to say the second file is just 4.7% of the size of the first recording. The audio quality was poor, but it was clearly understandable. In this second run, telephone quality was used; this compression technique is frequently applied to speech (Chapman & Chapman, 2009: 310).

The second part of the experiment used part of a famous song. As with the first part, the recording was taken twice. The first one recorded 1 minute of the guitar solo in *Comfortably Numb* by *Pink Floyd*. Using the aforementioned program, the first run was captured using CD quality (256 Kbps, 44,100 Hz, 16 bit, stereo, MP3). The sound quality was good, and the file size was around 1.88Mb producing an MP3 file. However, in the second recording, the same quality was used with another format (128 Kbps, 44,100 Hz, 16 bit, stereo, WMA). The windows media audio V11 compression created a smaller file of 990Kb with practically the same quality. In fact, the difference between the two was practically imperceptible.

In conclusion, radio quality is preferable for recording speech because it generates smaller files and the quality is good enough to accurately capture speech. Windows Media Audio V11 was the preferred format for working with music in this instance, due to its size and quality. The experiment showed that it offers the almost same quality as the MP3 format, but takes up much less space (Spanbauer, 2000: 236).

1.1.2. File Types on the Internet

The Internet is a space in which multiple technologies can share their resources, and the World Wide Web in particular has been a site of huge expansion in terms of application development. Several file types converge within applications through the WWW. Consequently, multimedia productions on the Web have to deal with issues related to the management of media files, due to the diversity of types (and formats) of files, and their differing size and quality, among other features.

With multimedia, different file formats must be used for each medium in order to store and manage information digitally. Binary data are stored according to the definition of the file formats which are used by the multimedia applications, and each file format follows different rules according to its origin (Chapman & Chapman, 2009: 33).

Four resource types are commonly used in a multimedia production; however, they are not the only ones available. These types include text, image, audio and video. Each resource has distinct features in terms of size, compression, quality, encoding, etc.

Take for example a project to create a multimedia production to promote the Carnival of Blacks and Whites, which taketh place every 6th of January in Pasto, southern Colombia; First of all, four resources should be gathered: text, images, audio and video. Given that the carnival has been recognized as World Heritage Site by UNESCO, the information presented should be of high quality and should display the multiculturalism and global appeal of the carnival, to ensure a wide circulation. The following is a quick exploration of each resource from a practical point of view.

Text:

The official language of Colombia is Spanish; however, given the cultural significance of the Carnival, it is important that its history be published in as many languages as possible. Unicode allows text coverage in several languages (Kay, 2006: 42), it is able to work with a large number of words, “no matter what the platform, no matter what the program, no matter what the language (sic.)” (Unicode, Inc., 2012). Size

is not a problem when selecting text in this kind of multimedia production, because all text is written directly in HTML using Unicode formats.

Image:

To do justice to the variety of different cultural expressions in the costumes and chariots adorned with bright colors, the photographs must be taken at high resolution. PNG would be suitable because of its palettes (24-bit RGB or 32-bit RGBA) with additional transparency features. Creative graphic compositions with the superposition and fading effects would help the production's esthetic. Lossless compression allows changes in sizing without compromising on the details, and the alpha channels could be used to make creative composition graphic user interfaces with overlapped images (Roelofs, 2009).

Audio and Video:

Music is also an essential part of Carnival; it would be ideal to record live music played during the carnival parades. The problem lies in capturing the music and not the ambient sound. To avoid this, each piece would be captured prior to the parade in a professional recording studio. If the multimedia production is to be published online, Audio Streaming could be a solution via MP3, Vorbis or AAC codecs. In the case of video, HTTP Live Streaming might be the best way to show the live video recordings; as such, quality in motion pictures is maintained while ensuring little loss in image quality (Apple, Inc., 2011).

In conclusion, a multimedia production for an important cultural event requires good-quality media. The ever present issue of size when it comes to storing information

(especially: audio and video) could be solved via streaming if the multimedia production is deployed online.

1.1.3. Addressing Accessibility in Multimedia Products

Since 2000, the inclusion of those with disabilities has been high priority, in order to accomplish the Millennium Development Goals established by the United Nations. This aim mostly comes under Goal no. 2, which concerns education for all. In reviewing the MDG process, an important step was the adoption of new criteria, in order to create strategies that promote inclusion (United Nations, 2011: 48).

The conclusions in the final report were based on 2 main ideas: firstly, that such goals cannot be reached without the full and effective inclusion of persons with disabilities and their participation in all scenarios and secondly, that technology plays a transcendental role in contributing to this cause (United Nations, n.d.).

Without doubt, the Internet is one of the all-time, greatest achievements in technology; its impact is profound, and touches practically every aspect of human life. One of the environments in which the Internet has had a marked effect, and is widely used, is education. The World Wide Web service in particular is working on the establishment of accessibility guidelines: the WIA –Web Accessibility Initiative.

The World Wide Web Consortium put forth a set of principles concerning accessibility, which take into account the huge diversity of web users and the way in which they use the web (W3C, 2011). These principles are formally called POUR – Perceivable

information and user interface, Operable user interface and navigation, Understandable information and user interface, and Robust content and reliable interpretation.

In the case of creating multimedia productions on the Web, its accessibility should be addressed under the same preceding principles which apply to Web production. There are strategies that can be implemented in order to achieve this with a multimedia product, which are: text size adaptations, close captions, built-in screen readers, navigation assistants, and so on.

When programmers use proprietary plug-ins such as Adobe Flash and Microsoft Silverlight in order to construct Rich Internet Applications, features associated with accessibility should be implemented as built-in components. Fortunately, the plug-in approach is designed to deal with heavy-logic applications and, in this sense, these features associated with the improvement of accessibility are not difficult to implement.

1.1.4. Applications based on X3D

Multimedia applications should entail a high level of interaction with the end users. When it comes to graphics, some flat images are altered to give them navigable features; for instance, geo-referenced systems publish their maps as images where users can zoom, drag, and work with interactive features to get information. In this context, the navigable features have been developed in 2D. However, a spatially more complex application is required for some specific fields: architecture, medicine, engineering, graphic design, biology, art, entertainment, and media, among others.

With the rapid growth of Web-based applications, how 3D graphics are handled has become a point of special interest in computing. Early productions involved VRML as a formal language for illustrating scenarios in 3D, but some issues concerning portability and scalability arose (Behr & Jung, 2010).

X3D is a “fresh” proposal for use in 3D environments in the Web. X3D has previously been supported by the Web3D consortium; in 2007 for example, the first experiments involving the integration of X3D and DOM were carried out, then a full integration with DOM was published in 2009. Finally, the Web3D Consortium developed an HTML/X3D integration model based on x3dom in 2010 (Behr & Jung, 2010).

However, the question still to be answered is where can X3D be applied in current contexts? It can be applied in many areas. According to the list above, any field of knowledge that needs to represent graphic information in 3D is potentially a place where X3D could be used. Some examples are provided below:

1. 3D GIS: “The demand for online 3D terrain visualization for GIS data has increased” (Mat et al, 2011: 31). In this field, users need to obtain topographical information about real terrains with a view to making precise calculations for civil construction. In order to avoid the complexity of creating 3D layouts using expensive software, online web solutions can reduce costs and expand usage.

2. New GUI approaches: Navigation experience can be improved by implementing VRML/X3D (Miyake & Araya, 2009: 28). A map-based navigation system can lend itself to a better experience in terms of movements within virtual 3D environments,

while at the same time allowing users access to an information system. Digital libraries, museums, offices, and so on, can be generated using map-based interfaces by following interactive models for their usage.

3. Medical Simulation: A high quality representation of the human body is an important tool in Health Science students' education and training (Ullrich et al, 2011). A virtual reality system (with proper hardware and software components) is quite expensive, so a web-based solution could be a good alternative. VR-based systems with educational purposes have a close relationship with theoretical approaches founded on HCI (Human-Computer Interaction). Even pedagogical methods can be improved through the use of such technologies.

In conclusion, X3D covers all aspects related to geometry, surfaces, textures and rendering with a view to creating more realistic images. This technology is applicable in several fields of knowledge, when information needs represented in 3D.

1.1.5. Building Web Pages in HTML5

HTML is as old as the World Wide Web itself, because it is the base language in which the Web pages are constructed. This language was created to represent hypertext-like information through integrated media (Berners-Lee & Connolly, 1993). Since its inception, HTML has played its role as a content descriptor within a web page.

Its structure is based on tags with attributes and respective values. A Web page is just a text file where tags and content are located by following syntax rules. As with all computing technology, HTML has its own evolutionary process; in this regard, some improvements have already been made in its extensibility, compatibility, presentation and layout, and data management, among others.

In spite of these improvements however, HTML has always been a descriptive language, and its level of interaction with end users is restricted to navigating Web content. With all its versions, HTML lacks the programmable capabilities needed to process information. Nonetheless, some scripting scenarios have been developed with a view to giving HTML more functionality. For instance, ECMAScript, VBScript, and JavaScript are some solutions which add programmable functions to the HTML remit; the latter is widely used today and it has given rise to some interesting developments, such as AJAX.

Today, according to the specifications of the World Wide Web Consortium, the latest version of the Hypertext Markup Language is HTML5. However, it is still considered to be a work in progress (W3C, 2011). HTML5 is also considered to have inherited many HTML 4 and XHTML 1.0 features (Chapman & Chapman, 2009: 370). In this version, some new features are available including: New HTML elements, Geolocation APIs, Drag-and-drop APIs, Local data APIs, Forms 2.0, Video and audio support, SVG and CANVAS graphics, CSS3, 2D/3D animation, and JavaScript 2.0, among others (Mathew. 2010: 8). These features can be sorted into the 5 categories outlined below:

- Core Page Structure: New elements are introduced in HTML5 for blocking content on the web page, media management, and form structure. In particular, the element which blocks content establishes a solid page structure within a hierarchy where information can be organized cleanly. Several sections on the page are highlighted to improve the clarity of contents.

- Visual Presentation: In using Cascading Style Sheets level 3, this language can improve web pages aesthetically. Color management, special effects, and UI themes are some of its features that can be utilized without needing any additional software components or plug-ins. It is possible to embed fonts via formal style descriptors, and transition effects are easy to design and to implement.
- Graphical Tool: The use of canvases and graphic elements allows Web designers/programmers to build complex interfaces and picture-based representations. It is possible to create 2D and 3D animations using some primitives.
- Rich Media Support: Without doubt, video is one of the main attractive features in this new version. HTML5 is an important solution that works independently of external plug-ins (Cugnini, 2011: 22). Its video quality and compression techniques facilitate the broadcasting of media rich information.
- Enhancement to JavaScript: The incorporation of APIs is a crucial way of giving the language functionality. As such, JavaScript has a point of evolution as an embedded scripting language within HTML5. DOM is widely supported and powered by new features in graphic mode.

In conclusion, HTML5 is an excellent choice for current web designers/developers to face today's challenges. It is important to note that this solution is open in terms of standardization, device-independent, and free of third-party software. Nonetheless, proprietary technology in terms of Web Development is an alternative to the construction of rich web solutions. HTML5 need not be seen as a rival of proprietary technologies such as Adobe Flash or Microsoft Silverlight because they complement each other. HTML5 and proprietary solutions can coexist to construct sophisticated, cutting-edge web applications to fulfil modern expectations.

1.1.6. Considerations on Graphics

Graphics has always been an aspect of computing which has attracted wide interest and enthusiasm. As hardware capacity has increased, graphics have been improving in quality, going from simple 2D primitives, to complex, 3D pieces of art. However, the computational cost involved in producing high quality graphics is reflected in its requirements in terms of physical size (for storage), memory allocation, and data processing.

Two main ways of visualizing graphics were conceived: bitmapped and vector graphics. The former simply saves information pixel by pixel, that is to say no objects are described within them. In this vein, the quality depends on pixel density, among other factors, and as such, using this type of pixel by pixel storage, files can be large (Chapman & Chapman, 2009: 102). The latter is based on geometric principles expressed in mathematical functions. 2D primitives can be shown as independent objects with points,

lines, curves, colors, gradients, etc. Vector graphics are ideal when the images are to be transmitted, due their being small. In addition, they allow for resolution independence, because they are scalable without losing any of their quality (Chapman & Chapman, 2009: 60).

As an example, a detailed explanation of the application for each type of graphic is provided below:

1. Botanical drawings: Assuming the definition of a botanical drawing is a sketch of botanical species for taxonomy purposes, it is normally also assumed that it is a skilled, freehand drawing. According to Karen Reeds (2004: 248), biology students in the past had to painstakingly draw species in all their detail, and it certainly was not an easy task. This approach has changed over time due to advances in technology (i.e. digital photography). Now, in theory, botanical drawings could be extracted from digital bitmapped images in the form of vector graphics using specialized programs.

2. Finger prints: As with photographs, finger prints have been widely used to authenticate actions or identities for generations. Much investigation has been carried out with a view to perfecting the processing of finger prints, a common feature being that the finger print is captured as a bitmapped image. The main focus for researches has been the processing phase, so as to recognize common patterns in the bitmaps. A standard is applied to the images to identify the curves that match with a given pattern, the bitmaps are put into gray scale, then some filters applied. Edge detection algorithms are run, and the final steps are noise removal and purification (Pathak, 2010: 46). Every algorithm is applied on a bitmap.

3. A 2D cartoon character (e.g. Woody Woodpecker): Cartoon characters are created for two main functions: the first, as an attractive icon to catch people's attention in advertising, the second as a character in animation based media, such as animated short films, video clips or web animations. In either case, the best way to construct cartoons is using vector graphics. Vector graphics are easily scalable without compromising on quality, which is paramount in marketing and publicity, and this type of image facilitates the mechanisms which produce 2D transformations, such as translation, scale, rotation and skew, among others, which is key in producing animation.

4. A satellite image: Apart from their use in mapping systems on the Web, the high resolution images taken from satellites are useful resources for academic studies, due to the exact detail and pixel density with which they are acquired and processed. Satellite images are bitmapped graphics in high resolution with great zoom capacity. For instance, cartographic studies require very high resolution images from satellites because the details they provide are the basis for topographical calculations (Castrianni et al, 2010: 97).

5. A photograph (a person's face): If the main purpose of a portrait photograph is identification, it should be represented as a bitmapped image to show in clear definition the details of a given person. Bitmap facilitates the use of high resolutions, providing realistic details.

1.1.7. Creating HTML Pages

The World Wide Web Consortium established HTML as a standard language for publishing web content. Its latest specifications are HTML 4.01 and XHTML 1.1; HTML5 is still considered a work in progress with substantial enhancements in user interface, media management, data forms, and semantics, to name but a few.

Since its inception, HTML was intended as a descriptive language using simple tags (Berners-Lee & Connolly, 1993). As such, HTML maintains a hierarchical structure. This description-based technique previously enabled the design of simple web pages. Today, with the use of CSS and AJAX Technology, HTML pages have been enhanced in order to face new challenges arising from current information management requirements.

There are different ways of writing or generating a code with a view to constructing a HTML page. The first technique is based on the fact that a HTML page is a text file, and as a consequence, one can type the code and its content directly into a text file with a simple text editor such as *Microsoft Notepad* or *GNOME's gedit*, or by using command-line editors such as *GNU's nano editor*, or even *UNIX-like vi* editor.

The main feature of this first technique, in constructing HTML pages by typing the entire code, is that it necessitates a profound knowledge on the part of the programmer. The editors do not have the complete set of tools relating to HTML syntax; only a few of them for highlighting the HTML tags. They lack full syntax analysis, so it is the responsibility of the programmer to correct the construction of the tags in terms of syntax and hierarchy.

When building straightforward HTML pages, typing the entire code is a viable solution; when the pages are more complex however (in terms of user interface, graphic presentation, and aesthetic design) this is not an appropriate technique. A programmer may waste a lot of time working without the tools that facilitate professional construction of HTML web pages.

Another way to construct HTML web pages is by using professional HTML designers, such as Adobe Dreamweaver and Microsoft Expression Web. The former has multiscreen preview panels, jQuery integration, CSS3/HTML5 support, and a new Live View Rendering (Adobe Systems Inc., 2012). The latter has powerful features for syntax checking called *Intellisense* technology, fast design based on wizards, AJAX support by several libraries, and *SuperPreview* visors for the final presentation in different web browsers (Microsoft Corp., 2012).

The advantages associated with the use of these kinds of HTML editors are evident in the quality of the web sites they produce. The time they save is also a key factor in favor of their performance in the workplace. Using these professional editors increases productivity in web page construction. As these software products belong to the proprietary line they are of course costly. However, there are some open-source alternatives, such as: Bluefish editor, Amaya Editor/Browser, and Online web editors based on CMS portals. These products are completely free; the software packages can be installed and used on a computer, but the online web editors can produce “live” results. The online web editors may have restrictions in the code used and some limitations in programmability. It all depends on the policies of the web site based on CMS.

In conclusion, it is preferable to work using proprietary software when creating HTML pages due to their powerful tools, WYSIWYG approach, compatibility options, standard-compliant, wizards, and so on. The open-source alternatives are good, but they have less functions than the proprietary solutions.

1.1.8. Criteria for Selecting Video Clips

The promotion of the Carnival of Blacks and Whites demands well-chosen media in terms of quality, fidelity and performance on the Web. Video in particular is a powerful tool for expressing ideas and making a strong impact; in a few well designed scenes, a message can be delivered more succinctly than by using any amount of written text. As such, video clips must have a good design, and be of good quality at the moment of broadcasting.

In the construction of a Web-based multimedia product, the video clips have to be in a digital format. The term “digital video” is associated with the storage and manipulation of the information digitally within video sequences; the technology related to these tasks comes from computing solutions (Chapman & Chapman, 2009: 199).

The multimedia project was conceived using Microsoft Silverlight technology, so the media elements within this application will be 100%-compliant with the supported formats and protocols. With this, the following standards should be used in order to work with media on the web:

Container	Supported Codecs
MP3	Mp3
ASF	Audio: WMA standard and WMA professional Video: WMV and VC-1
MP4	Audio: AAC-LC, HE-AAC v1 (AAC+), HE-AAC v2 (EAAC+) Video: H.264

Table 1.1 Containers and Codecs

Source: MSDN (n.d.)

When installing this web application, the delivery technology is based on progressive download via supported containers such as: ASF (WMV/WMA), MP4, and MP3. Within a Microsoft Silverlight application, a media element can be displayed through an active URL, so the property source of a media element should point to a URL; this means that the protocol HTTP is used.

In accordance with the previous information, all sources should be captured in MP4 format using H.264 and ACC compression for video and audio track. Such formats provide good quality video clips. At the moment of playback using Microsoft Silverlight, the MP4 video clips will be converted into a streaming format WMV; as such, video clips can be downloaded progressively within the web application on the client's browser.

In practice, these criteria were used to obtain video clips of 3 minutes, and no more than 8 Mb (480 x 360 pixels, 512 Kbps, stereo, 44100Hz). The tool used was Microsoft Expression Encoder 4. All video clips were supported by the Silverlight Application the initial test showed a good response time from client's browser.

In conclusion, the media is available online via the Microsoft Silverlight media player, video clips are fast to load and play, and most importantly: they are of high quality.

1.1.9. Definitions of Multimedia Systems

The term multimedia has several definitions, but the common element in all of them is the usage of more than one medium. According to Havaldar and Medioni (2009: 1), the term “multimedia” applies to all applications which use different types of media, such as text, audio, video, graphics and animations. However, when specifying a multimedia *system*, the definition must be extended to include certain other qualities: digital storage, voluminous information, interaction, real-time response, and synchronization (2009: 5).

Other authors such as Steinmetz and Nahrstedt (2010: 6) state that Multimedia Systems integrate with other more complex systems. Operating Systems, Networking, Communications, and Real-Time Systems are the base on which Multimedia Systems function. The definition of Multimedia Systems is, more or less, the usage of several mixed media, according to a previous well-designed plan (2010: 1).

The simplest definition of multimedia is that it is the combination of several media (Chapman & Chapman, 2009: 7), but for our purposes it must, of course, be digital media. Most authors agree on the most basic definition: multimedia is the usage of more than one digital medium to express something. From this springs the main purpose behind the construction of a multimedia system; the term system merely refers to the organization of elements with a clear purpose.

Arguably, the most appropriate definition of “multimedia systems” should include the integration of digital media and its following the principles of a system. In other words, the integrated elements respond to a well-defined system design, with a clear purpose appropriate to a specific area or discipline (or several integrated ones) such as education, entertainment, simulation and training, and author tools, among others. In addition, a multimedia system should meet the guidelines relating to professional developing in software engineering, as it comes under the umbrella of computing applications. Taking all the above into account, the definition below has been formulated:

A multimedia system is an organized set of digital media that have a clear and well-defined objective under the criteria of interactivity and easy usage. A multimedia system in a professional environment, as a software product, should be created following all the appropriate pre-established processes in software engineering.

1.1.10. Design Criteria for a Video Clip and a 3D Animation

Continuing with the theme of multimedia production, the Carnival of Blacks and Whites project previously mentioned requires a video clip and a 3D animation. It stands to reason that those media should be created with high image quality (full color and high resolution) and audio (good quality folk music and narrations). If the promotion and publicity of this cultural event depended on this multimedia product, software and digital resources would play an important role in the making of a video clip and a 3D animation.

The following criteria have been highlighted as relevant in producing the aforementioned short film:

1. A Video Clip

The short film will be created using Microsoft Live Movie Maker 2011. This software is free, and it has basic functions related to professional video editing, such as import clips and pictures, splitting, video effects, transitions, animations, captions, and audio edition among others (Microsoft Corp., 2012).

The piece has 12 main scenes. The first scene is a panoramic view of Pasto which flies in, and then the Galeras volcano explodes, shooting colorful particles which take the form of the word “Carnival”. Finally, the screen divides in 2, black and white.

Obviously, the entire video clip is accompanied by documentary narration and authentic folk music as a soundtrack. The fifth scene is a menu in which each option describes the activities for each day of the Carnival. These short parts of the video show specific events of the day: January 2nd with the Colonies Parade, January 3rd with the Children’s Carnival, January 4th with the Arrival of the Castaneda Family, January 5th with the Blacks’ day, and January 6th with the Whites’ day and the Magnus parade. Finally, at the close, a welcome flashes up.

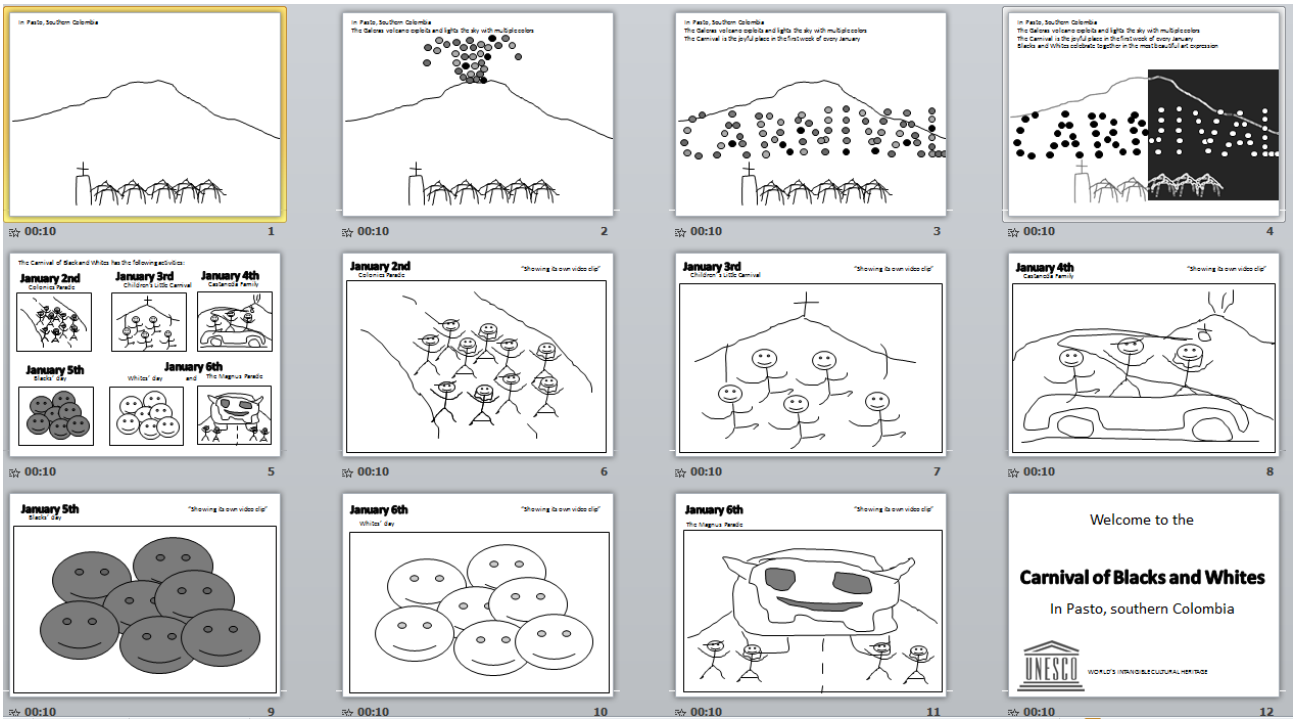


Figure 1.1 General mockups

2. A 3D Animation

All the 3D models and rendered scenes will be produced using Blender; however, some effects such as transitions and audio narrations in the short film will be assembled with Microsoft Live Movie Maker 2011. In accordance with the main theme of the multimedia production, an explanation about how to construct a Carnival Chariot will be provided as a 3D animation.

Following a tutorial model, the 3D animation will develop a step-by-step explanation on how to create a Carnival Chariot, from its design up until its final stage in the Magnus parade. Nine moments are required in the animation in order to show the entire construction. Groups of people assemble to carry out different tasks, some of them design the chariot, others work with the structural model in a big truck and the remaining

group work with the fine art and color management. The final result is an assembled chariot of huge proportions; then follows the design and the creation of the performers' costumes. Finally, the chariot is ready to take its place in the Magnus Parade on January 6th.

In the capturing of each scene, different camera movements are employed to give the best possible view of the process. There will be camera translations, zoom, and 3D rotations, which all enhance the viewer's experience.

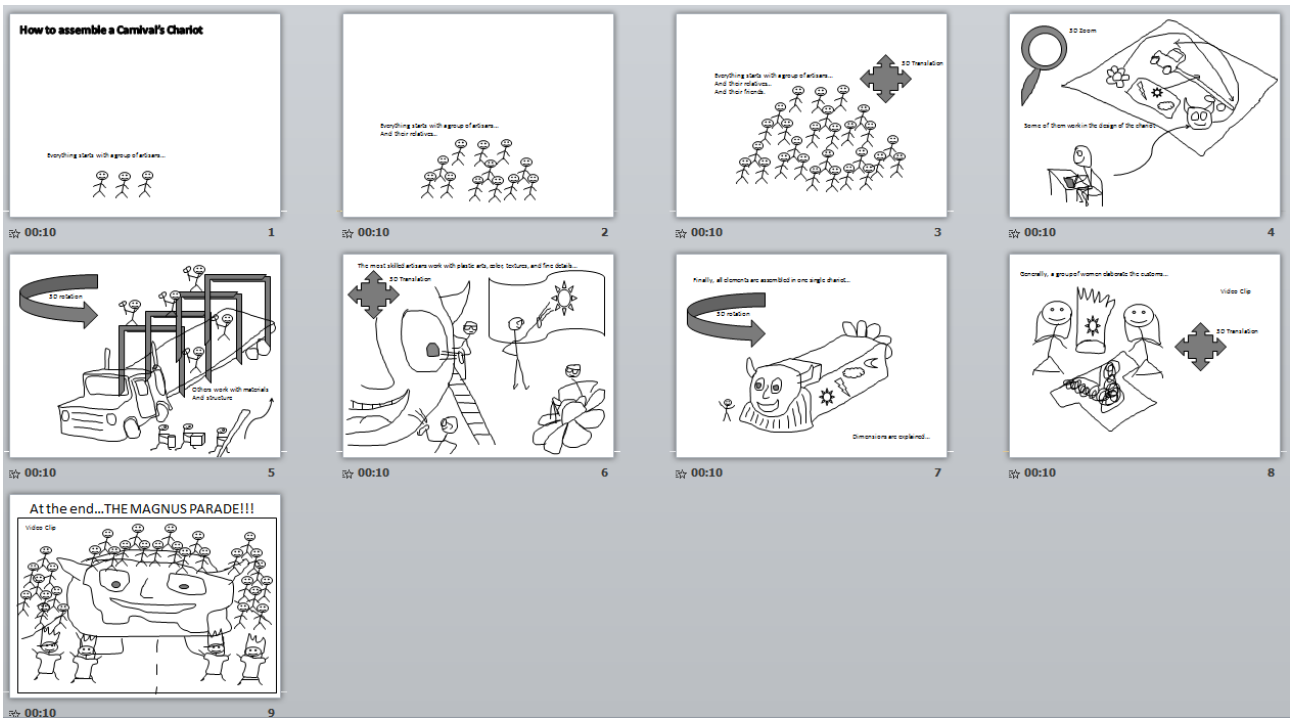


Figure 1.2 Specific Mockups

1.1.11. Image Fidelity

Fidelity is the exact reproduction of something; in this case, image fidelity means that an images' total content must be represented without any modification and preserving

the same quality. The fidelity of images is highly relevant when they are used for security purposes (Lin, Lee, & Chang, 2011).

The main concept behind image fidelity relates to completing the rendering process without any visible distortion (Silverstein & Farrell, 2004: 1). Therefore, technology plays an important role in achieving high fidelity images. Several features are required to improve image fidelity in home computers.

In home computers, the first requirement that must be taken into account is screen resolution and color depth. Today's computers, desktops, laptops and tablets have high capabilities when it comes to graphics; the minimum resolution is 1024 x 768, and their graphic cards have included acceleration and a special infrastructure for 3D and TV performance.

Memory is another key element in image fidelity. The main memory, RAM, is key in rendering processes, and also the "exclusive" graphic memory is a requisite for good results. The latter feature in hardware technology has a close relationship with color depth; good image fidelity can be achieved with 24-bit or 32-bit color depth. Some popular image formats, such as JPEG and PNG (among others), are able to deliver high quality images based on those color depths.

Undoubtedly processors do effect performance, but actually they do not have a direct influence on image quality and image fidelity. Software also influences image fidelity, but in essence, hardware devices make the difference when it comes to quality.

In conclusion, the minimum hardware specifications should be: a computer with at least 1 GB in RAM, a graphic accelerator that supports the highest screen resolutions

(1024 x 768 minimum, preferably with dedicated graphic memory), and software that complies with popular standard image formats, in order to guarantee at least 24-bit color palettes.

1.1.12. Multimedia Support in GISs

Geographic Information Systems are applications which integrate data with georeferences on maps. These computing solutions are used in various contexts where map-based information is required, for example: cartography, urban planning, real estate management, land management, vehicle monitoring system, package tracking, and security and surveillance, among others.

Most Geographic Information Systems use the web-based services provided by world-class enterprises such as Microsoft and Google. These online services have advanced technology for handling satellite images, terrestrial measurements, navigation, zoom, scale, transit paths, identifying addresses, and so on.

Two main web systems are available: Google Maps and Microsoft Bing Maps. The main features of both are described below:

Google Maps:

The first generation of this system used the AJAX model in 2005. Over the last decade we have witnessed many developments in weather forecasting, real estate, tourism, and asset management due to Google Maps; as a tool it provides a geospatial perspective on data which influences decision making in all the above fields (Rajesh,

2011: 36). One of the most important features available on Google Maps is that it is API usable. Mashup techniques can be used to share data originating from different computing resources, such as Web Services, REST, RSS feeds, and so on (Webber & Weins, 2009: 45).

The content of a map can be customized, if the user has a Google account and it is in active session, it is possible to add/edit placemarks, using lines and geometric shapes on the map, and link web content to the placemarks. A useful feature is the freedom to add multimedia content, including 3D constructions and media elements, to each placemark in order to provide more detailed information about map locations (Google, 2012).

Microsoft Bing Maps:

Recently, an increasing number of enterprises have started using Microsoft Bing Maps to promote their services. It has a Bird's Eye viewing option, enhancing user experience, and approximately more than 80% of the USA and Western Europe have been captured using high quality images. Furthermore, its databases expand by around 30 terabytes per month in its updating tasks (Bentley, 2009: 30). As part of a new feature provided by Microsoft, Bing Maps now has integrated airport maps, a useful feature for travelers (Kolakowski, 2011: 7).

As with Google Maps, Microsoft Bing Maps offers a full set of customization, software development and extensibility features. With a Microsoft Live or Hotmail account, users have at their fingertips several features which enable them to: construct 3D buildings, attach web content, have access to multimedia support, and personalize

certain geographic areas. Microsoft Bing Maps acts as a strong foundation for programmers constructing new software based on mashups. Furthermore, it has multimedia interaction enabled via Microsoft Silverlight Technology (Microsoft Corp., 2012).

In comparing the two, both do practically the same job. However, Google Maps has more updated satellite images than Microsoft's in places outside the USA and Western Europe, therefore Google's Street View is more detailed than Microsoft's Bird's Eye. Nevertheless, in terms of multimedia support, Microsoft Bing Maps is superior, as it works with the media management system provided by Microsoft Silverlight Technology. Using this it is possible to integrate video/audio streaming with a Bing Maps-based solution with a high level of performance and quality, whereas Google Maps just creates links with existing web content.

1.1.13. Bitmap Filters

Bitmapped images display information as a huge array of individual dots called pixels, and each one may be a different color (Chapman & Chapman, 2009: 102). As such, bitmapped images cannot display information about graphical objects based on mathematical principles, as vector graphics do. Therefore, specific techniques are required to process bitmapped images, to change the appearance of individual, or groups of pixels.

Altering the color of bitmaps means changing the basic settings, such as hue, saturation and brightness. However, when applying more sophisticated filters and special

effects, more complex mathematical processes are needed. In order to explain the process in the application of some filters, two samples are provided as below using GIMP 2.6.11.

1. **A Pencil Freehand-like photo:**

Figure 1.3 Original image

The goal is to alter the photo in appearance to give it a hand drawn effect. The first filter applied was “Edge Detection”, using the “Laplace” algorithm with 4.5 in Black. Then, colors were inverted.

Figure 1.4 Filters applied

Finally, the last filter belongs to the “Décor” group and is called “Old photo”. “Sepia” default parameters were applied obtaining the following result: A freehand pencil-style photo.

Figure 1.5 Sepia preview

2. <u>Terminal velocity:</u>

In this sample, a real photo was altered to change the setting and give the appearance of speed. The first step was to upload the original photo to the image editor. Then, a background was required; a second photograph with which the previous one was overlapped (Egypt Sons, n.d.)

Figure 1.6 Original image

Figure 1.7 Background

My son and his car were cut from the original photo in order to put him on the road using the GIMP fuzzy select tool. The rest of the original photo was erased and to create a new shape with a transparent background. The following shows the overlapping of the images.

Figure 1.8 Transparent-based filter

Figure 1.9 Final composition

Finally, a motion blur effect was applied to the background with the new relocated object within the “Blur” group. Its parameters were: Blur type = zoom, value = 10. The final result: Terminal Velocity.

Figure 1.10 Effects added at the end

1.1.14. Still images for Multimedia Project: The Carnival of Blacks and Whites

As the Carnival of Blacks and Whites was given the title of Intangible Cultural Heritage by UNESCO in 2009, its artistic pieces deserve to be captured in high quality and exact detail. As such, stills from the Carnival are captured using professional, high resolution cameras.

The exhibitions of artistic talent from both amateurs and professionals are manifold throughout Carnival. All social classes, united in one celebration, enjoy a whole week of folklore and traditional culture in music, dance, parades, games, chariots, and so on. To capture the variety of cultural events, several photographs are required; the following table summarizes the preliminary plan for capturing stills.

Expression Type (*Original Spanish* / English Translation)	Meaning	Photographs	Features
Murgas / Street Band	A *Murga* is a small group of musicians in colored suits. Its main role is to play live folk music in different places within the parades.	• Musicians.jpg	• 1.46Mb, 3477 x 1801, 96dpi, 24-bit color depth
Juegos de Cosméticos / Cosmetics Games	On January 5, everyone is painted black, and 06 January, people are painted white. These games are a peaceful show of camaraderie and friendship. In addition, the people of Pasto create drawings in the streets using colored chalk.	• Games01.jpg • Games02.jpg • Games03.jpg	• 167Kb, 679 x 451, 96dpi, 24-bit color depth • 174Kb, 742 x 495, 96dpi, 24-bit color depth • 533Kb, 937 x 1333, 96dpi, 24-bit color depth
Comparsas / Group Dancers	There is a parade almost every day during the first week of January. In each parade, dancers steal the show with their chorography and colored costumes.	• Dance01.jpg • Dance02.jpg	• 176Kb, 677 x 451, 96dpi, 24-bit color depth • 1.89Mb, 3485 x 1781, 96dpi, 24-bit color depth
Carrozas / Chariots	The last day of the Carnival is January 6th (the Whites' day). The Magnus Parade is the main attraction, and its highlight are the *Carrozas* constructed on trucks and dominated by highly colored and stylized figures which fit the Chariot's theme.	• Chariot01.jpg • Chariot02.jpg • Chariot03.jpg • Chariot04.jpg	• 1.11Mb, 1741 x 2373, 96dpi, 24-bit color depth • 1.73Mb, 3441 x 1821, 96dpi, 24-bit color depth • 381Kb, 1600 x 1063, 96dpi, 24-bit color depth • 1.75Mb, 3533 x 1777, 96dpi, 24-bit color depth

Table 1.2 Preliminary plan for stills

It is important to note that all still images are the property of CORPOCARNAVAL (2009), the official institution that organizes, protects and promotes the Carnival of Blacks

and Whites in Pasto. All the images will form part of a web-based multimedia project, the original format of some images will of course have to be modified due to its physical size.

1.1.15. Making a Simple Snail

This section shows how to create a simple snail image with a background. First an appropriate “environment”, and a model for the snail must be identified. In this case, a children’s toy and a public background from the Internet have been chosen, as shown below in figure 1.11:

Figure 1.11. The real model and the background (Channel 5 Broadcasting Ltd, n.d.)

In this exercise, Blender 2.61 (64-bit Windows Platform) was used. In this program, the start is the welcome screen with a box. The first step is to delete that box in order to get a clear 3D environment (select the object with right-click and press X). Then, use the Quad-view screen (Ctrl+Alt+Q). Now, to create the snail, follow the steps bellow.

1. **The Snail's head.**

In order to create its head, 7 UV Spheres and 3 cylinders are needed. The first sphere (Add -> Mesh -> UV Sphere) will be the main part of its head. It is formed in scale over the y-axis (scale) and then it is rotated over the x-axis. Finally, the sphere is put into the position shown in figure 1.12.

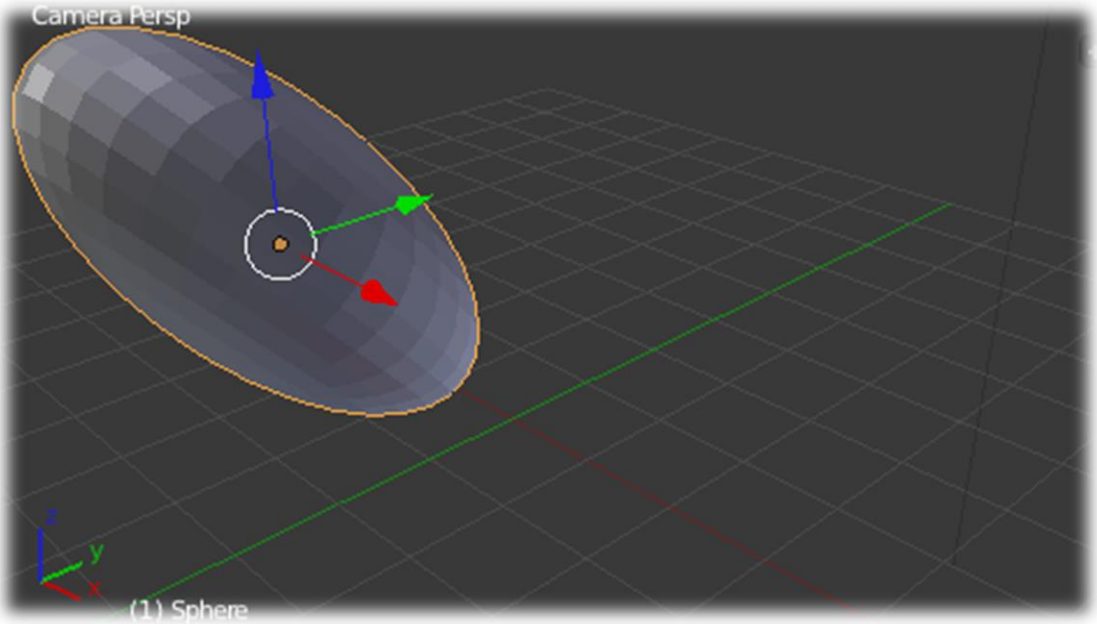


Figure 1.12 The main part of the snail's head.

The eyes are formed using 4 spheres which are scaled and then translated, shown in figure 1.13.

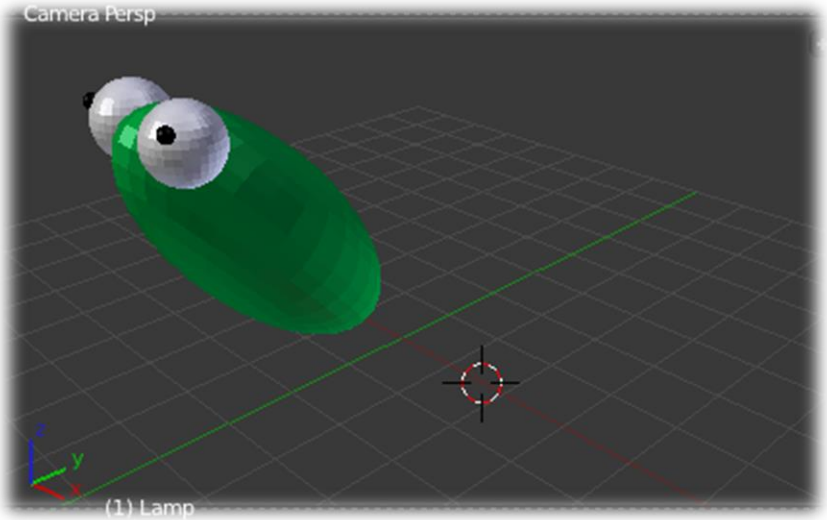


Figure 1.13 The eyes.

The mouth is composed of 3 shapes: a short blue cylinder is required for its smile and the 2 red spheres are its cheeks. All those objects are transformed in translation, rotation and scale as shown in figure 1.14.

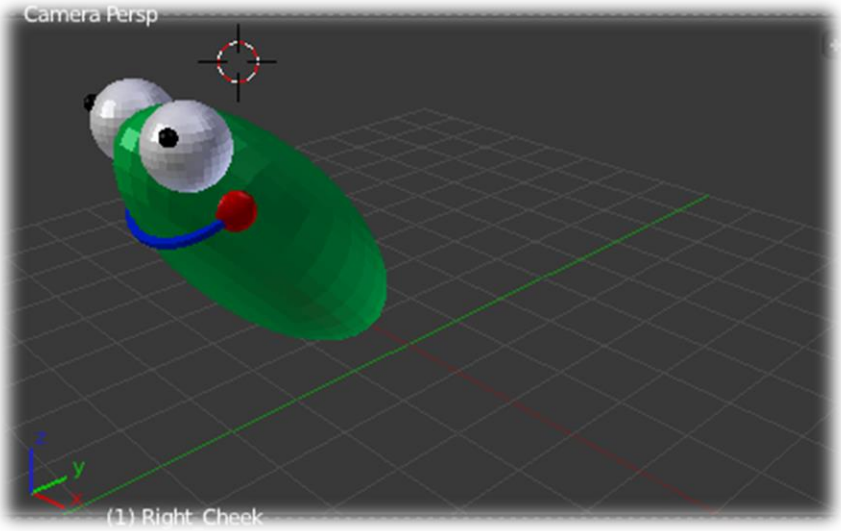


Figure 1.14 The mouth.

Finally, the antennae are created using 2 semi tori. Those tori are cut by editing objects and selecting the faces that were not needed. Then both semi-tori are put in place, see fig. 1.15.

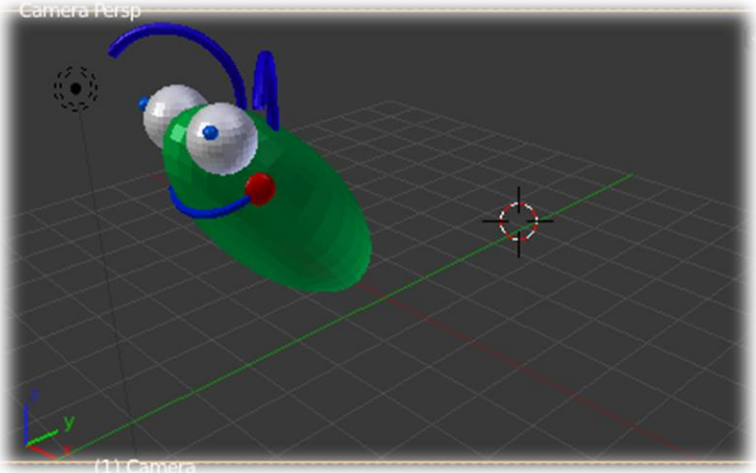


Figure 1.15 Antennae.

2. The Snail's Body

The main body is built using the same flattened sphere which makes up the snail's head. 3 more spherical shapes are overlapped to create the snail's body. Figure 1.16 shows their final position and appearance.

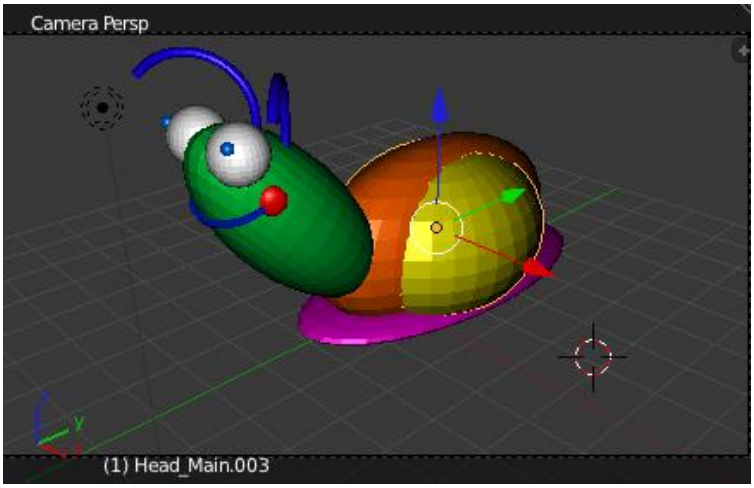


Figure 1.16 The snail's body without the spiral.

The spiral is added by creating a new texture for the flattened yellow sphere. In the material editor, a bitmap (spiral.png) is selected with the repetition parameters (X:2 and Y:1). The final rendering of the body is shown in figure 1.17; the image is produced by using the camera (F12).

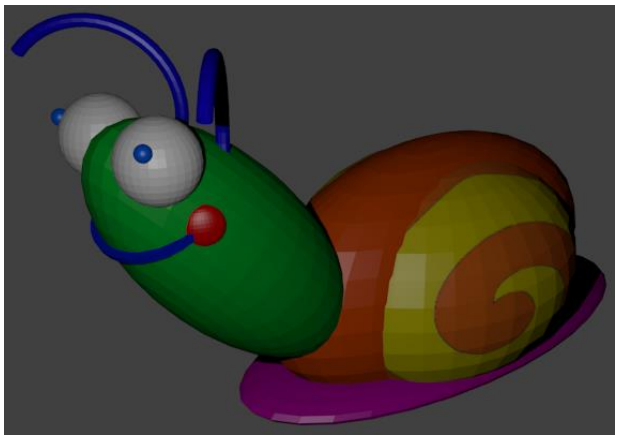


Figure 1.17 The snail's body with the spiral

3. The Background

The last step is to configure the background and put the final touches to the rendering. In order to create a background, activate the camera view in first plane (Ctrl+Alt+Q). From the dropdown View menu, select Properties and check Background images and then select the file "colour_in_snails.jpg". All the elements in the scene are modified by sub-surfaces in order to create 4 sides for each one (to create a smooth surface for each element of the snail). The final result is shown in figure 1.18. For each object a modifier called "subdivision surface" was applied to smooth the edges and surfaces. The background was created in a plane behind the snail with "object" properties in the texture section.

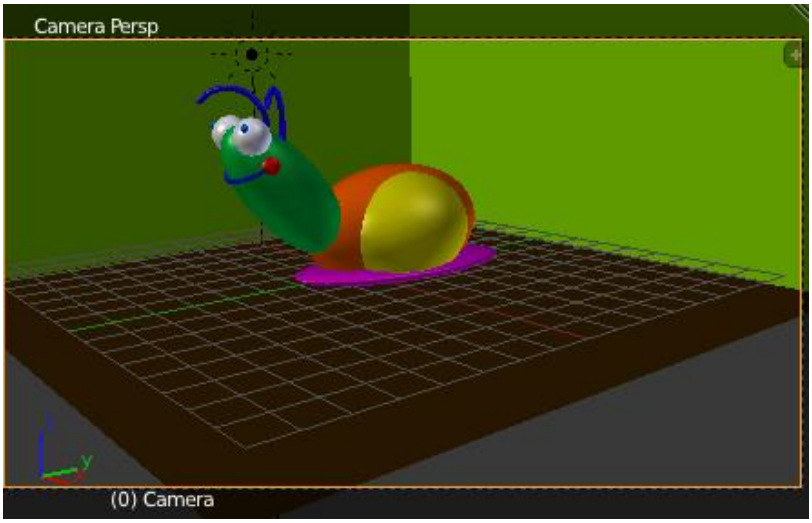


Figure 1.18 Final scene from the default camera.

Finally, the rendered scene is shown in figure 1.19 from 3 angles:

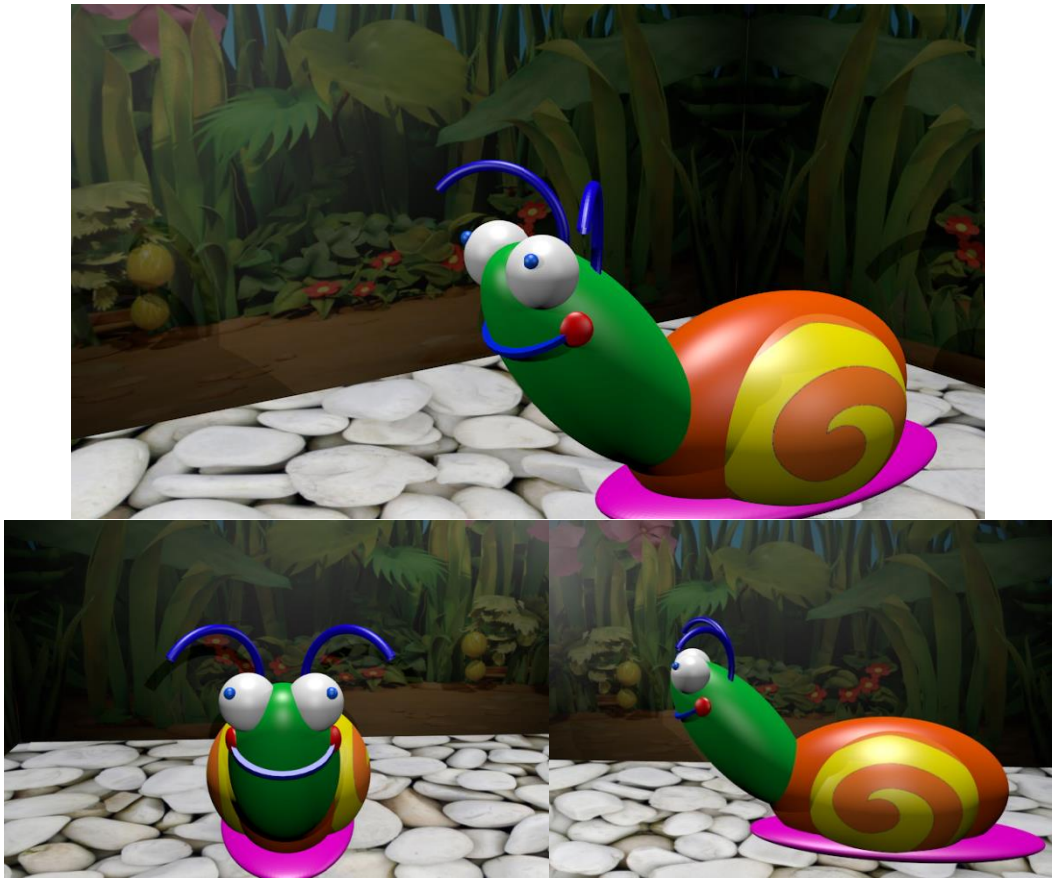


Figure 1.19 Rendered images (camera and other views).

1.1.16. The Web: A Space Under Construction

The web is always under construction. In the sciences, there can be no *final* draft, there is no complete knowledge and there is no absolute truth. With this in mind, the World Wide Web Service (a product of many advances in technique, knowledge, infrastructure, and so on) is a context whose disparate elements continue evolving; one of them is the field of programming languages and software development.

Web applications must be constructed to meet modern requirements and problems. That is why programming languages and development platforms have been created to confront the toughest information management problems on the Web.

At the beginning, HTML was created as a description language, web developers simply “described” web scenarios using hypertext. According to the authors of the HTML’s first draft, the Hypertext Markup Language served to represent the web of information around the globe (Berners-Lee & Connolly, 1993). Now however, the demands on programming languages are more complex than before.

In the HTML evolution process, the most notable changes were those related to data forms, user interface, portability, globalization and multicultural aspects, among others. Currently, the latest version of the language is HTML5; however, according to the World Wide Web Consortium, this most recent version is still a work in progress (W3C, 2011). This latest specification has new features related to APIs extensibility and rich media management, among others.

In recent years, HTML use as the descriptive language which developers would use to give their web products more interactivity and functionality became limited. As a result, developments in dynamic behavior in web applications became available, such as Adobe Flash and Microsoft Silverlight. These proprietary solutions were seen as powerful alternatives for constructing rich web applications; however, they need plug-ins in the client's web browser to work.

In addition to plug-ins, there are helper applications which work in parallel as installed software on the client's machine and give web developments more functionality. These helper applications use computing resources which are separate from the web browser, such as memory allocation and threads. Plug-ins however, are components installed within the web browser; ergo, they share computing resources with their hosts.

These solutions arose due to the classic HTML's having no dynamic response to programmers' needs. Now, the latest specification does provide functionality. This could be interpreted by some people as an "open" response from the World Wide Web Consortium in answer to the proliferation of "closed" third-party developments (Bothelo, 2011: 9).

According to the authors however, HTLM5 will not affect the usage of proprietary solutions such as Adobe Flash and Microsoft Silverlight for three reasons. Firstly, the W3C established in its policies that the Web is an open space for everyone to create; it thrives on diversity. Secondly, proprietary developments have now existed for a long time on the market and continue to flourish; there are trusted products with a strong client

base. Finally, the aforementioned technologies can coexist from a technical point of view, without any collateral issues arising.

1.1.17. Video Conversion

Video, defined as a medium whose main purpose is to capture action and movement, is one of the most commonly used resources in multimedia productions (Chapman & Chapman, 2009: 198). The main problem which arises when working with video is its physical size. This depends on the area, resolution, color depth, audio channels, and compression algorithms used.

To understand the effect of the aforementioned parameters, the same video can be processed in different ways:

The short film used lasts about 3 minutes (Eggleston, n.d.), and was captured using professional software with editing and publishing options. Two different types of conversion were used, see below:

1. **Web Production:**

   In the first conversion process the following data were applied:

Files created:

capture01.mp4

capture01_controller.swf

capture01.html

swfobject.js

expressInstall.swf

FirstFrame.png

Content duration: 00:03:20 (hh:mm:ss)

Content size: 15.22 MB (total)

14.83 MB (movie)

406.37 KB (controller)

Video Dimensions: 634x340

Total Dimensions: 794x340

Production Options:

Frame Rate: Automatic

Key frame rate: 5

Pause at start: Enabled

Bitrate Mode: Quality Mode

Video Quality: 50 %

Audio Bitrate: 56 kbps

Audio Format: AAC

Watermark: Disabled

Table of Contents: Enabled

SCORM: Disabled

Production Preset: Web

Table 1.3 Parameters of conversion

In this case, the main purpose is to create a video for the web. The dimensions of the screen were the same as those of the original version of the video and an MP4 compression system was applied. At the end the file was 15 Mb.

2. Custom PC video:

In this case the same video was used, but this time it was intended for use on a PC instead of the Web:

Files created:

capture02.avi

Content duration: 00:03:23 (hh:mm:ss)

Content size: 28.48 MB

Production Options:

Video Codec:	Cinepak Codec by Radius
Colors:	256
Frame Rate:	15
Size:	320x171
Audio Codec:	No audio
Audio Format:	
Watermark:	Disabled
HTML:	Disabled
Table of Contents:	Disabled
SCORM:	Disabled

Table 1.4 Custom Conversion

It is important to note that a Cinepak Codec was used, and that the dimensions of the final video are almost half those of the version. In addition, AVI format was used to capture the video, with a significant reduction in color depth (only 256 colors). With these reductions in color and dimension, it would be natural to assume that this version is also smaller in size. However, its physical size is actually larger than that of the first exercise. The reason: compression algorithms matter.

In this exercise, the use of special compression algorithms makes the difference in terms of size, quality and transmission time. The first video had good quality performance, while also having good physical size and screen dimensions. In contrast the second, which had a different codec applied to it, reducing color depth (only 256 colors) and almost halving the screen dimensions. Unfortunately, the second video was still larger than the first, and the quality of the second was quite poor.

In conclusion, compression algorithms can make the difference in processing files' features (file size, resolution, color depth, and quality).

1.1.18. Video on the Web: The Case of Adobe Flash

As multimedia content on the Web has always been a topic which has received much attention, and video is arguably the most important element of multimedia, video itself has seen rapid advancements in line with the evolution of technology. The transition of formats from analog to digital did not solve the constant issue which surrounds video, managing its size; and size is still a hurdle in transporting and publishing digital video.

This has given rise to the development of compression techniques (Chapman & Chapman, 2009: 210).

In the race for video domination on the Web, several developers were (and are) competing to establish themselves as *de facto*. The case of one particular company will be analyzed here. Adobe Inc. originally bought the development “Flash” from another enterprise called Macromedia. Since this acquisition in 2005, Adobe developed a strong strategy which led the marketing of Rich Internet Applications. In tandem, Microsoft produced a product of its own called Silverlight which boasts several attractive features. The market is still led however, by Adobe Flash (O'Leary, 2007: 28).

The basis of Adobe Flash is the production of animations in SWF format (called Flash Movies). When these animations come from video sources, it is best to use compression techniques to load the video, so it can be streamed or progressively downloaded (Chapman & Chapman, 2009: 240). Clients merely need a web browser with the proper plug-in installed. Its major limitation however, is its compatibility with mobile devices, some in the industry forbid its usage on specific devices.

The use of video clips on web pages became popular with Adobe Flash technology, and the latest specifications also work with well-known video formats, such as MP4, H.264/AVC, MPEG-2 and MPEG-4, among others. This enables good quality performance at the moment of online viewing when using Adobe Flash technology (Chapman & Chapman, 2009: 241).

Most world-class websites use video, and for its performance and quality, Adobe Flash technology is the first choice for most of them (Braverman, 2006: 14). Outlined below are three scenarios in which companies use Adobe Flash technology:

1. **YouTube**: Perhaps the greatest online video repository, YouTube offers its users the ability to create accounts and upload video content, and it is exclusively managed using Adobe Flash. Although there is a project in progress to upgrade to the most up-to-date technology (i.e. HTML5), YouTube still uses Flash Movies to play video content. An innumerable quantity of websites use the YouTube video repository through embedded code. For example, MIT Open Courseware has many video resources, and most of those resources (perhaps all) come from YouTube's repository.

2. **Yahoo Movies**: The seventh art, the cinema, still has universal attraction. The film industry owes its success in part to advertising via diverse mediums, and especially nowadays online advertising. Yahoo Movies is a paradigm of online cinema advertising, using Adobe Flash technology to publish short films, previews, and movie trailers for fans all over the world.

3. **The Discovery Channel**: Most of its documentaries, TV series, movies and special presentations are videos with Adobe Flash format. People can interact with the website and its media gallery, which has a huge catalog of video clips. Fans of all ages can enjoy their favorite programs online thanks to Adobe Flash technology.

1.1.19. Website Usability

A website's user interface and functionality are key factors that affect its usability (Chapman & Chapman, 2009: 427). If a website's purpose is to handle information, it is expected that its processes and operations run with precision and efficiency (Cappel & Zhenyu, 2007: 117).

There are several common errors, which originate at the design stage, and generate problems in a website's usability. According to Cappel and Zhenyu, some such errors are: excessive use of splash screens and popups, horizontal scrolling, and self-links on the home page, text links without underlying and breadcrumb trails not provided, among others. These all have an effect on how users navigate websites. Lack of organization can cause stress, confusion and frustration for those attempting to use a website (2007: 118).

In their investigation, Cappel and Zhenyu propose a way of measuring the usability of a website outlined in the table below:

Usability Measures and Coding Scheme
For each measure: 1 = Yes (desirable) 0 = No (undesirable)
1) A Splash page is not used (on the opening screen): 1 = Yes 0 = No
2) Horizontal scrolling is not required 1 = Yes 0 = No
3) Homepage does not contain a self-link 1 = Yes 0 = No
4) Text links are underlined: 1 = Yes 0 = No, or they appear only on a "mouse rollover"

5) Text links are blue (Some shade of blue, not necessarily the default shade):
1 = Yes 0 = No, or they appear only on a "mouse rollover"

6) Text link color changes after a link is clicked:
1 = Yes 0 = No

7) A company logo serves as a "Home" link on intimal pages:
1 = Yes (a logo is present, active) 0 = No (no logo is present or if it is, it is not active)

8) A "Home" text link appears on internal pages (or a "Return" link):
1 = Yes 0 = No

9) A breadcrumb trail is provided
1 = Yes 0 = No

10) Site search capability is provided
1 = Yes 0 = No

11) A FAQ or Help option is provided
1 = Yes 0 = No

Table 1.5 Usability Measures and Coding Scheme

Source: Usability Measures and Coding Scheme (Cappel & Zhenyu, 2007: 120).

The questions, and therefore the implicit solutions to common problems encountered by end users, can be put into three categories: Avoidance of Web Design Errors (questions 1 to 3), Use of Web Conventions (questions 4 to 8), and the Inclusion of Features to Promote Usability (questions 9 to 11).

In a practical exercise, the survey was applied to the following website http://www.dokimos.org/ajff/ by 5 individuals, and the results obtained are seen below:

Question	Yes	No
1	5	0
2	0	5
3	0	5
4	0	5
5	5	0
6	5	0
7	0	5
8	0	5
9	0	5

10	0	5
11	0	5

Table 1.6 Responses

In addition to these results, it was noted that the color scheme and background music used for the pages of this website were inappropriate, and that this could cause users some stress. The final results show that this website has serious usability problems.

1.2. WORKSHOPS

1.2.1. The Carnival of Blacks and Whites Multimedia Project

The Carnival of Blacks and Whites was given a World Heritage title by UNESCO in 2008; the Carnival represents the single largest and most diverse cultural exhibition in southern Colombia and it all happens in one place: the city of Pasto.

The beginning of every year, the first week of January, is an enchanting time when people of all social classes celebrate together the majesty of the artisanal exhibitions. Carnival costumes, choreographed dances, popular games, and folk music, all complement the main show of the whole event: The Magnus Parade.

The Magnus Parade takes place on the 6th of January. On that day, a column of enormous floats is the finale at the end of a long string of dancers, musicians and smaller floats. Each float has its own theme chosen by the presiding artist, and all are hand crafted. A typical float can be up to 8 meters high, 4 meters wide, and 15 meters long. On average, 30 of these vast creations take part in the Magnus Parade.

Taking into account the vital cultural importance of this event, it is essential that time and effort be put into its promotion. Creating an online multimedia production would be a good way of achieving this end. The project which follows is based on the construction of an online multimedia production with interactive activities to promote the Carnival of Blacks and Whites.

1.2.2. Resources

This multimedia project should include:

TEXT: History of the Carnival of Blacks and Whites, Description of Main Characters, Myths and Legends of the Carnival, Cultural Heritage, About the Chariots Contest, etc.

AUDIO: The folk music that accompanies the Carnival of Blacks and Whites. Narration. Interviews with Carnival experts. Press comments.

VIDEO: Compilation video of the winners of “the best chariot” with a timeline, from 2000 to present day. TV commentaries. Official opening speeches.

IMAGES: The photo gallery of the Carnival of Blacks of Whites from 2000 to the present day. The Artisans. The Jury of the Chariots Contest. The Costumes.

ANIMATION: An interactive game related to Carnival.

1.2.3. Required Technology

This multimedia project will be created by using:

Operating System: Microsoft Windows 7 (For Development), Microsoft Windows Server 2008 R2 (For Deployment)

Development Platform: Microsoft Visual Studio 2010 (Microsoft .NET Framework 4.0)

Application Style: Web (Microsoft ASP.NET + Microsoft Silverlight)

Programming Language: Microsoft Visual C# 2010

Database System: Microsoft SQL Server 2005 Express Edition

Multimedia Tools: Microsoft Expression Studio 4 (Web, Designer, Encoder, etc.)

All this software is covered by the Microsoft Campus Agreement/Microsoft Academic Alliance with the University of Nariño.

In this case, a web-based multimedia project was created by using Microsoft technology. The specific technology used in the design/development phase is explained in the following table:

Development Platform	Microsoft .NET Framework 4.0
Product Type	Microsoft Silverlight Application 4 + ASP.NET 4.0
Programming Language	Microsoft Visual C# .NET 2010
Web Server	Microsoft IIS / Microsoft Expression Development Server

Table 1.7 Design/Development Criteria

Clients who want to see this website will need an up-to-date web browser (Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, and Apple Safari) and the appropriate Microsoft Silverlight Player 4 or later.

This project is available online at: http://190.254.4.13/carnival/Default.html, and attached to this document is the project folder with 5 screenshots.

The project is based on the following navigation structure:

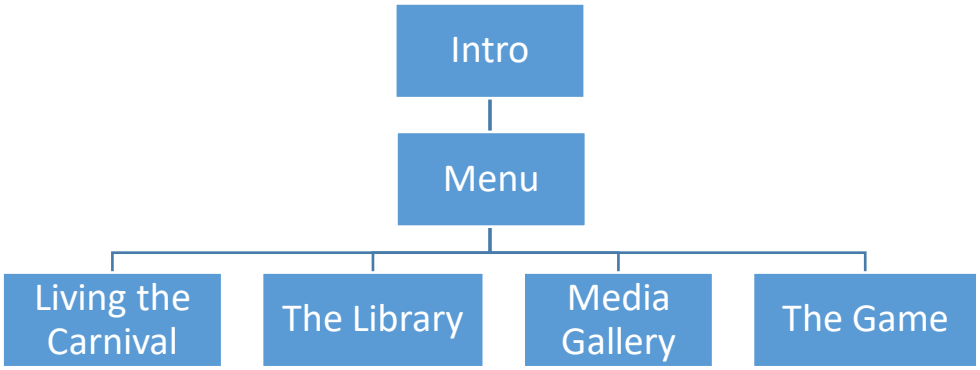


Figure 1.20 Structure

Intro: A welcome to the website.

Menu: Shows 4 options.

Living the Carnival: Each day of the carnival is shown, alongside information about each tradition.

The Library: Snippets of historical aspects of the Carnival, some myths and legends, characters, artisans, manufacturing, etc. Somewhat like a specialized encyclopedia.

Media Gallery: Containing pictures, audio files (music) and video recordings.

The Game: A simple flat shooter with animations.

The platform on which this multimedia application runs is within the client-server model. This product was created using Microsoft Silverlight technology. In this vein, the

server launches the application with a movie (Silverlight component) included in the web solution.

The application is available online at: http://190.254.4.13/carnival/Default.html, it is possible to see this multimedia web application in the following updated web browsers: Microsoft Internet Explorer, Mozilla Firefox, Apple Safari and Google Chrome; Microsoft Silverlight plug-in is required.

The criteria used for the web application design are explained below:

1. **Appropriate choice of a complex application**

   In practical terms, this application was designed for individual experiences. The main idea is to attract tourists.

2. **Original UI. design**

   This application has attractive, themed user interfaces, which are colorful, with special embedded fonts, clear iconography and sound effects. Multimedia resources are included.

3. **Analysis of technical considerations:**

   Special hardware: None, but graphic acceleration in hardware is preferable.

   Storage: 38 Mb, taking into account that it is a multimedia application, this is not unreasonable.

   Communication: HTTP is required

   Software Tools: Microsoft Silverlight player is required.

4. **Budget estimation considering the technical choice.**

Fortunately, the entire project was self-financed by the University of Nariño in cooperation with CORPOCARNAVAL. In spite of the fact that the official institution has its own website already (CORPOCARNAVAL, 2012), this project is more attractive. Microsoft Technology is covered by the Microsoft Campus Agreement and the Microsoft Academic Alliance with the University of Nariño.

We want to thank COPOCARNAVAL for the valuable material provided to create this multimedia product.

2. SYSTEM ANALYSIS AND DESIGN

2.1. REFLECTIONS

2.1.1. A Class Diagram for the "Borrowing" Part of a Library Computer System

In creating a diagram of the "borrowing" part of the library computer system, the following features must be included:

- The library loans books, CDs, DVDs, videos and magazines to registered members.
- The library monitors all borrowed items.
- The library updates the list of available items in its stock.
- The library keeps the information of people who borrowed items with their return info.
- The library manages late returns penalties.

The following figure shows an example class diagram which represents the system described, as a model for requirement analysis (Bennet, McRobb, & Farmer: 2010: 197).

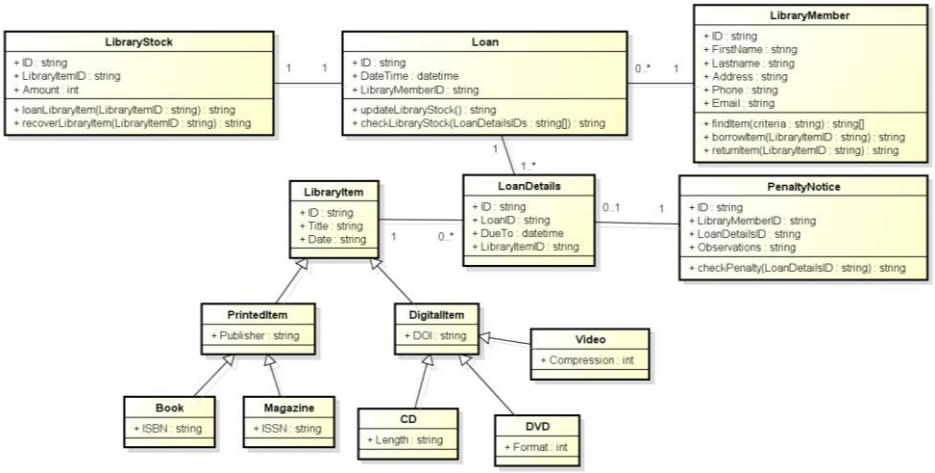


Figure 2.1 Requirement analysis

2.1.2. Design by Contract (DbC)

In the United States, DbC –Design by Contract– is protected by copyright laws. In the field of software development, it is also called Programming by Contract, Contract Programming, or Contract-First Development. This kind of conceptual technology is produced by Eiffel Software, and they claim it promotes a higher level of integration between systems' elements. The system's elements work using clearly defined obligations and benefits (Eiffel Software, n.d.).

DbC is an approach within the object-oriented paradigm and is highly adaptable to the Component-based Development techniques (Bertolino & Mirandola, 2003: 3). In DbC, there are certain assertions: software contracts are specified using logical expressions such as preconditions, postconditions, and so on. It has been argued that DbC could be used in designing robust software, using rules based on logical expressions, these logical expressions belonging to the sequential-procedural paradigm. This means that algorithm logic would be a part of the assertions (Rossel & Manna, 2003: 6). In this case, routines and subprograms would be part of DbC's application in this context.

If a precondition is violated, it denotes a failure on the client's side. However, if a postcondition is violated, the failure may come from the provider. As such, the creation of code units is paramount, in order to test whether the system –or component– is working according to the QA criteria. Another important feature is the possibility of having an inheritance hierarchy in well-formed class structures. This feature could be useful in making complex systems more robust.

The DbC approach could be used in environments where information is processed by sound algorithms. In such cases, the declaration of assertions, following the DbC principles would be a key part of the design phase; it would be possible to create test techniques following quality requirements. However, there is always a risk when it comes to increasing the complexity of a large information system managed following DbC.

2.1.3. A Simple Library Computer System

A library is a place for everyone, where people can find information in books, journals, newspapers, etc. Information is simply knowledge presented one of several ways, and there are a variety of mediums for displaying information: text, images, audio, and video recordings among others, which are all managed and accessed by library staff and visitors.

The people who interact with a library computer system fulfil the following roles: director, librarian, tech support staff (software maintenance, digitization, backups and security, etc.), auxiliary staff, and end users. A User Case Diagram is shown in Figure 2.2 that describes the system's functionality from the perspective of those who interact with it (Bennet, McRobb, & Farmer, 2010: 154).

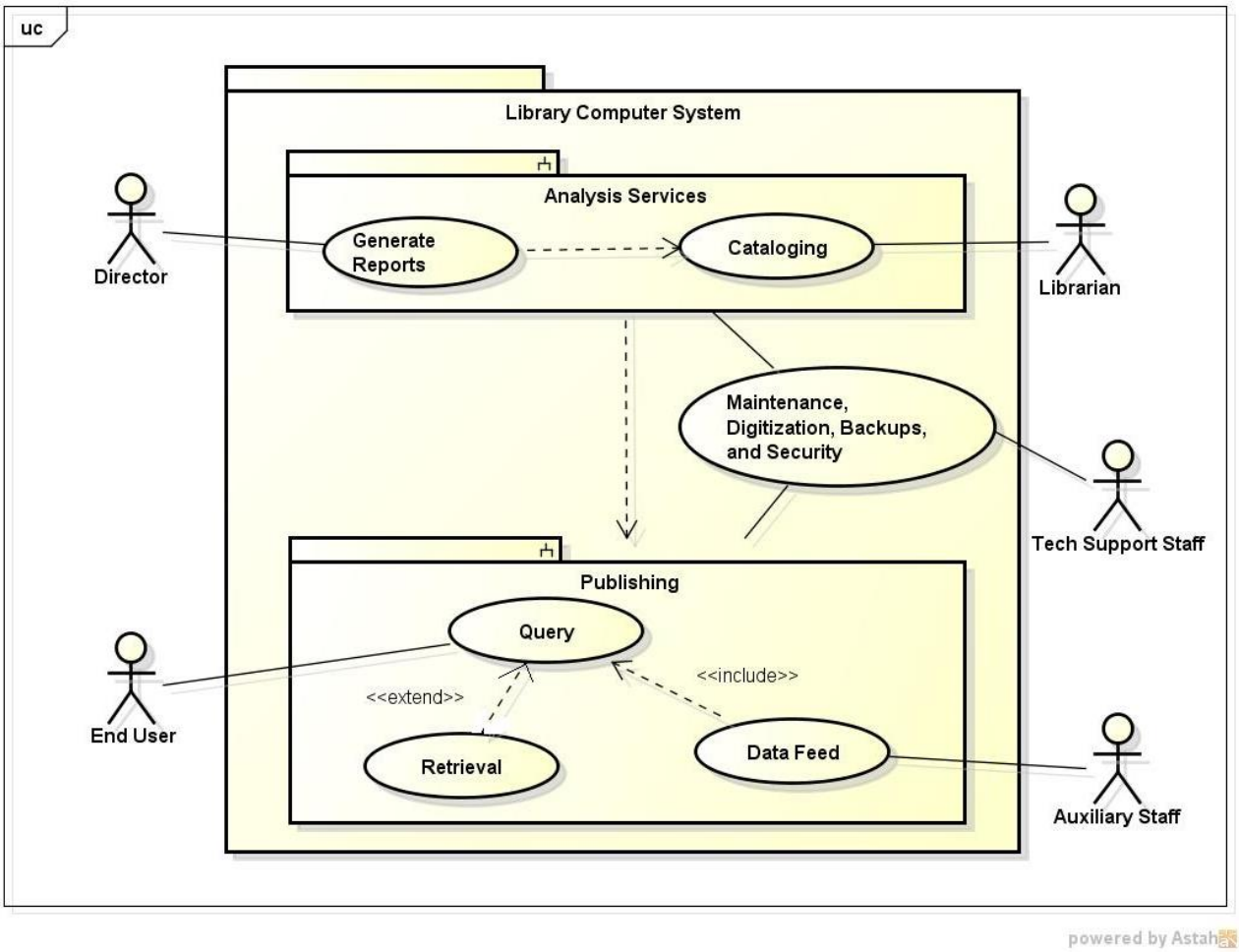


Figure 2.2 User Case for a Library Computer System.

Looking at the above diagram, there are two main subsystems: Analysis Services and Publishing. The former facilitates the director's decision making and the librarians cataloging work. The latter is responsible for publishing, this is the part available to end users, and allows the auxiliary staff to feed data into repositories. The most common functions are Query and Retrieval. Finally, tech support staff are in charge of guaranteeing the functionality of the entire system; the relationship between the two subsystems depends on them.

A list of some non-functional requirements that would not be identified through user cases might be the following:

Criteria	Requirement
Speed	The response time is a key factor in the success of the entire system, especially in query and retrieval tasks. The overall performance of this web-based application should be of a high standard to maintain end user's satisfaction.
Size	Taking into account the technological nature of this system, the physical size of the user interfaces should be small. Particularly mobile clients which deal with limited computing resources – such as mobile devices. As such, the loaded interfaces should be light weight
Easy-to-Use	Users need ergonomic applications to handle information. Modern expectations in the field of computing solutions require the usage of intuitive user interfaces. Especially in academic-scientific environments, users expect up-to-date technology.
Portability	This system would be installed as a series of web-based applications. Thus, users can interact with the system via the Internet. The computing solution would therefore be portable in the sense that any user would be able to interact with it anytime, anywhere.
Robustness	The quantity of end users is increasing. The system should be able to deal with thousands of end users locally and even worldwide. The web-based application should have a top quality infrastructure when it comes to storage, deployment, and memory allocation, in order to handle its millions of records.

Table 2.1 List of some non-functional requirements

2.1.4. OODBMS

The following is a contrast of two approaches to managing information; the relational and the object-oriented approaches. The differences can be summarized as follows:

- A DBMS with a relational approach might be said to have 5 main characteristics: Persistence, Concurrency, Disaster Recovery, Secondary Storage Management and Ease of Consultation. Here, information is separated into data fields in tables.

- An OODBMS contains the previous features with the following additions: Encapsulation, Identity, Inheritance and Polymorphism. There are other desirable features such as Control types and Persistence. Information is stored in its original form, in objects.

The Object-Oriented Data Bases manifesto is the result of a number of experts' working in collaboration and it has been very influential in defining the OODBs objectives. The key features of the proposal for what an OOBD must have, are as follows: the management of complex objects, identity usage, encapsulation, classes as type, inheritance, dynamic links usage, full-supported DML and DDL, data persistency, concurrency control, support for recovery tasks, and an easy way of looking for information (ODBMS.org, n.d.). Taking all this into account, the main structure was established as seen in figure 2.3.

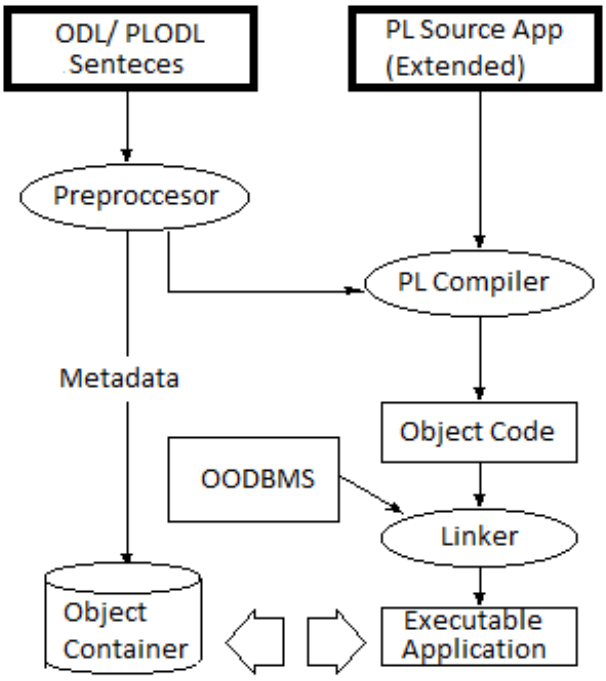


Figure 2.3 Proposed structure for ODMS (ODBMS.org, n.d.)

There are several real life implementations of OODBs, and one of them is called Matisse: the post-relational SQL database. In a test session using this database engine, it was noted to be highly compatible with the newest development platforms such as Oracle Java and Microsoft .NET (Matisse Software Inc., 2012). However, performance when accessing large volumes of information is a key feature that is covered by relational engines. Perhaps, this might be one of the reasons why the relational model has been in great demand in business environments. The object-oriented model was tested in an academic/research context.

Another aspect to take into account is the existence of “hybrid” models, which integrate both relational and object-oriented approaches. Oracle 11g Enterprise Database is considered an Object-Relational implementation; in the same way, Microsoft SQL Server 2012 follows the mixed paradigms. In the history of its use in a

business context, the relational approach has proved the top choice, probably due to its superior performance.

2.1.5. Design Patterns that Interact with User Interfaces

Using design patterns is considered good practice in software design, where reusability is a key factor in constructing software systems with high levels of quality, robustness, and performance (Bennett, McRobb & Farmer, 2010: 422). At the design stage, regardless of the software process, the last stage is design for user interfaces; which means taking into account human-computer interaction. As such, at this part of the design stage it is important to establish a good source of assets for software implementation (Laakso, 2003). There are several proposals concerning user interface design patterns, and one in particular will be explained below.

In the cadre of user interfaces, “form” is associated with the collection of graphic controls which operate software systems. This is the point at which end users have direct contact with the software application. Given the importance of the graphic user interface in today’s systems, the *Presentation-based Pattern* was created to function on different platforms, including desktop PCs, laptops, tablets, PDAs and smartphones, to name but a few.

The MVP design pattern was created according to the following the principles. It is an interesting implementation of the philosophy of separating presentation from business logic, and was conceived as an unambiguous format, representing a graphic user

interface which functioned independently of the context in which it was to be implemented (Seffah & Gaffar, 2006: 1415).

MVP was thought to handle the business logic, enabling a unit test without direct dependency on a specific user interface. In this regard, MVP can be implemented on practically all known technological platforms. Graphically, MVP follows the scheme shown in figure 2.4.

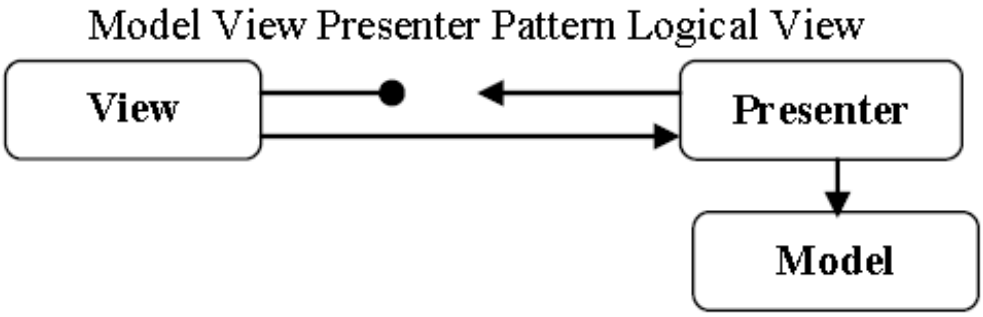


Figure 2.4 MVP Design Pattern (Pau, Mihailescu & Stanescu, 2010: 173)

This design pattern is useful for the implementation of software systems with platform independency. In practical terms, MVP might be applied in the ATM study case, in the scenario of creating the user interface within the ATM physical device. In the MVP design pattern, View would be the ATM's screen; there is an IView –understood as an interface between View and Presenter– that connects the processed information with the screen. Presenter is the item who manages entities, fills data in the view, and so on. Finally, Model is the set of classes that represent the logic of the ATM.

2.1.6. About RUP

In Software Development Processes, one important contribution is made by UP, Unified Process. In terms of methodology, UP is composed of phases, iterations and workflows and an important feature is that it is, “a use-case driven, architecture-centric, iterative, and incremental process” (Bennet, McRobb, & Farmer, 2010: 614). UP is used in large software development environments where complexity demands the participation of lot of people. A “customized” software process, based on the original UP proposal, called RUP was created by IBM. The name change was in accordance with IBM’s Rational® ecosystem philosophy (including theory, software implementation, and certification programs).

Practices within the UP framework include an incremental/iterative development style component-based development as a key factor in the construction of software assets, the use of requirements-driven development, and the centrism in architecture with visual modeling techniques (Bennet, McRobb, & Farmer, 2010: 128). IBM’s RUP inherits all those features and introduces the concept of disciplines in which activities are clearly defined with their respective, expected products, later called assets.

RUP –Rational Unified Process– is a proven method for building complex computer systems’ software. Based on the evolutionary approach, RUP’s main goal is building complex software systems engineering along clear guidelines, faithfully following the definition of life cycle phases and disciplines (Ambler, 2005: 2).

According to Krutchen's definition (2004: 24), RUP is a well-defined system development process (computational in fact), often oriented to use technologies based on objects and components. Founded on software engineering, RUP is based on an iterative/evolutionary process and is driven by the requirements and architecture-centric software development.

RUP's proposal model

In general terms, RUP is presented as a Framework Process; the framework is clearly described by a bi-dimensional matrix of phases and disciplines. According to Ambler, RUP outlines a structured method for creating processes (2005: 17). Indeed, RUP promotes adapting the process to be performed in each project to meet a given user's particular needs. This model clearly calls for organizations to standardize processes to respond to particular needs.

In summarizing the proposal, it has two main dimensions:

Dimension	Components
Phases	Inception, Elaboration, Construction, and Transition
Disciplines	Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration & Change Management, Project Management, and Environment

Table 2.2 Dimensions and Components

Consequently, each intersection (between phases and disciplines) generates software assets, such as: domain model, business case uses model, glossary, views document, additional specifications, design model, architecture document, data model, implementation diagrams, tests and deployment, and transition plan. It is important to consider that each asset represents a set of single assets described in well-formed UML

notation. Figure 2.5 shows the general model of RUP. In short, RUP is highly iterative, feedback is the key factor in improving software production. In addition, RUP produces extensive well-formed documentation in professional environments for the construction of complex information systems.

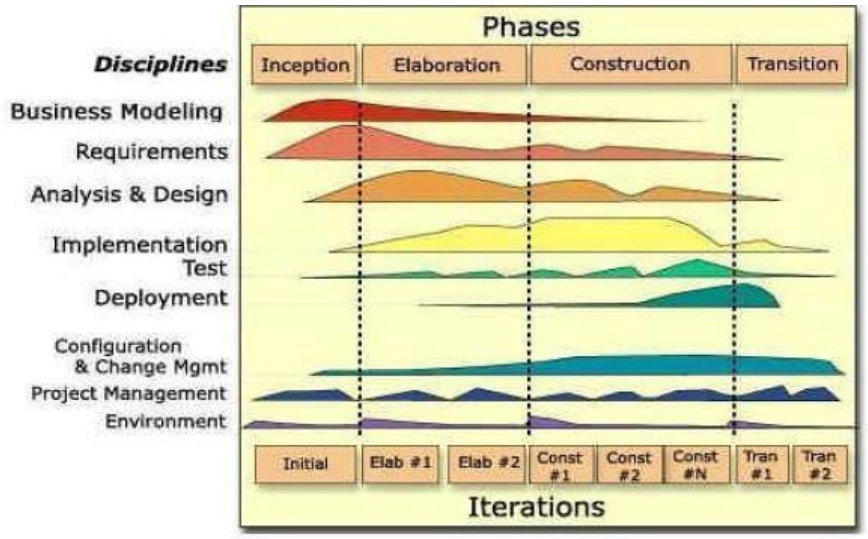


Figure 2.5 Process of RUP proposal (Ambler, 2005)

Reusability and the use of repositories:

According to its creator, IBM, Software reusability in RUP is a priority. According to the authors of the IBM's RUP model, a complete module that is integrated with IBM ® Rational ® Method Composer V7.1 was designed for handling RUP for reusing software assets. That proposal is called Asset-Based Development (Asset-Based Development - ABD) describes a process for reusable asset management, production of reusable assets, and consumption of reusable assets. The plug-in asset-based development also provides tools that support assisted reuse processes. The authors of RUP state that, "Asset Governance establishes the policies, the organizations, enables the teams, establishes the tools, and determines what is to be measured for ABD" (IBM, 2007).

Therefore, the object-orientation approach is a key factor in each and every single documentation's process in RUP. Software assets produced in the different disciplines of RUP are part of large documentary of bodies where nomenclature is made through UML diagrams in accordance with OMG specifications. In this vein, those software assets involve one or more diagrams to make descriptions in an appropriate way.

Critique of RUP:

RUP is a useful in professional environments which require large software systems. RUP is rigorous in producing documentation. That is why RUP is generally adopted by large companies in the software industry. Such developments demand many individuals skilled in software engineering and development activities. RUP provides mechanisms which help coordinate several people working on the same software project, so that they "speak the same language". In essence, well-formed documentation promotes continuity in a software based information system, safeguarding against the fact that the human talent/workforce will change over time. RUP contributes to the sustainability of software projects. In contrast, RUP could not be considered a good methodology for small developments, as RUP needs time to produce all the well-formed documentation. For those cases, Agile methodologies work better.

2.1.7. An Activity Diagram from a Decision Table

The diagram that follows is a decision table, showing potential solutions to exceeding a client's budget in the context of an advertising agency. The table follows

three rules: If the budget is not overspent, no action is needed. If the budget is overspent by less than 2%, a letter is sent to the client. Finally, if the budget is overspent by more than 2%, a letter is sent to the client and a meeting is scheduled (Bennet, McRobb, & Farmer, 2010: 297).

With this information, an activity diagram has been produced according to the figure 2.6 below.

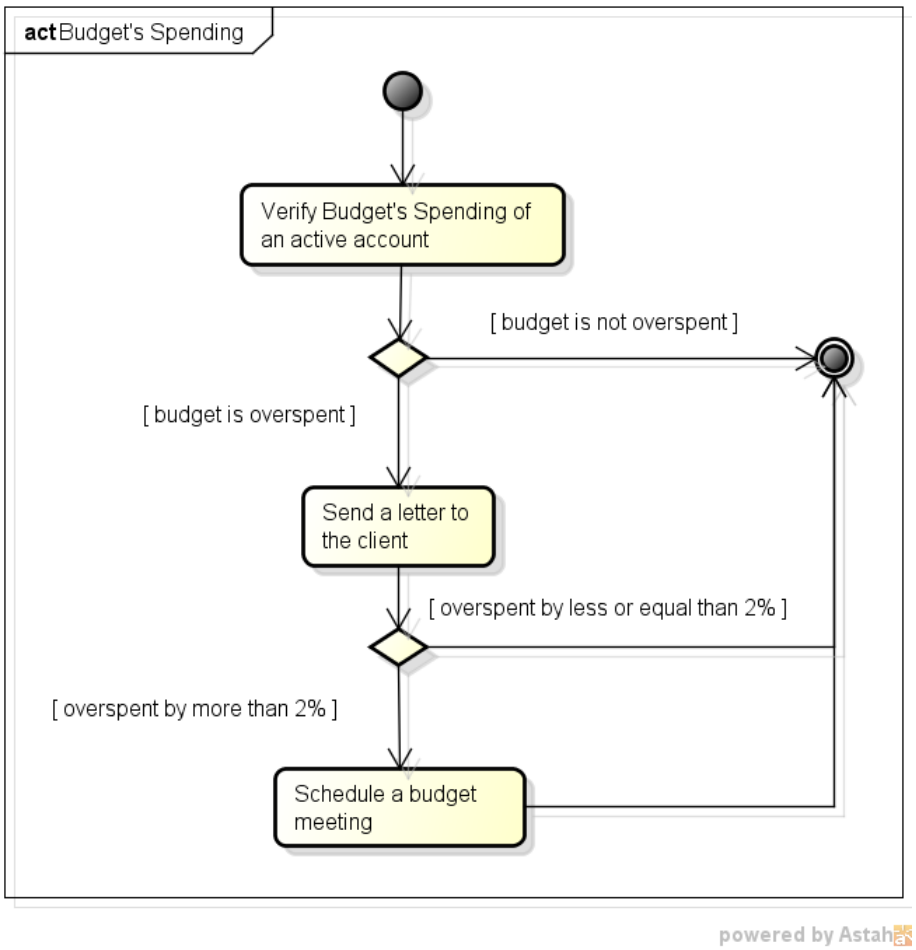


Figure 2.6 Activity Diagram on the Budget's spending for an advertising company.

2.1.8. An Architecture based on a Data Warehouse Design: The Case of a Public University

An information system is a set of interrelated components that collects, processes, stores and distributes information to support decision-making, taking into account the control and management of an organization (Laudon & Laudon, 2002: 6). In the specific case of the integrated information system at the University of Nariño in Southern Colombia, a relatively small, public university, an architecture had to be constructed based on a data warehouse design, due to the complexity of the information it managed.

The architecture contains layers which correspond to transactional processing, decision-making support, and executive support. In addition to this, the integrated information system at the University of Nariño was built to manage huge volumes of information, the layers are described below.

The transaction processing systems are systems that perform and record daily transactions and routines, necessary for running the business, in this particular case academia. These systems support the operational level of the organization and data are what matters most. Then, the decision-making support systems are implemented at the information management level (tactical level) of the university; such systems combine data and advanced analytical models through data analysis tools in order to support decision-making. Finally, the executive support systems serve the strategic level of the university; these graphic tools provide judgment, evaluation and understanding of the university context. In addition, these systems filter, compress and track critical data, with

emphasis on reducing the time and effort required to obtain useful information for managers (Insuasti et al, 2008:15). All these layers are illustrated in figure 2.7.

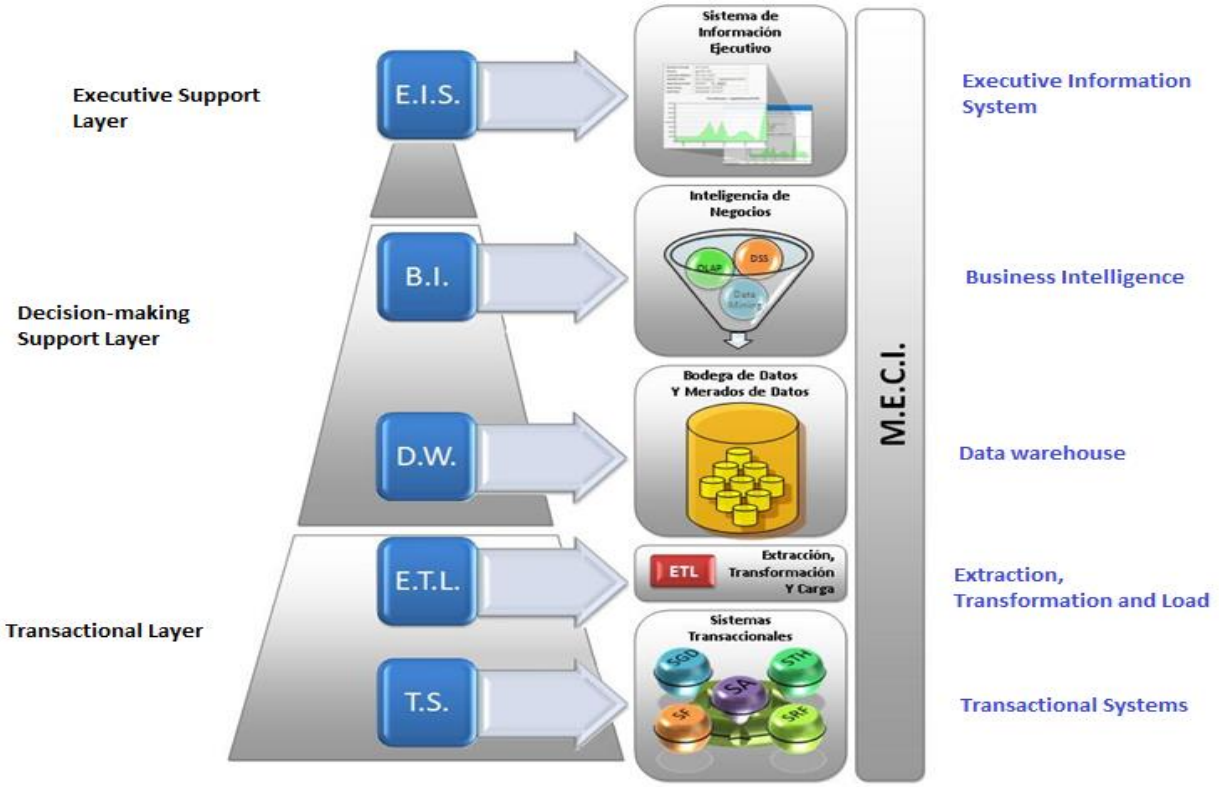


Figure 2.7 Integrated Information System Architecture at the University of Nariño (Insuasti et al, 2008: 18).

It is important to note that the architecture has a transversal entity called M.E.C.I. the Spanish acronym for “Modelo Estandar de Control Interno” (Internal Control Standard Model) which is a national model to control public institutions. It was established by the Colombian government, for use in public education. In accordance with the architecture diagram, M.E.C.I. must be present at every single stage in the entire system in response to government requirements.

The features derived from the implementation of this architecture include: the usage of a waterfall software process, extensive UML-based documentation, and internal control policies applied to each subsystem. Many people were involved in the construction of these integrated systems. It was possible to interact with large teams which is the main benefit; on the other hand, a negative might be considered to be the lengthy development period. In fact, there are still some modules under construction.

2.1.9. Analysis Model for the Scheduled Maintenance Sub-System: The Case of an Airline Information Management System (AIMS)

According to the previously analyzed requirements, the following use case diagram represents a global perspective of the scheduled maintenance sub-system.

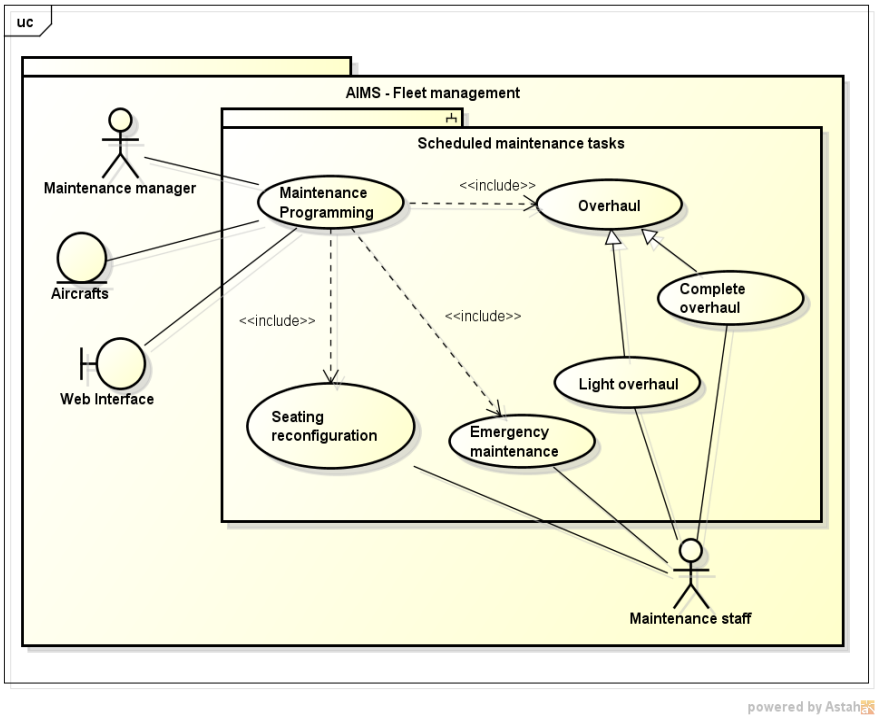


Figure 2.8 Scheduled maintenance sub-system expressed in a use case diagram.

Analysis Model

The scheduled maintenance sub-system is created using a partial analysis model, which includes a class diagram for the whole sub-system, a sequence diagram, and a set of state machines as a common specification (Bennet, McRobb & Farmer, 2010: 181). The first structure is shown as a class diagram in which the external (the individuals involved) and internal classes (some of them entities and others boundaries) are integrated. It is important to note that this proposed analysis model for the scheduled maintenance sub-system depends on the other analysis models for the rest of the sub-systems in the AIMS. The feedback in the group section on the learning platform helped the construction of this partial analysis model.

The first stage is shown as a class diagram, see figure 2.9.

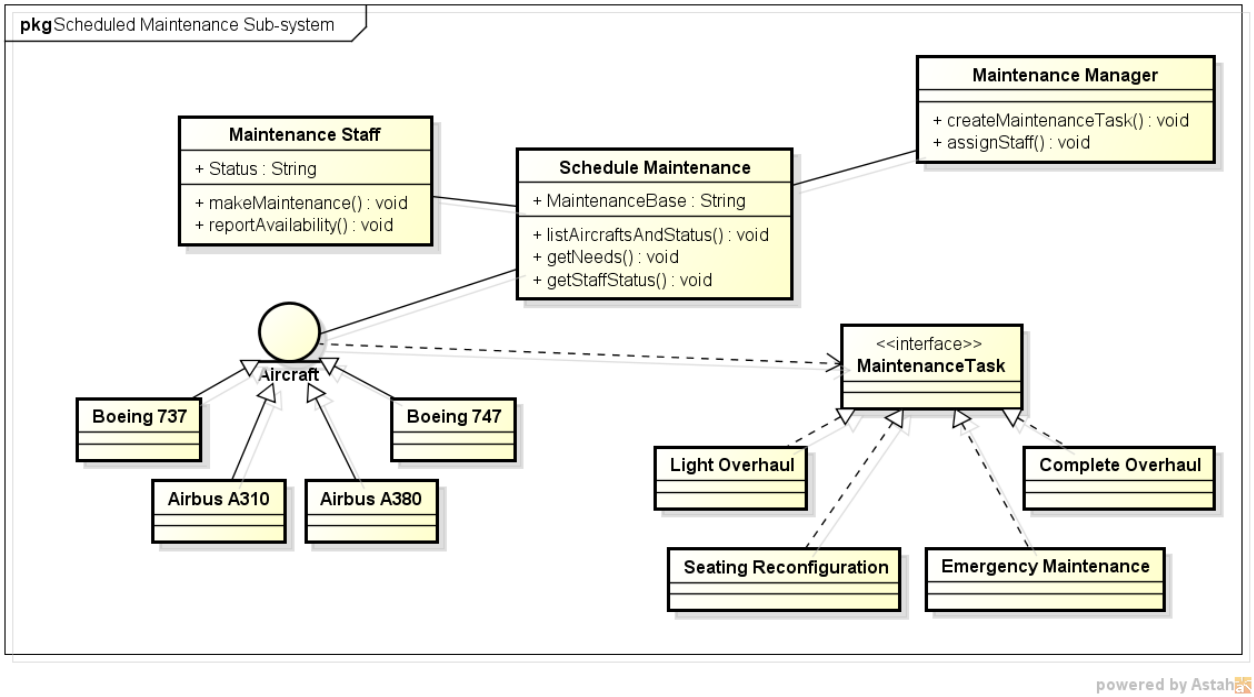


Figure 2.9 Scheduled maintenance sub-system's class diagram

Following the above, figure 2.10 represents the sequence diagram when interaction between previous classes is established.

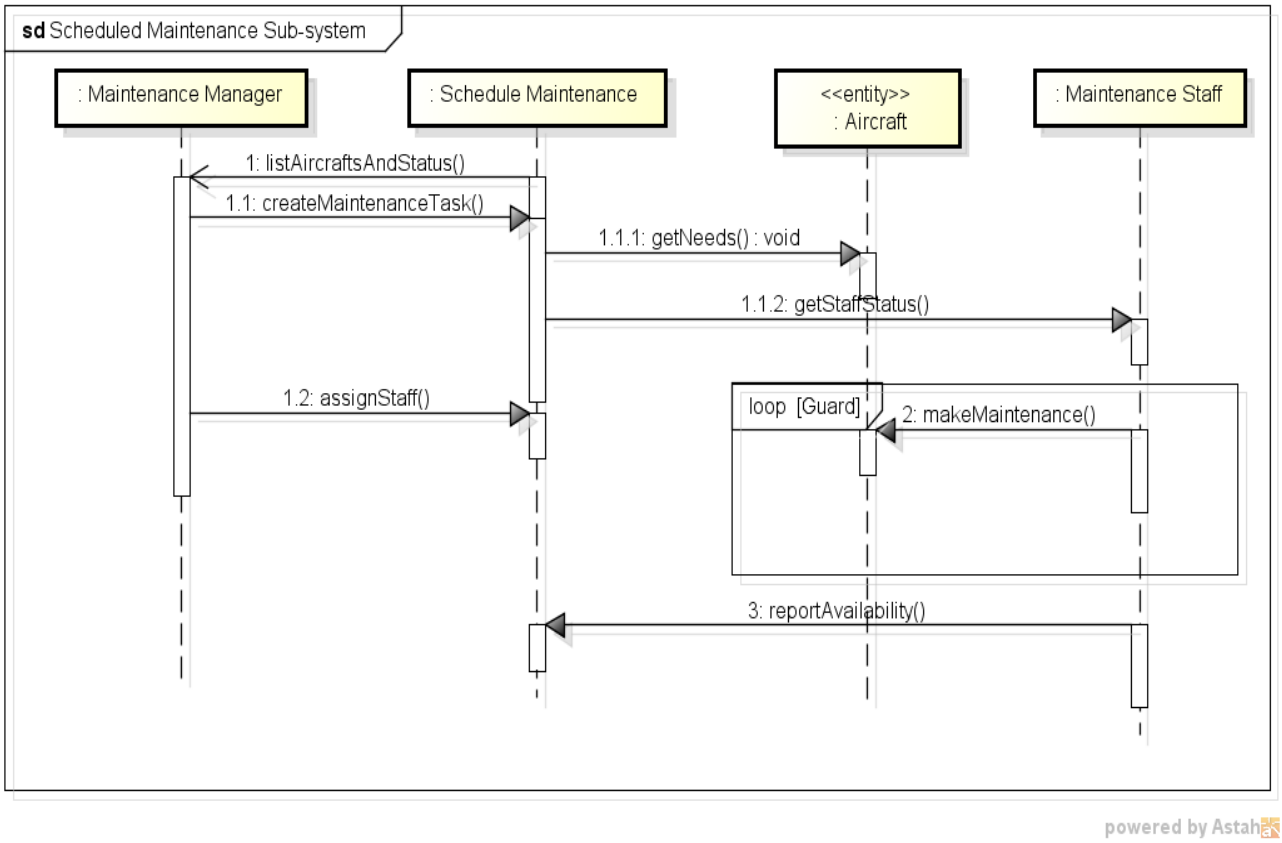


Figure 2.10 Scheduled maintenance sub-system's sequence diagram

Finally, these are the state machines that define the behavior of some classes where their status is changed according to variable circumstances. The state machines are shown in figure 2.11 and 2.12.

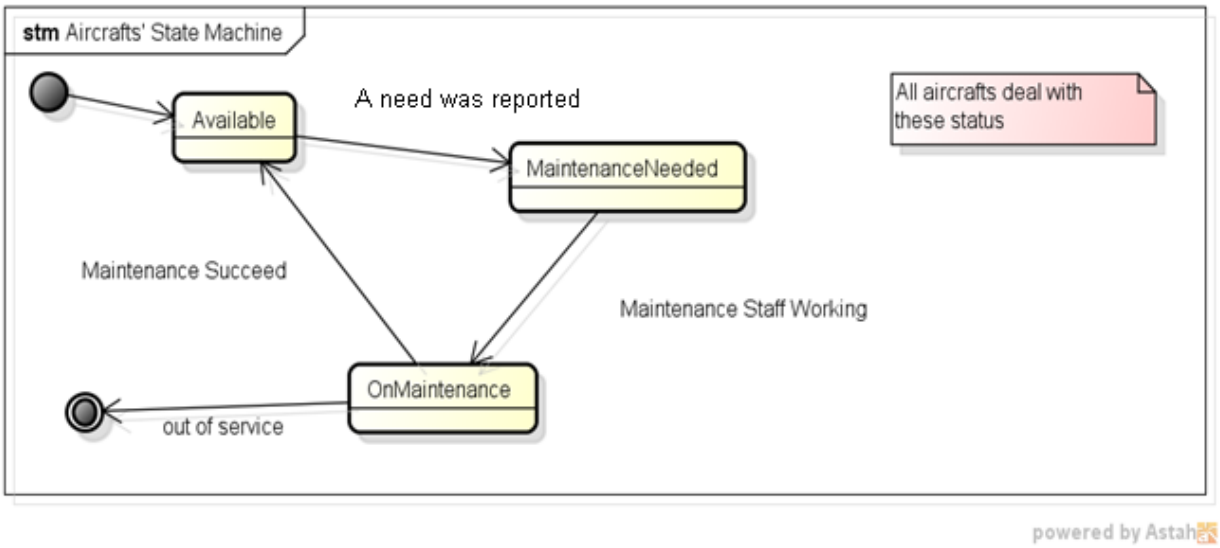


Figure 2.11 Aircrafts' State Machine

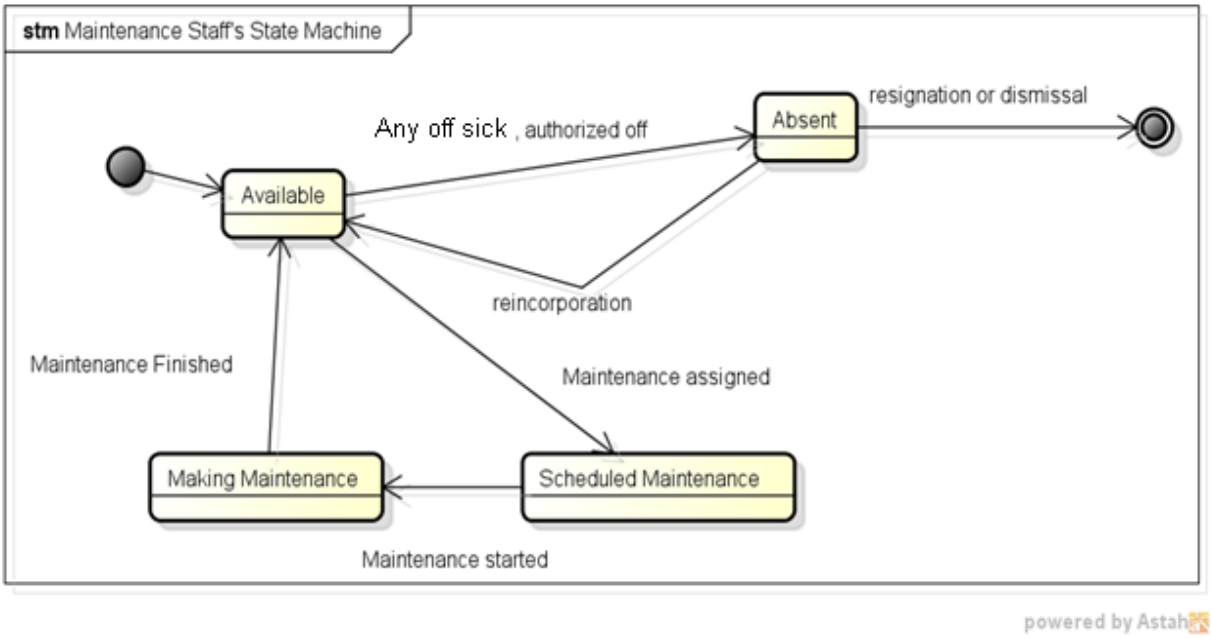


Figure 2.12 Maintenance Staff's State Machine

2.1.10. A Contrast Between a "Top-down" Approach and TDD

In recent years, an important concept and practice in software development concerning quality criteria has emerged, Test-Driven Development, or TDD. TDD belongs to the field of Agile methodologies. Its origins, the primary idea of testing each software component as they are created, are closely related to the eXtreme Programming practices of 1999/2000 (Bender & McWherter, 2011: 8).

Thus, TDD became the accepted design methodology in which end users take part in the software construction process. TDD is a development practice in which developers write tests for each programmed component. This is a common practice in TDD, known as “test first development” was widely used for satisfying business requirements in an organized way. By writing test units, TDD looks to improve a software product’s quality. Some of the benefits of using TDD include: quality criteria on coding from the start, high levels of fidelity, creation of more focused libraries and APIs, and promotion of good communication.

The classic “top-down” approach is also widely applied in software engineering. There are common phases or stages in all software processes, regardless of the methodology employed in its construction. These phases are: requirement analysis, design, implementation, and deployment, among others, according to a project lifecycle (Bennet, McRobb & Farmer, 2010: 70). Looking at this general overview of software construction, there is an implied need to deal with complex situations which can be solved using a “top-down” approach, including the general point-of-view (the architectural design) and the specific field (detailed design).

While detailed design extensively describes the classes, attributes, operation signatures, and their associated data types (Bennet, McRobb & Farmer, 2010: 395); TDD methodology is characterized by programmers writing tests in advance related pieces of production code. This approach is profoundly related to quality criteria in the professional production of software (Madeyski, 2010: 2).

It could be advisable to integrate TDD principles into the “top-down” approach in order to promote good practices in terms of quality assurance from the classic perspective. The “top-down” approach is popular in environments where software construction matters most. However, this integration might be possible if the “top-down” approach comes from an Agile methodology.

2.1.11. Browsing Concept Notes (A Case Study)

A case study about campaign management is herein put forward as a proposal. In the example subsystem, “Advert Preparation”, two individual use cases, called Create Concept Note and Browse Concept Notes are integrated, see Figure 2.13. The analysis which follows focuses on the second use case: Browse Concept Notes.

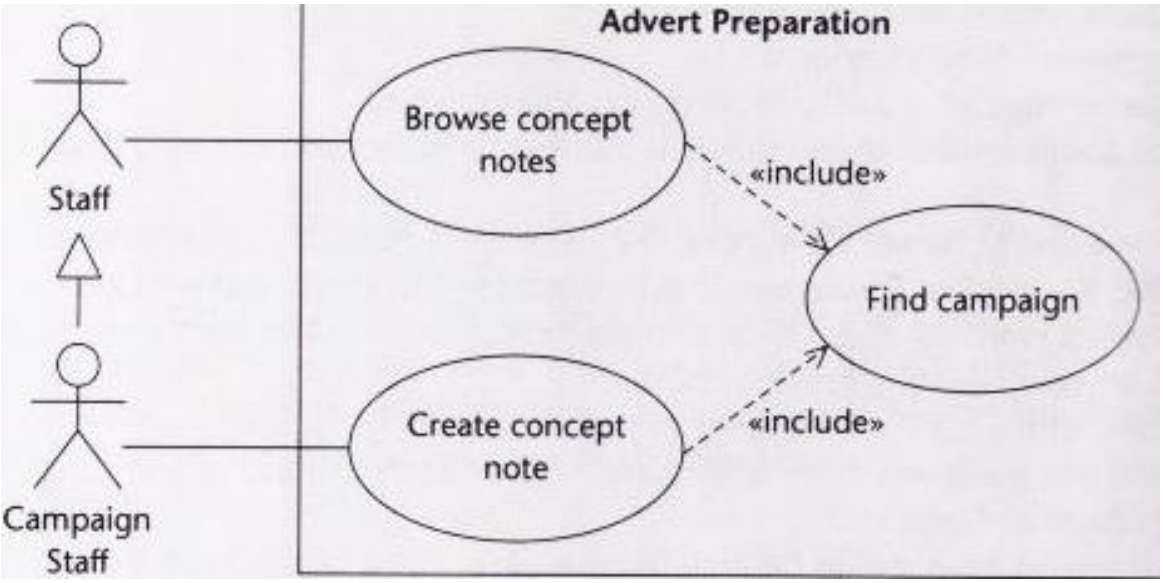


Figure 2.13 Use cases for “Advert Preparation” Subsystem (Bennet, McRobb & Farmer, 2010: 166).

Browse Concept Notes described in the template (Thomas, 2012)

Use Case Title	*Browse concept notes*		
Version	1.0	**Date**	October 13rd, 2012
Package	*AdvertPreparation*		
Summary	*This use case allows Campaign Staff to view concept notes for campaigns from a digital repository (perhaps, a database). This is a read-only function of the Advert Preparation subsystem.*		
Primary Actor	*Campaign Staff*		
Secondary Actors	*Staff*		
Inherits	*There is a generalized relationship between Campaign Staff and Staff. The former inherits features from the latter.*		
Includes	*This use case includes the “Find Campaign” use case*		
Extension Points	N/A		
Business Rules	*Staff may consult concept notes; however, they are read-only. New concept notes can be added without altering the existing ones.*		
Pre-condition(s)	*It is necessary to provide the title, or the creator’s name, or the creation date/time in order to find concept notes that match the criteria.*		

Table 2.3 Template for browsing

Typical Sequence of Events

Actor Stimulus		System Response
1. Select Search Criteria (by title, creator, date, time)		2. Activate the filters according to the search criteria
3. Providing strings		4. Do the search with the strings
5. Browse the results (if they are available)		6. Link the result list with the digital recording (files) in order to create a preview
7. Request a specific concept note		8. Retrieve the requested file to the client (the actor who requested it)

Table 2.4 Typical Sequence of Events

Post-condition(s)	*Add some log records about the search process*

Table 2.5 Pre-conditions

Priority	*Highest*		
Outstanding Issues			
Author	*Jesus Insuasti*		
Business Owner	*?*		
Notes	*The user interface will be highly portable in order to allow it to function on practically in all platforms/devices. Web-based interfaces are suggested.*		
Version History	*N/A*		
Use Case Title	*Browse concept notes*		
Version	1.0	**Date**	October 13rd, 2012

Table 2.6 Alternative Sequences of Events

Alternative 1: No data found at step 1

Actor Stimulus		System Response
Staff do not select any criteria		A message is shown recommending the selection of one criterion at least.
Staff do not provide any string		A message is shown recommending the insertion of one string at least.
Staff provides strings that do not match (in joint) with any result.		A message is shown recommending the usage of alternative strings.

Table 2.7 Alternative 1

Alternative 2: Too many data at step 1

Actor Stimulus		System Response
Staff provide a very common string (e.g. an article: the, a, etc.)		A message is shown recommending the usage of more string in order to filter the query better.

Table 2.8 Alternative 2

Alternative 3: File not found at step 2

Actor Stimulus		System Response
Staff get the result list according to the query, but digital files are not available		A message is shown recommending contact with the system manager to fix this problem

Table 2.9 Alternative 3

Alternative 4: Add a new Campaign Note at step 2

Actor Stimulus		System Response
After searching, the Staff want to add a new campaign note through clicking the “add new” button.		The system shall redirect to the “Create Concept Note”, use case.

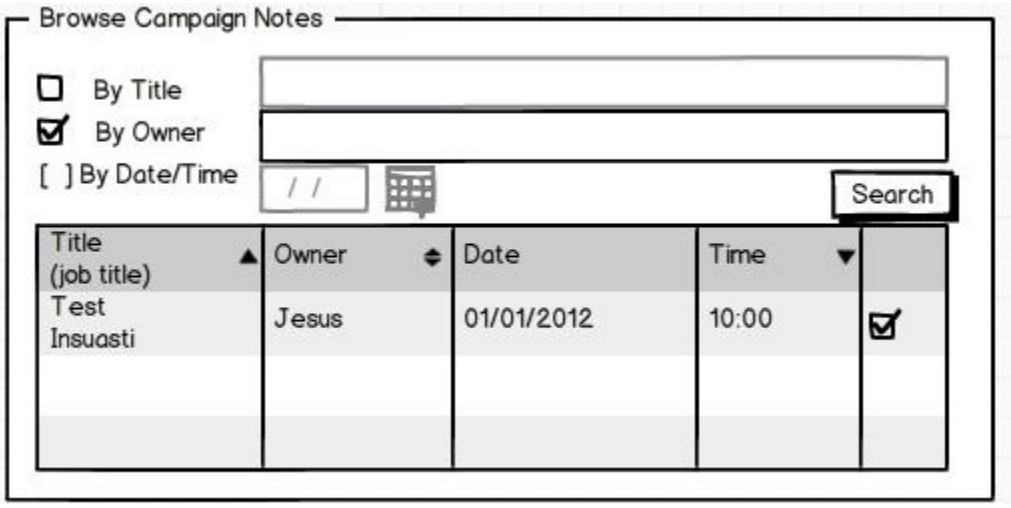
Table 2.10 Alternative 4

Use Case Title	*Browse concept notes*		
Version	1.0	**Date**	October 13rd, 2012

Table 2.11 Use case

Screen or Report Mock-ups

Screen or Report Title	*Browse concept notes*
Interface Reference	*Base Interface (No messages are included)*



Data Elements or Attributes Used in this Screen	
Check Boxes	Selection criteria
Text Boxes	Input Strings
Data Grid	Result List

Table 2.12 Report mockup

2.1.12. Design Quality Metrics

At the design stage of the software building process, all the input from previously carried out analysis constitutes the foundation for establishing how the software system should work; and more specifically, how the software system must carry out its tasks. Design therefore, defines software's inner functionality, how a system carries out operations in order to solve a specific problem (Bennet, McRobb, & Farmer, 2010: 348).

In professional contexts, software must be equipped with parameters that ensure high quality and the design stage of the process is essential in establishing quality criteria. In terms of metrics, there are several concepts related to quality criteria, including: cohesion, coupling, completeness and primitiveness, among others. It is important to note that cohesion is a common denominator in quality metrics studies. In fact, the level of cohesion in terms of metrics is based on the degree to which members of a class are interrelated; from an implementation perspective, measurements are interrelated depending on to what degree their methods and attributes share parameters when they are in action (Kaur & Singh, 2012: 66).

Other quality variables however, must be taken into account; in software construction processes for example, there is the Goal-Question-Metric Paradigm. This relatively new proposal is based on the common metrics used in the realm of professional software development. This approach seeks to improve modeling and designing practices for software systems stemming from the object identification process (Rajput & Singh, 2011: 345). As such, the Goal-Question-Metric Paradigm integrates all the metrics, starting with those that are convenient and easy to measure. This identification process

is done following three main steps: list the major goals, define the main question that must be answered for each individual goal, and finally decide what needs to be measured in order to answer these questions properly (Rajput & Singh, 2011: 346).

The Goal-Question-Metric Paradigm was designed for highly competitive business environments or organizations, including insurance companies and banks. It therefore stands to reason that an ATM system would benefit from this approach, in terms of improving the quality of its software construction processes. The design stage for the construction of an ATM system could use the Goal-Question-Metric Approach to define the following scheme shown in figure 2.14.

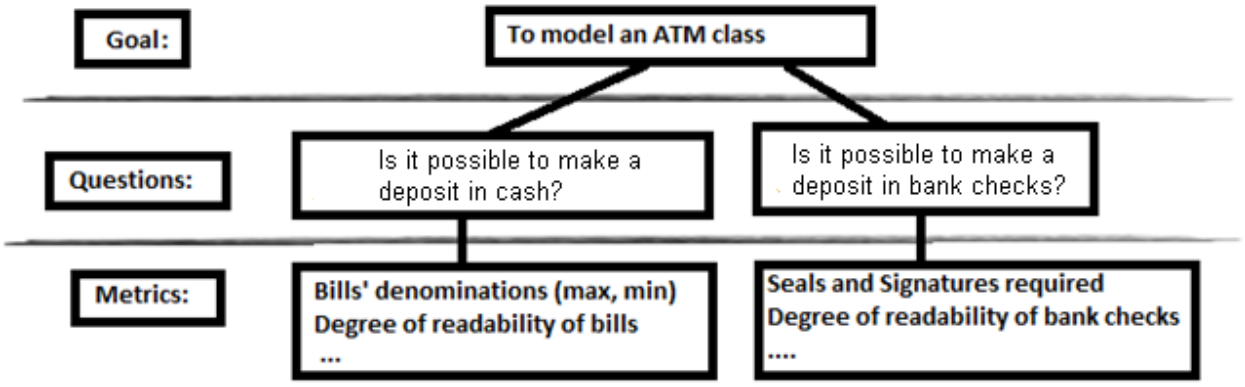


Figure 2.14 The Goal-Question-Metrics approach defining deposit tasks in an ATM system.

The incorporation of a “deposit” function in the ATM system also effects the definitions of use case diagrams and interaction diagrams. The key point here is the articulation of metrics at the point of making the design model to improve quality. The particular advantage of using the Goal-Question-Metric Approach is that all kinds of metrics can be integrated into a complex software design.

2.1.13. Issues when Adopting a New Software Process

One of the widely held assumptions about software engineering is that the costs of changing a program increase exponentially over time (Bohem & Papaccio, 1988). Traditional software methodologies are in fact based on this assumption. The idea revolves around changing the software itself; however, what is the impact of changing to an entirely new software process? This paper explores four potential issues when changing to a new software process.

1st Issue: Required knowledge for a brand new software process. As software development is a carried out by individuals, people can only build new software based on their own prior knowledge. It is important to note that in undertaking to install a new software process, those involved must be sufficiently qualified to ensure results. Professional knowledge concerning the usage of a new process is a key factor in successfully adopting it.

2nd Issue: Reuse of legacy software. In software construction, corporations have their own sets of legacy software, built using traditional software processes. In this vein, the adoption of a new software process demands the migration of software, as it is advisable to reuse as much as possible. Otherwise, the development team would have to recreate the new software and all its components from scratch. This would be incredibly time consuming and therefore represent a loss, affecting software production.

3rd Issue: Required technological platform. According to the expectations and requirements of today's information based society, new software applications are required to work on several platforms. Computing pervades our daily lives, in different forms of hardware such as personal computers, laptops, tablets, PDAs and smart phones, among others. This necessitates the use of cutting-edge technology in terms of development platforms and languages. Thus, the necessity of upgrading development platforms must be taken into account, which represents cost, time and required training.

4th Issue: New policies for software support. New policies in software support might be needed to face the software construction tasks. This could require extra work from the development team when software support is needed in accordance with the changes in software platforms. In addition, it is possible that customers will need changes in software support from the development team due to the changes in software implementations.

Obviously, these situations represent an investment in time and money to any software development company. Perhaps the best way to minimize the impact of changing a software process is for the process to be based on a well-defined transitional plan previously created by the company's head alongside the development team. Such a transitional plan would have to cover staff training and therefore the transition would have to have a budget reserved for that purpose.

2.1.14. Reviewing an ATM case

In an example using an ATM, a state machine was created in order to understand the additional functionality in terms of the working response for an invalid PIN entry, or the use of an unknown card. The state machine can be seen in figure 2.15 below.

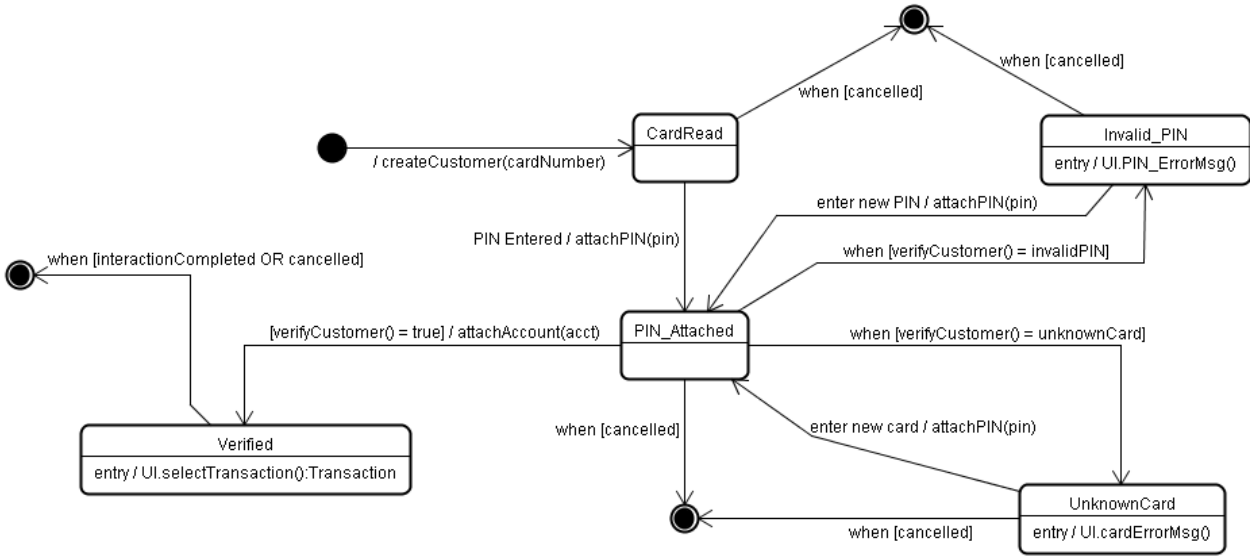


Figure 2.15 Proposed state machine for an ATM case (Laureate Online Education, n.d.: 16).

The state machine above shows two states which have two possible ways out –try a new entry or quit– and those states are *Invalid_PIN* and *UnknownCard*. The incorporation of those states implies changes in the interaction between the classes however, there is no obvious implication of this in the general structure of the class diagram. As such, new interaction diagrams were created and can be seen in figures 2.16 and 2.17. The new additional operations in the classes are enough to deal with the new

situations; so there was no need to create a new class diagram. In other words, it was only necessary to show a change in behavior, rather than a change in structure.

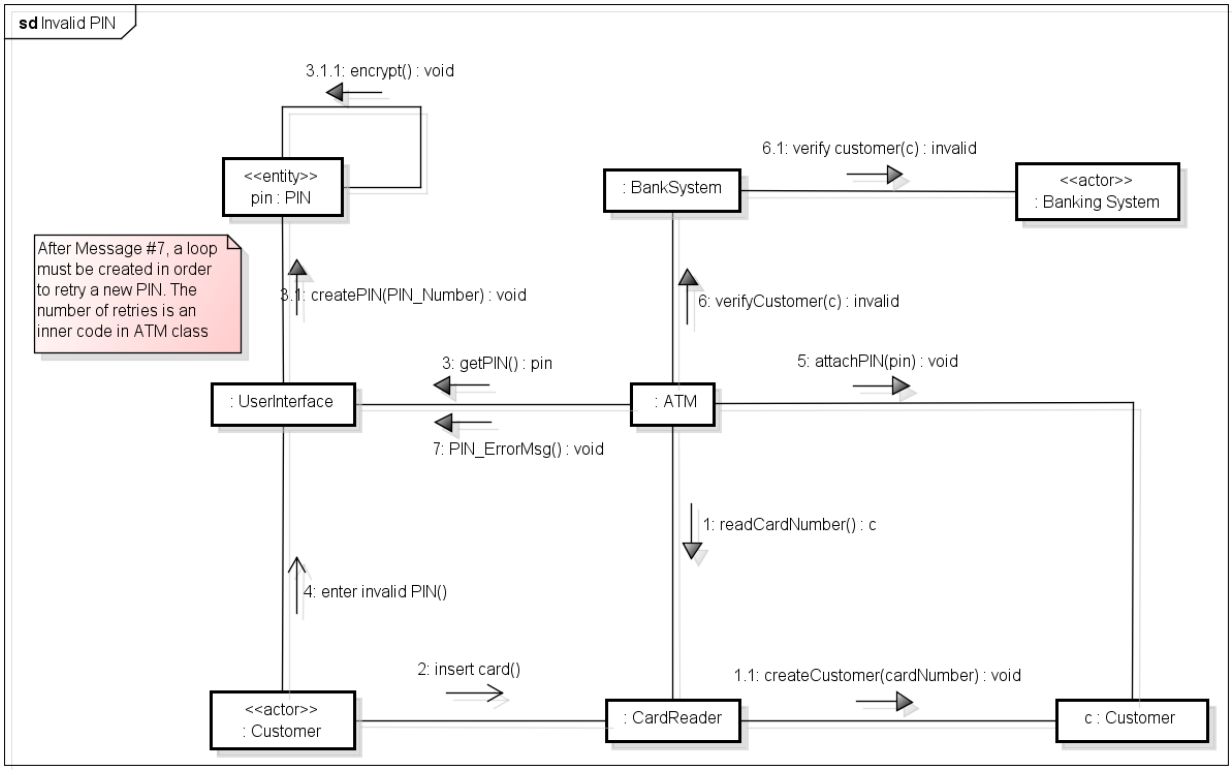


Figure 2.16 The modified *InvalidPIN* interaction diagram.

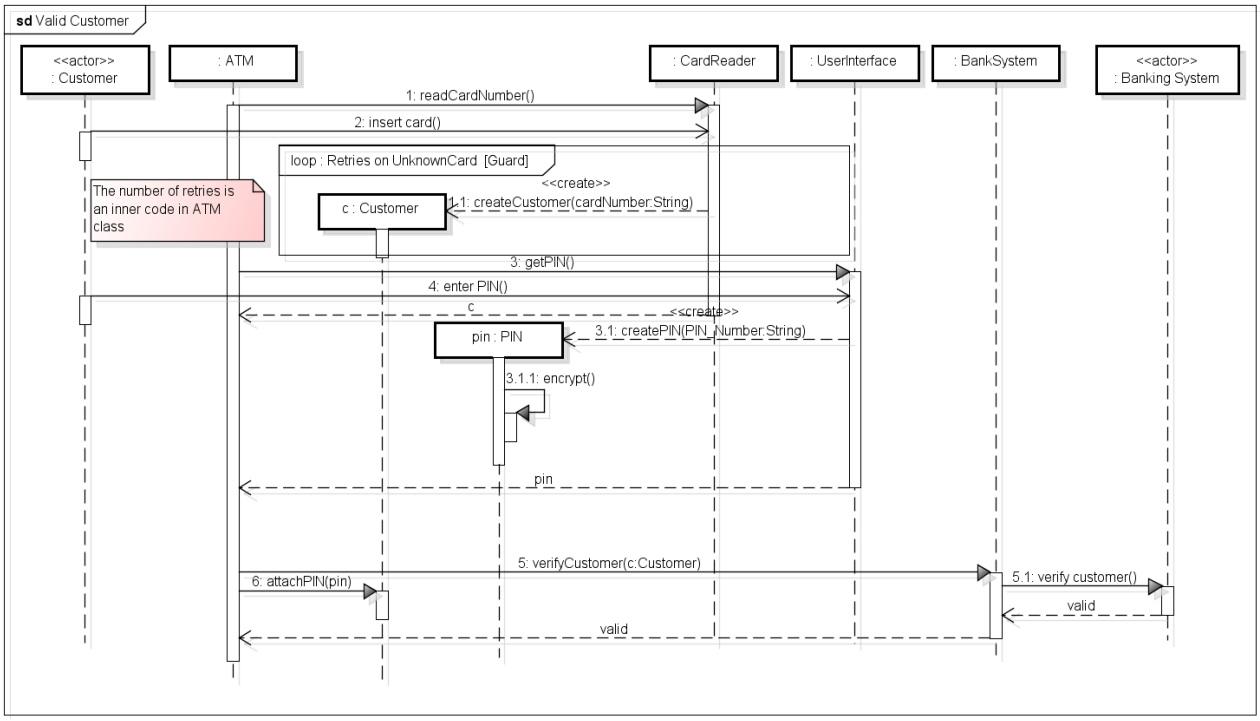


Figure 2.17 The modified *ValidCustomer* interaction diagram.

The actions associated with the new behavior in both cases are based on the "retry" option. This means that the state machine shows a scenario in which a new input is possible. Indeed, in most bank systems the number of tries is a small number (i.e. 3, 4, or 5). After that, the bank system changes the account's status from *active* to *blocked*.

The state machine allows for the possibility of retries in the user interface of the ATM. As previously stated, such changes imply behavioral adaptations but not necessarily in the definition of the structure, however the structure in classes still works. In conclusion, the alternative solution is based on the same classes' structure with a small addition, the retry option, in both *ValidCustomer* and *InvalidPIN* diagrams.

2.1.15. Scheduled Maintenance Tasks: A Use Case Description

The following use case diagram shows the functionality of a part of an airline system in terms of scheduled maintenance tasks. This use case diagram represents the interaction between actors and the software-based system (Bennt, McRobb, & Farmer, 2010: 154). Regardless of the specification of a given problem, there are 4 maintenance categories: seating reconfiguration, light overhaul, complete overhaul, and emergency maintenance. When maintenance tasks are programmed, they are allocated in three maintenance bases: Europe, Asia and North America. This is the system for aircrafts such as Boeing 737 – 747 and Airbus A310 – A380 (the first number represents short-haul flights and the second long-haul flights).

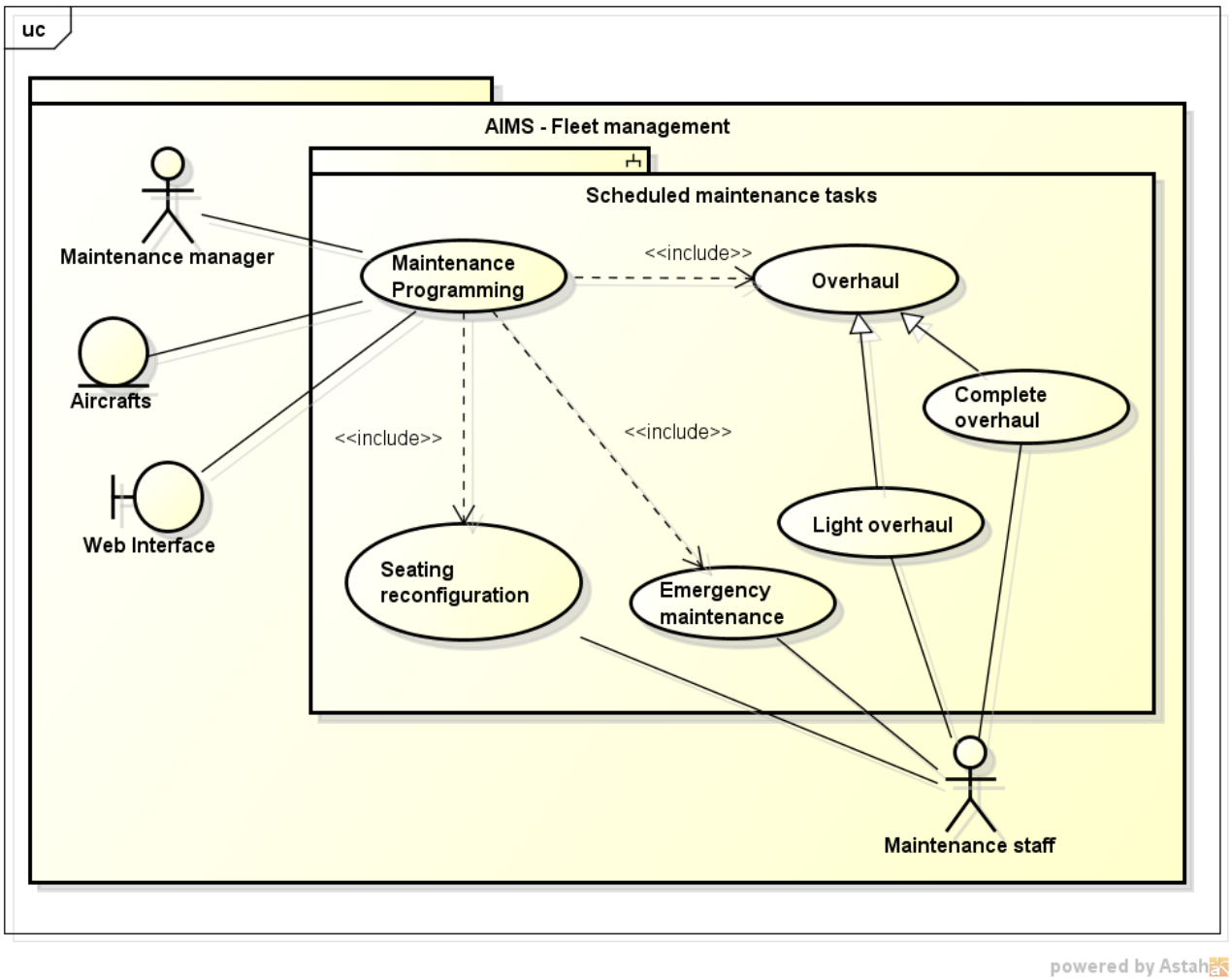


Figure 2.18 General use case

According to the above diagram, a maintenance manager is (evidently) in charge of programming the aircrafts' maintenance. The aircrafts are entities in the system which require scheduled maintenance tasks. The web interface is a boundary class where customers outside can check the status of seating reconfiguration. Finally, the maintenance staff work on technical tasks concerning seating reconfiguration, emergency maintenance, and light/complete overhaul. The general description of the use cases is provided as follows.

Use Case Title	*Maintenance Programming*		
Version	1.0	**Date**	October 24th, 2012
Package	*ScheduledMaintenanceTasks*		
Summary	*The core of the ScheduledMaintenanceTasks package is this use case. It manages schedules as timelines in each aircraft's "life-story". Suppliers are part of this use case. In addition, this use case provides information on external queries through a web interface. This use case always checks the current status of the aircrafts.*		
Primary Actor	*Maintenance Manager*		
Secondary Actors	*Aircrafts, Web Interface, and Maintenance Staff*		
Inherits	*N/A*		
Includes	*Seating reconfiguration, Emergency maintenance, and Light/Complete Overhaul*		

Table 2.13 Use case 1

Use Case Title	*Seating reconfiguration*		
Version	1.0	**Date**	October 24th, 2012
Package	*ScheduledMaintenanceTasks*		
Summary	*This use case allows the reconfiguration of the seats within aircrafts according to specific requirements and customers' expectations, including the plans about flight types.*		
Primary Actor	*Maintenance Staff*		
Secondary Actors	*Aircrafts, and Maintenance Manager*		
Inherits	*N/A*		
Includes	*N/A*		

Table 2.14 Use case 2

Use Case Title	*Emergency Maintenance*		
Version	1.0	**Date**	October 24th, 2012
Package	*ScheduledMaintenanceTasks*		
Summary	*This is a high priority use case which follows the current status of the aircrafts. Its inner work must be done as soon as possible*		
Primary Actor	*Maintenance Staff*		
Secondary Actors	*Aircrafts, and Maintenance Manager*		
Inherits	*N/A*		
Includes	*N/A*		

Table 2.15 Use case 3

Use Case Title	*Overhaul*		
Version	1.0	**Date**	October 24th, 2012
Package	*ScheduledMaintenanceTasks*		
Summary	*This use case is based on a detailed review of the parts of the aircraft. Depending on what type of overhaul is required, the time spent might change. This is a generic use case, and the maintenance staff will perform their duty according to the current need.*		
Primary Actor	*Maintenance Staff*		
Secondary Actors	*Aircrafts, and Maintenance Manager*		
Inherits	*N/A*		
Includes	*N/A*		

Table 2.16 Use case 4

2.1.16. Software Reuse

Software reuse integrates two threads: development for reuse and development with reuse. Developing for reuse is the development of reusable assets and the definition of the activities involved. Assets produced in development for reuse are often cataloged and stored in a library. Development with reuse is how to use assets in building applications, and defines mechanisms for search and retrieval by team members (Jarzabek, 2007).

Research addressing the systematic reuse of software (Sommerville, 2007) outlines that the key benefits for organizations that incorporate reuse are improved quality and reduced effort, which result in the increased productivity of the development team. When it comes to standardization, one of the standards developed by IEEE Computer Society, which was coded 1517, details all the relevant aspects to be considered in the development of a software life cycle from a reuse-oriented approach (IEEE Computer Society, 2010).

Although the basis of software reuse is universal, there are several approaches to reuse, such as class libraries, reusable components, application frameworks, patterns, and service oriented architecture, among others. In a real life context, such as a public university, the authors' experience in terms of software reuse within a formal software development process is mainly based on the CBSD –Component Based Software Development- process.

Component-based software development –CBSD– is designed to produce scalable software systems by reusing pre-built components. As such, the purpose behind CBSD can be summarized as group of concepts, mainly the concept of component. A component is a reusable unit of deployment and composition. A common view is that one component is closely associated with an object, and that the CBSD is therefore an extension of object oriented development (Crnkovic, et al, 2002). According to Sametinger, the phases of the CBSD's life cycle involve aspects of reuse at each transition. Figure 2.19 shows how such transitions re-take predesigned assets, in the case of software components.

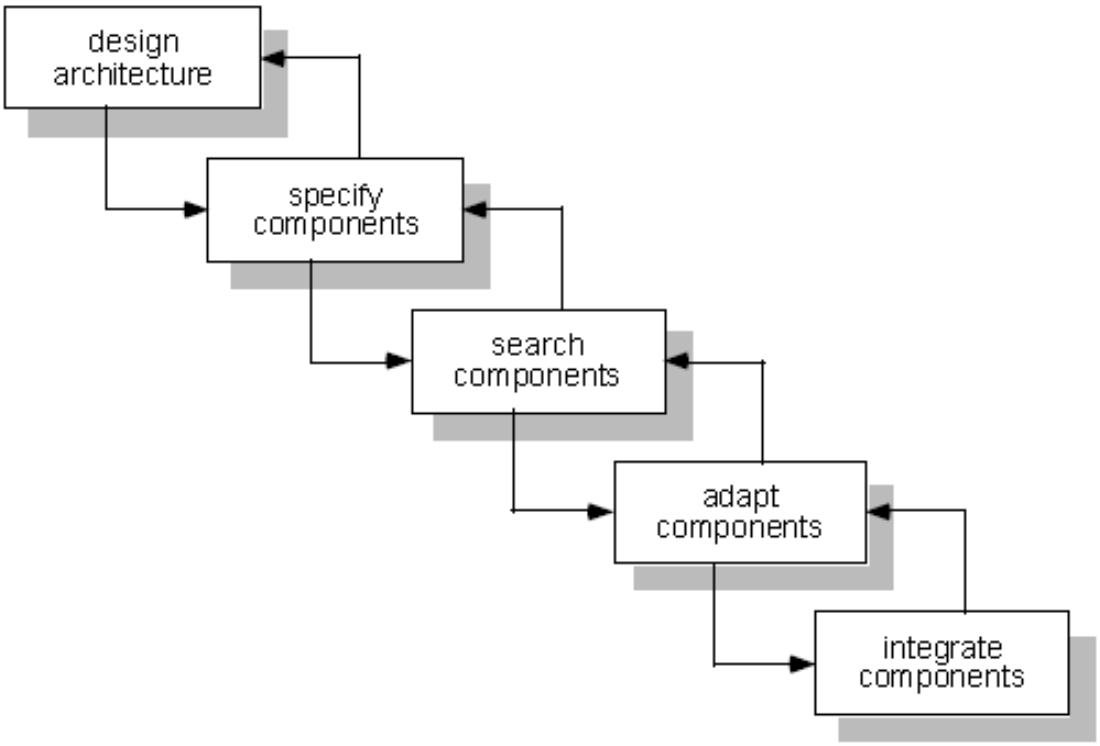


Figure 2.19 CBSD's life cycle (Sametinger, 2010)

In conclusion, as previously stated CBSD was used by the authors in a small public university in Southern Colombia. The experience demonstrated with what ease the

proposal can be adopted by using components. Staff training was necessary in order to face the challenges surrounding the construction of complex software systems for the university. In general terms, the level of acceptance was adequate. Software reuse based on a strong, well-defined approach such as CBSD, was therefore successful in this case.

2.1.17. The use of CRC Cards

Regardless of the methodology employed, requirement analysis is a mandatory phase in software system construction; in one way or another, this phase determines the software system's function. There are different techniques used to identify classes and their relationships; if the analysis is based on an object-oriented approach, one technique that can be used is CRC (Class-Responsibility-Collaboration) cards.

CRC cards establish a framework in which responsibilities are assigned to classes, taking into account the role that those classes play as they interact with one another. CRC cards work in a system in which the main focus is the interaction between the disparate elements, and not their individual functionality (Bennet, McRobb, & Farmer, 2010: 215). In spite of CRC cards not being officially part of UML, they are widely used to explain some software system functionalities from a Responsibility-Driven Design perspective. In some cases, CRC cards are part of complex UML-based documentation in the requirement analysis phase (Briton & Doake, 2005: 149). In other words, CRC cards are defined as tables, with a header as the class name, and two columns below in which are listed responsibilities and collaborations of a given class.

In some cases, mocking up the tables is an alternative to processing all the details concerning objects' interaction in a specific use case. Thus, CRC cards are a useful tool for interactions in which team members test the roles of the objects which are part of the design; this 'acting out' is a technique used in the identification of responsibilities and collaboration scenarios Wirfs-Broke, Wilkerson, & Wiener, 1990). This is carried out when the problem to be solved is thoroughly-understood. Experience of the problem in question when using CRC cards is of paramount importance. In '*CRC Cards for Product Modelling'*, the authors emphasize the need for a profound knowledge of the system domain in order to 'play' roles in creating CRC cards. (Hvam, Riis, & Hansen, 2003: 64).

This 'acting out' technique is only feasible in teams with a high level of empathy. A team with strong communicative skills is able create the right "atmosphere" needed to 'rehearse' scenarios. However, this does not necessarily mean the team has to be in one geographical location, distances could be overcome using ICT-based solutions. 'Acting out' scenarios and role play using CRC cards' descriptions is possible regardless of the physical presence of team members.

2.1.18. XP

The basis for this chapter was an abstract written by Azhar Muhamad Arifin concerning eXtreme Programming (XP). This abstract outlines a rapid software development technique considered to be one the most popular Agile methodologies. In addition, XP promotes close relationships between customers and the development team.

In fact, customers are practically part of the development team, due to the creation of story cards which are used to understand an organization's inner functionality.

This constant feedback allows the creation of prototype-style applications which are introduced into real live contexts from their inception. Arifin states that XP is suitable for development teams in which the work can be done in pairs (egoless environments). In addition, the constant revision of a product allows for huge improvements in the quality of software production. On the other hand, Arifin also outlines a possible scenario that could create problems in the software process. In spite of XP's many advantages, the will and whim of the customer has a strong influence throughout the process. As such, a weakness in XP is that the development team is always affected by the time, will, and resources of the customers.

The main source of this theory is based on Ian Somerville's Software Engineering textbook. Somerville emphasizes that the use of eXtreme Programming is often most successful in cases in which the information systems on which the software is based are small, and the documentation involved is not rigorous (Sommerville, 2011). Somerville also states that XP is perhaps the best-known and most widely available Agile method today. According to its inner functionality, XP uses an "extreme" approach within its iterations; illustrated in the following a release cycle shown in Figure 2.20.

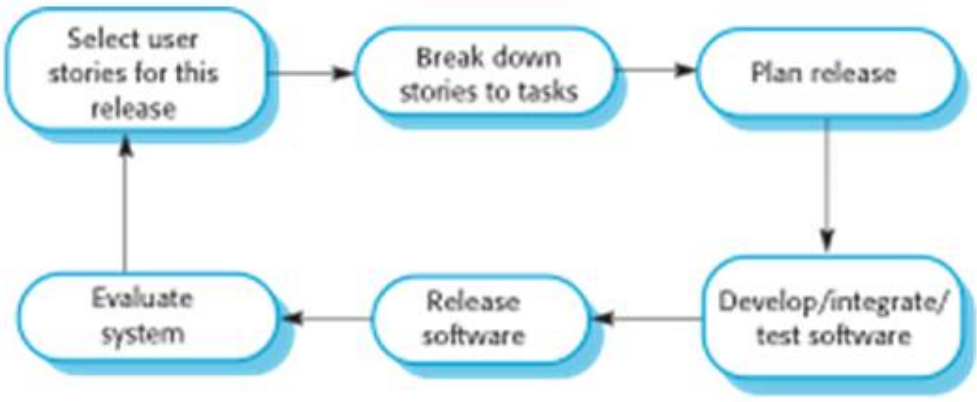


Figure 2.20 The eXtreme Programming release cycle (Sommerville, 2011).

The release cycle above is carried out as fast as possible (often in 2 weeks). With this constant interaction between customers and the development team, the use of simple designs, small releases, and pair programming tends to be most effective, to suit the incremental planning process. This close and constant communication is a key factor in the success of a software project. The following flow chart was proposed by Wells (2009) in which communication (daily if possible) is a constant feature, see Figure 2.21.

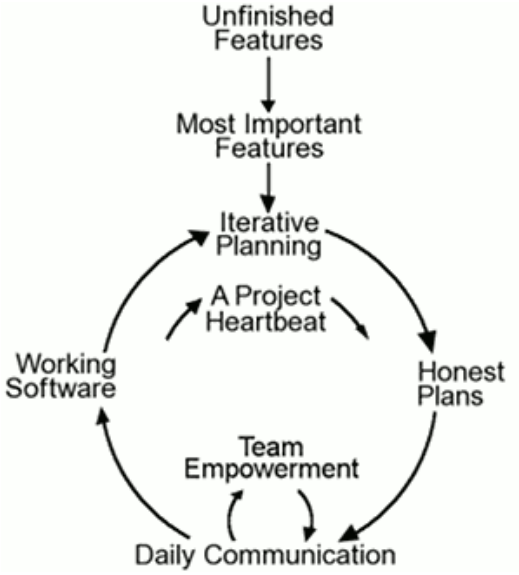


Figure 2.21 XP communication flow chart (Wells, 2009)

Critique of XP

Arifin is completely justified in his claim that XP's success entails a strong dependency on the customers' time and attitude. It is true that one of the main advantages of using XP is the fast development and testing in context. XP requires constant and active participation of the stakeholders and as such, is not to be undertaken if any of the stakeholders cannot commit.

Furthermore, not all information systems based on software are suited to being constructed using XP. Highly complex information systems that need input from lots of people need to be constructed using a different software process. However, even in XP the development team has the last word, and the head of the team has the "freedom" to decide which software process should be followed.

Contrast between XP and RUP

RUP was used by the authors in the previous DQ. Following this, two main trends in software process were identified. The first one has already been exhaustively documented, and the other has not. Both approaches are valid; it is practically impossible to say which one is better than the other. Arguably, it all depends on the context, e.g. the development team (the number of people and their skills), time, the complexity of the proposed solution, and even finance policy, among other factors.

In reality, RUP is used mostly by large enterprises in the software construction industry, whereas XP is commonly used in environments in which the software solution is small. At least, this is the situation in Colombia.

2.2. WORKSHOPS

2.2.1. Hand-In Assignment 1

1. <u>Definitions of class, object and message</u>

The Object-Oriented approach is a vision of the real world in which real objects are represented in terms of data and the related code. This approach uses abstraction to define objects through their features; objects are categorized according to their properties, behaviors and how they interact with other objects in a given environment. This approach has solved several problems associated with computer programming. The Object-Oriented approach works with several concepts; the terms *class*, *object* and *message* are defined as follows:

Class: When organizing information hierarchically, class is the defining characteristic, or characteristics, of a set of objects. Such characteristics are defined according to the common properties, methods and events that give a set of objects their identity. According to Bennet, McRobb and Farmer, a class consists of objects that share a common specification in accordance with their characteristics (2010: 93).

Object: Commonly known as an instance of a class (Bennet, McRobb & Farmer, 2010: 93), an object is the “materialization” of the characteristics, that can be properties, methods and events, among other features. In other words, an object is a sample of all possible elements that define a class. An object is present in the dimensions of space and time in computing terms; it exists while it is instanced (it occupies space in the memory and processing cycles for running its inner code).

Message: Messages are the communications between objects. Messages allow objects to interact with one another in a relational environment (Bennet, McRobb & Farmer, 2010: 102). Communication is controlled by clear message protocols, which dictate how the operations within objects are defined (signatures).

2. Message passing and encapsulation

Encapsulation is an important feature of the Object-Oriented approach, because it allows the details of operations to be hidden within objects from the beginning. It is important to note that encapsulation is defined at the same time as a class is formed, following this the objects are created according to their image and likeness, hiding the details of how the information is processed in their operations.

When it has been established how messages will be passed through valid signatures between objects, invocations are produced in order to establish communication with well-defined parameters. This is possible only if encapsulation was originally clearly defined. Thus, the concept of message passing is closely associated with the encapsulation processes, both protect against the exposure of information about the implementation (Bennet, McRobb & Farmer, 2010: 101).

3. FoodCo case study

For this case study, the example of FoodCo has been taken, and its range of products for the purposes of this study are: pickle, pre-packed vegetables, washed salads, sauces, sandwich toppings, tomatoes, peppers, courgettes, chilies, exotic vegetables,

condiments, coleslaw, and Waldorf salad. The following class diagram (Figure 2.22) shows the distribution of these products.

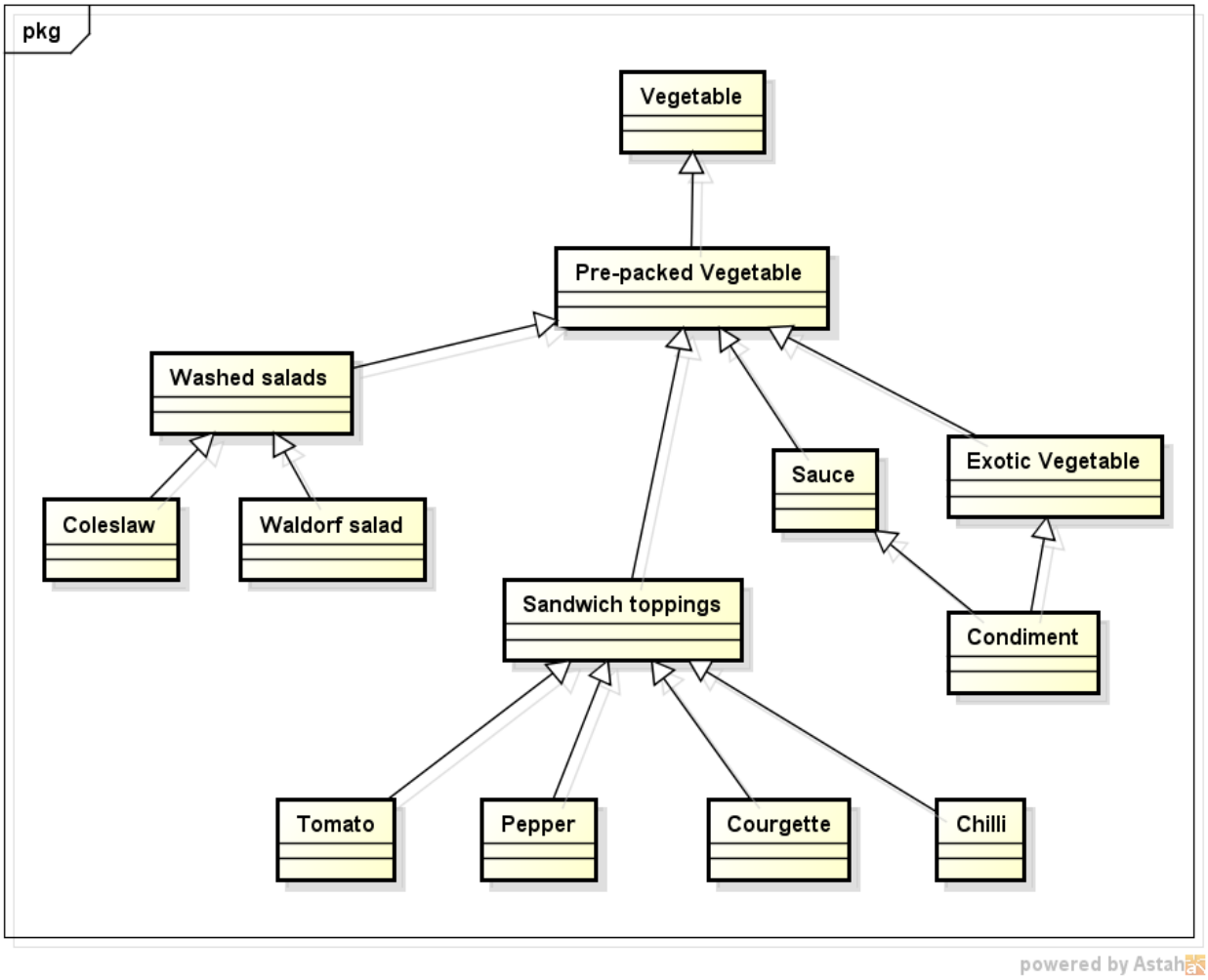


Figure 2.22 Class diagram of FoodCo's products.

The next figure includes more elements, making the diagram more complex. Figure 2.22 includes the Waldorf salad's ingredients: red apples, celery and mayonnaise; the last is considered a sauce.

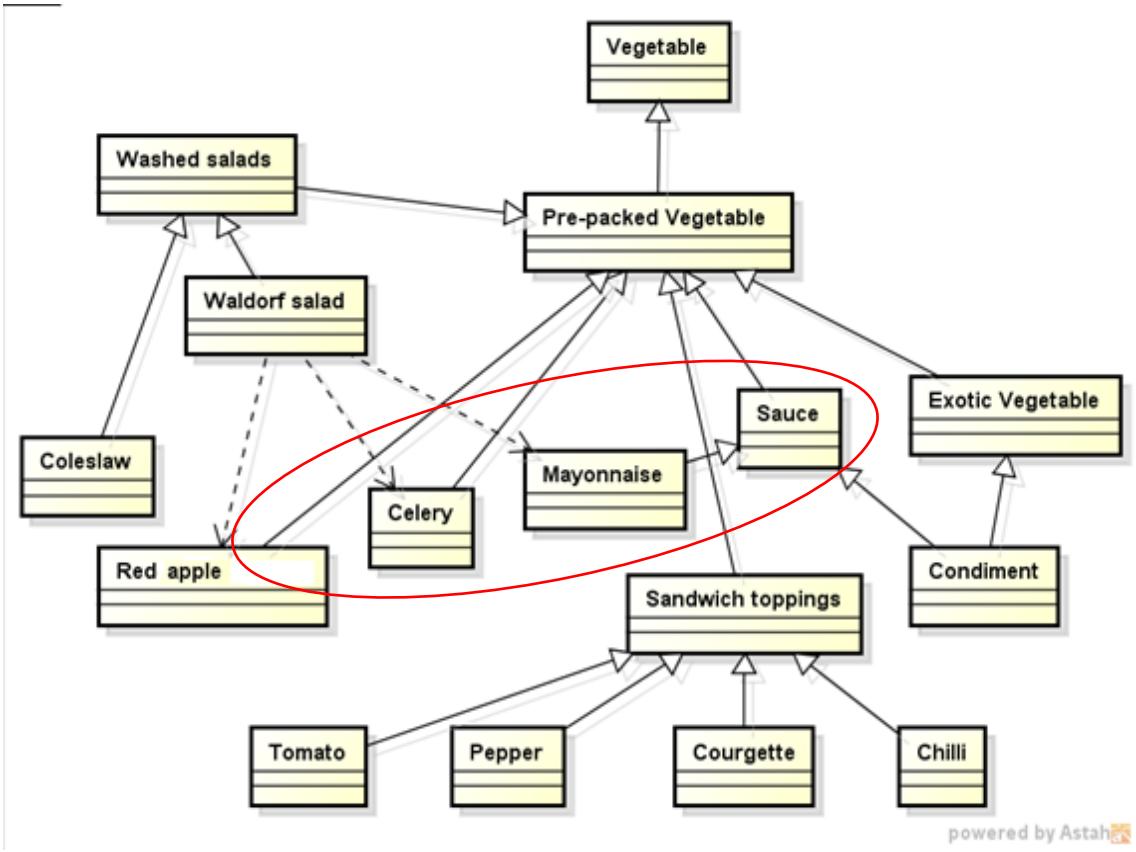


Figure 2.23 More products included.

It is difficult to apply the LSP (Liskov Substitution Principle) to the above diagram. It is based on the particular inner properties of each class in the system. Substitutions are difficult to establish here, as some items in the production line have a specific function. For instance, the mayonnaise class has been tied to sauce; however, a Waldorf salad requires Mayonnaise, no other type of sauce will work. Therefore, substituting mayonnaise with another type of sauce would mean unsatisfied customers.

In conclusion, this model is not suitable for the application of LSP (Liskov Substitution Principle).

2.2.2. Hand-In Assignment 2

1. <u>Extend and include relationships</u>

UML use cases are an efficient way of providing a description of a system's functionality in which users' interaction is clearly stated. This technique, which involves drawing use case diagrams, is widely used in software construction scenarios, especially in the requirement engineering phase. Use cases point towards a well-defined action within the functional parameters in a software system.

The involvement of several use cases in a single diagram is fairly common (they are the ovals in the diagram). Those use cases establish a close relationship with one another. According to UML specification, there are several relationships that can be established between use cases; however, the two most widely use relationships are: include and extend.

Both include and extend relationships use the notation of stereotypes enclosed by quotation marks; they show variations in terms of functionality, and in this way they complement each other in describing use cases which increase in complexity.

An <<Extend>> relationship is present when, "a use case provides additional functionality that may be required by another use case" (Bennet, McRobb & Farmer, 2010: 157). In contrast, an <<Include>> relationship is used when a use case is part of the logical sequence of functionality; in other words, a use case is included as a part of the functionality of another use case (Bennet, McRobb & Farmer, 2010: 158). With this in

mind, an <<include>> use case stereotype might be reused several times without needing to repeat the functionality with the UML use case diagram.

According to the above, the difference between extend and include relationships is based on functionality. In an include-type relationship, a use case might be reused due to the common routines or steps that are required by others. On the other hand, in an extend-type relationship a use case establishes complementary behavior in terms of functionality. The use case that required the extend relationship already had enough functions to accomplish its task in the first place; thus, the use case that extends acquires new ways to complete its function.

2. Question 6.B of the case study

In this case study, an interview was carried out to identify FoodCo use cases. In this interview, one of the production planners of FoodCo provided the information, Figure 2.24 shows the use case diagram.

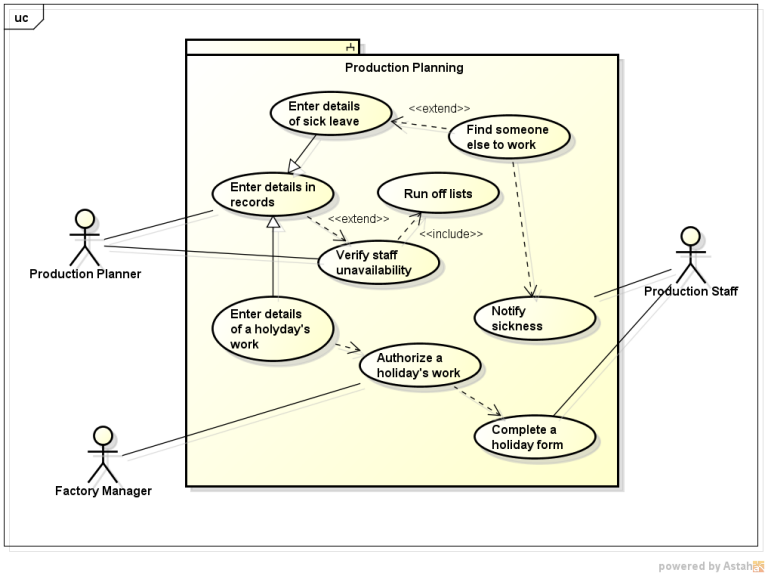


Figure 2.24 Production Planning's UML use case

Use Case Title	*Verify staff unavailability*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*Task oriented to find staff with some kind of problem that prevents them from working normally.*		
Primary Actor	*Production Planner*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	N/A		
Business Rules	*Production planners search for information in the records about sickness and requested holidays.*		
Pre-condition(s)	*N/A*		

Table 2.17 Verify staff unavailability

Use Case Title	*Run off lists*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*Look for unavailable staff in records*		
Primary Actor	*Production Planner*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Look for records of sickness and requested holidays.*		
Pre-condition(s)	*N/A*		

Table 2.18 Run off lists

Use Case Title	*Enter details in records*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*Save records permanently (databases)*		
Primary Actor	*Production Planner*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	*It extends functionalities to 'Verify staff unavailability' use case*		
Business Rules	*Customized forms are required to save the information*		
Pre-condition(s)	*N/A*		

Table 2.19 Enter details in records

Use Case Title	*Enter details about sick leave*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*Permanently save records of staff's sick leave (databases)*		
Primary Actor	*Production Planner*		
Secondary Actors	*N/A*		
Inherits	*Enter details in records*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Customized forms are required to save the information on sick leave*		
Pre-condition(s)	*A notification of sickness is required*		

Table 2.20 Enter details about sick leave

Use Case Title	*Enter details about a holiday request*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*Permanently save records on holiday requests (databases)*		
Primary Actor	*Production Planner*		
Secondary Actors	*N/A*		
Inherits	*Enter details in records*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Customized forms are required in order to save the information on holidays*		
Pre-condition(s)	*Depends on an authorization stated by a factory manager*		

Table 2.21 Enter details about a holiday

Use Case Title	*Find someone else to work*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*Look for staff available to work*		
Primary Actor	*Production Planner*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Look for available staff to replace a sick staff member*		
Pre-condition(s)	*Depends on notification of sickness*		

Table 2.22 Find someone else to work

Use Case Title	*Notify sickness*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*If a production staff member is sick, then he/she must report his/her condition using the form*		
Primary Actor	*Production Staff*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Fill out a form to declare that he/she is sick*		
Pre-condition(s)	*N/A*		

Table 2.23 Notify sickness

Use Case Title	*Complete a holiday request form*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*If a production staff member wants a holiday, then he/she must report it using a form*		
Primary Actor	*Production Staff*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Fill out a form in order to request holiday*		
Pre-condition(s)	*N/A*		

Table 2.24 Complete a holiday request form

Use Case Title	*Authorize a holiday*		
Version	1.0	**Date**	October 17th, 2012
Package	*ProductionPlanning*		
Summary	*This authorization is up to the factory manager*		
Primary Actor	*Factory Manager*		
Secondary Actors	*N/A*		
Inherits	*N/A*		
Includes	*N/A*		
Extension Points	*N/A*		
Business Rules	*Provides the production manager with details so that they may record them.*		
Pre-condition(s)	*A request is required*		

Table 2.25 Authorize a holiday

2.2.3. Hand-In Assignment 3

1. <u>Inheritance and extensible design</u>

Within the object-oriented paradigm, inheritance is an inner feature of hierarchical database environments, which consists of the transmission of characteristics and behaviors from an ancestor or a parent, to a descendant (Bennet, McRobb, & Farmer, 2010: 96). On the other hand, the principles of extensible design promote the reusability of components where there is a well-designed object oriented code (Clune et al, 2012: 2). There is therefore, a close relationship between the concept of inheritance and extensible designs; this is due to the benefits associated with the usage of object inheritance when those objects become software components.

In order to explain the important role that inheritance plays in designing extensible components, an example from the field of visual design may be used. In a graphical user interface –a.k.a. GUI– the concepts of windows, frames, and canvases are straightforward; GUI also provides mechanisms which accept end user's inputs such as textboxes, lists, radio buttons, checkboxes, and so on. Given the object-oriented environment surrounding the construction of GUIs, those objects –or components in the programming sense– use inheritance in order to transmit information on identity and related behavior. For instance, almost all components utilize a rectangular area as their "physical" space in which to exist. In this vein, a superclass of rectangular regions appears, with its own properties, methods and events. Thus, the rest of GUI's components inherit the features of such this superclass, while still able to create new components from their basic form.

2. Multiplicity in association relationship

According to Bennet and others authors (2010: 189), multiplicity in an association relationship shows the amount of objects –as instances from a class– that are participating in a relationship. Multiplicity occurs mostly in class diagrams. In addition, multiplicity can be displayed in several ways, such as a concrete number, a range of numbers, or a combination of the two. Multiplicity is a way to exemplify the number of items which interact with a class model.

To explain multiplicity, a good starting point is simple example using academic records. In an academic environment such as a university, students take several courses; at the end of the semester or academic period, each course gives a single grade, in spite

of the fact that within such courses several grades are normally given throughout the academic period. In this vein, a simple class diagram for this particular case might be used to organize these three classes (student, course, and grade), as seen in figure 2.25.

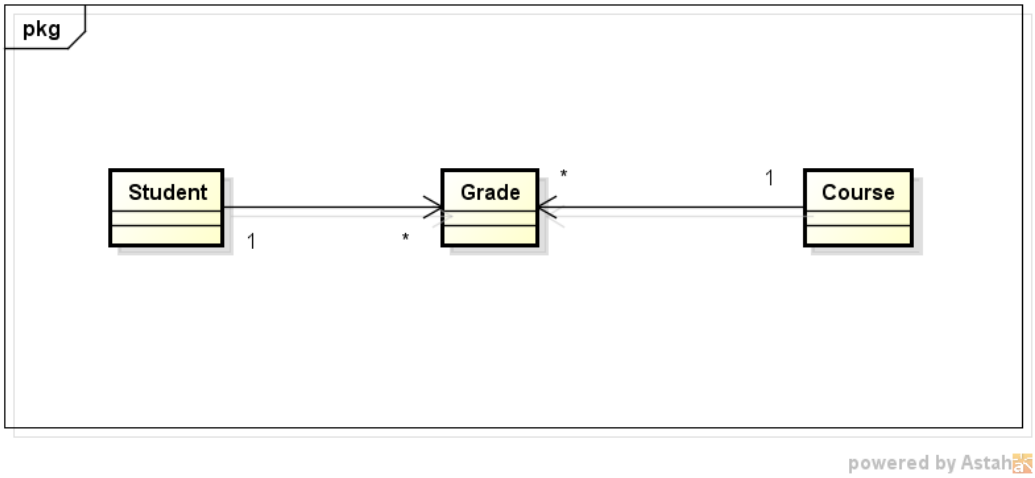


Figure 2.25 A simple example of an academic record

The previous figure represents that a student has several grades, and a course has several grades as well. This is a simple example of multiplicity in action.

3. Encapsulation and information hiding

Encapsulating is how pieces of information and their operations are defined to create a class, and determine the objects' behavior proceeding from that class (Bennet, McRobb, & Farmer, 2010: 101). Furthermore, encapsulation forms part of the object-oriented philosophy; it is considered a method for correctly managing information using classes.

On the other hand, hiding information is the true aim of encapsulation. Encapsulation may be a method of organization, but information hiding is the purpose

that it serves. Information hiding protects data which does not need to be shared; in other words, each class appropriates their own information resources.

With this in mind, it stands to reason that encapsulation is integral to information hiding in object-oriented software construction. In fact, “encapsulation is a cornerstone of object-oriented design” (Harmes & Diaz, 2008: 25) because it enables the hiding of information, promoting private and protected sections (through loose coupling environments), in which it is possible to maintain data integrity while maintaining the relationship between classes. In this way, the code produced is easier and more reliable to debug and maintain; especially when several people are working on the same software project. In conclusion, encapsulation and its relationship with information hiding promote the creation of robust and extensible software.

2.2.4. Hand-In Assignment 4

Small and self-contained classes

Classes and objects are at the core of the object-oriented paradigm. They allow the construction of robust and scalable software based on encapsulation techniques, hierarchy usage, polymorphism implementation, and so on. Taking into account the benefits of using classes and their instanced objects, a class is created in order to define a set of objects which share the same characteristics and behaviors (Bennet, McRobb, & Farmer, 2010: 93). In this vein, the description of the generalized objects should be clear and concrete.

The definition of classes within a software construction process is key when creating a logical structure and reasonable communication. A strategy of *divide et impera* or divide and conquer, is therefore employed to simplify software processes when it comes to the design phase. In this scenario, the “atomic” elements in an object-oriented design are classes which are responsible for representing a real environment, and to do that, abstraction processes are required. It is possible to compartmentalize the specific problem in order to make sense of it, into classes which tend to be simplistic, to ensure that the instantiated objects fulfil their assigned functions.

Classes should be small and self-contained; in that way, designs tend to be less complex and easier to understand and maintain. Following the philosophy of every single thing has its own duties and responsibilities, the design of classes promotes the idea that simple elements are programmed to do certain things, regardless of how they do them. In conclusion, small and well-defined classes make software construction easier.

Interaction diagrams and state machines

The analysis phase has a close relationship with the design one. Indeed, the set of diagrams which are produced in the analysis phase are the input for starting the design process. In order to gain a thorough understanding of a specific problem, the identification of classes and objects, alongside their responsibilities and interactions is one of the points that matters most. Two important diagrams are integral to this process: interaction diagrams and state machines.

An interaction diagram shows how the objects in a specific model –generally a use case– establish communication with one another by passing messages (Bennet,

McRobb, & Farmer, 2010: 262). In terms of layout, an interaction diagram is composed of vertical alignments which represent objects, entities, or actors through their lifetime in the model; messages connect the vertical alignments with clear interaction between elements. An interaction diagram represents how information is moved through the existence of the objects.

In contrast, a state machine represents the set of states and the transitions between them in the lifetime of an active object. This is a graphic tool to represent dynamic behavior in the changing status of an active object. In a state machine, one object changes its own status according to the received stimuli; at the end, it is just one object.

Perhaps the most remarkable difference between an interaction diagram and a state machine is the number of objects which are involved in it; in spite of the same dynamic behavior of both diagrams, an interaction diagram can deal with several objects at the same time, but a state machine represents transitions over one single object.

2.2.5. Hand-In Assignment 5

Refactoring in Software Development

Martin Fowler stated that, “refactoring is the process of changing software in such a way that it does not alter the external behavior of the code, yet improves its internal structure” (1999: xvi). According to his theory, it is possible to improve the inner structure of the source code in the construction of a software system to achieve two main goals: to

facilitate the maintenance of the software through easy-to-read source code, and to simplify the source structures.

In terms of coding, the software's functionality does not show any alteration from the user's perspective; refactoring must preserve the original behavior while the software is running. This technique is used at design time, where source code is available to the programmers; they can use refactoring to improve a given source code in order to make it more readable and easier to understand. This also improves code performance, which is faster when it is less complex.

The transition between analysis and design: an ATM study case

First of all, the architecture presented for the ATM case study provides a bird's eye view of the software system. In this case, a component diagram, alongside a communication architecture, forms the system's global structure. Following the design logic, it is presented as a design model based on set of packages which contain class diagrams. The top view of these packages shows a design model based on accounts and transactions that interact with the ATM Structure package.

Obviously, the class diagrams provided have more specific details at the design stage. Relationships between classes are fairly clear in their respective multiplicity; in addition, there is a close relationship between attributes and operations in a scenario with a high level of cohesion. Taking into account that the design processes are associated with the creation of a solution that meets predetermined requirements (Bennet, McRobb

& Farmer, 2010: 348), the solution should be closely linked to the implementation stage. That is to say, that programmers understand the design model clearly at the moment of coding the solution in a professional programming language.

At the design stage, its products –the design models– are the input for the implementation stage where programmers turn these models in "real" items, including: packages, libraries, and applications, among others. The design model is the equivalent of the architectural mockups in the construction of a building. The final product –in this case: software– can be reproduced in several distinct ways according to the design model.

2.2.6. Hand-In Assignment 6

Adapting the MVC pattern

The MVC pattern is the response to the need to represent information which is implicit in processing tasks. MVC (Model-View-Controller) is an architecture in which a software application is divided into three main sections: models that rule the main functionality of the software, views that are in charge of representing information through user interfaces, and controllers that allow the update interaction between views and models (Bennet, McRobb & Farmer, 2010: 379).

As a whole, MVC provides a context, problem, resources and a solution, and a space in which all the relationships between these elements can be viewed; it gives an

accurate representation of the information which is then transformed by the software. The structure and behavior of MVC is shown in figure 2.26.

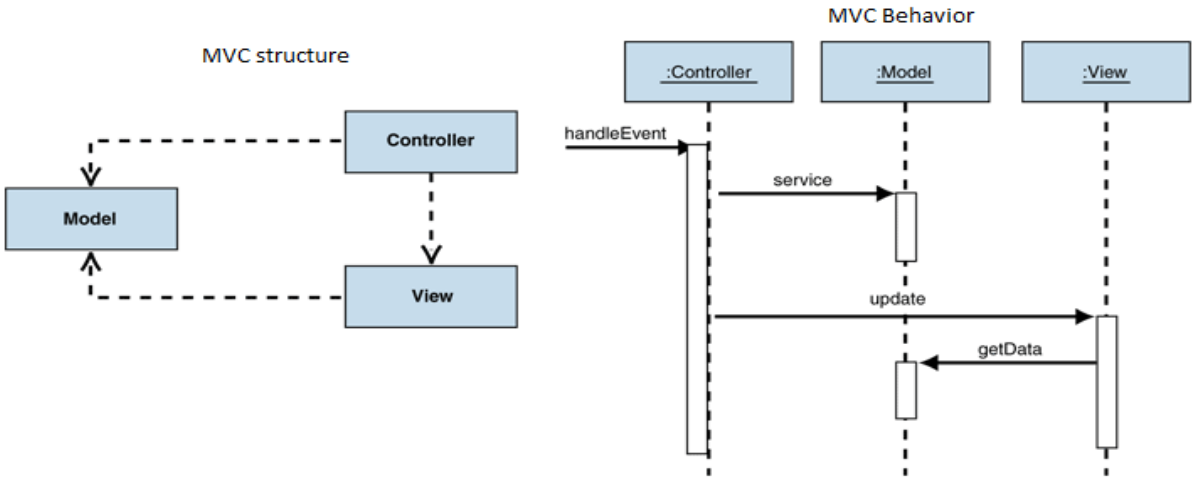


Figure 2.26 MVC structure and behavior (MSDN, n.d.).

This design pattern is highly adaptable to the AIMS subsystem, or “Scheduled Maintenance”, because of the interaction between actors (maintenance managers, maintenance staff, and even customers) within the subsystem. Each actor can order events that cause action in the model, to represent information via user interfaces. Following this principle, the design below is a proposal based on following the MVC pattern:

1. The maintenance manager gets the list of aircrafts with their statuses.
2. If an aircraft needs maintenance, the maintenance manager creates a maintenance task.
3. The following step concerns getting the relevant information about the needs of a specific aircraft.

4. The maintenance subsystem gets the list of the maintenance staff with their statuses.
5. If there are enough personnel, some of them are assigned to do the maintenance job.
6. When the maintenance is finished, the staff reports on the final status of the aircraft.

In the above, both managers and maintenance staff interact with the system in which the aircrafts' statuses are updated all along the maintenance process. This implies that the user interfaces will experience changes in the information shown; the MVC patters fits with the features of the subsystem.

In terms of design, the controller resides in the *Schedule Maintenance* class with the companion of the *Maintenance Task* interface and its inherited classes as the model. Finally, views are represented by the user interfaces for both managers and maintenance staff.

The benefits of implementing the MVC pattern in the Scheduled Maintenance subsystem would be most obvious when several platforms were used. In that case, the MVC pattern could be implemented in a web-based application in which different devices could use the applications, even on mobile devices.

2.2.7. Hand-In Assignment 7

1. Detailed Design Information for the ATM example.

In the ATM system architecture based on a class diagram, changes in visibility were observed as seen in the following table:

Class name	Element*	Previous Visibility	New Visibility
Cash	(O) determineHowManyBills	Public	Private
Dispensing::Money	(O) updateBillCounts	Public	Private
	(O) fundsAvailable	Public	Private
CustomerIdentificati	(O) startMonitoring	Public	Private
on::CardReader	(O) lock	Public	Private
	(O) unlock	Public	Private
	(O) keepCard	Public	Private
PIN	(O) encrypt	Public	Private
Accounts::Acount	(O) ATM_Accessible	Public	Private
	(O) withdrawalAllowed	Public	Private
Customer	(O) attachPIN	Public	Private
	(O) attachAccount	Public	Private
BankSystem	(O) performTransaction	Public	Private

Table 2.26 Architecture of an ATM

*The element of the class can be (A)ttribute or (O)peration.

The result of updating role names and visibility scopes is shown in figure 2.27.

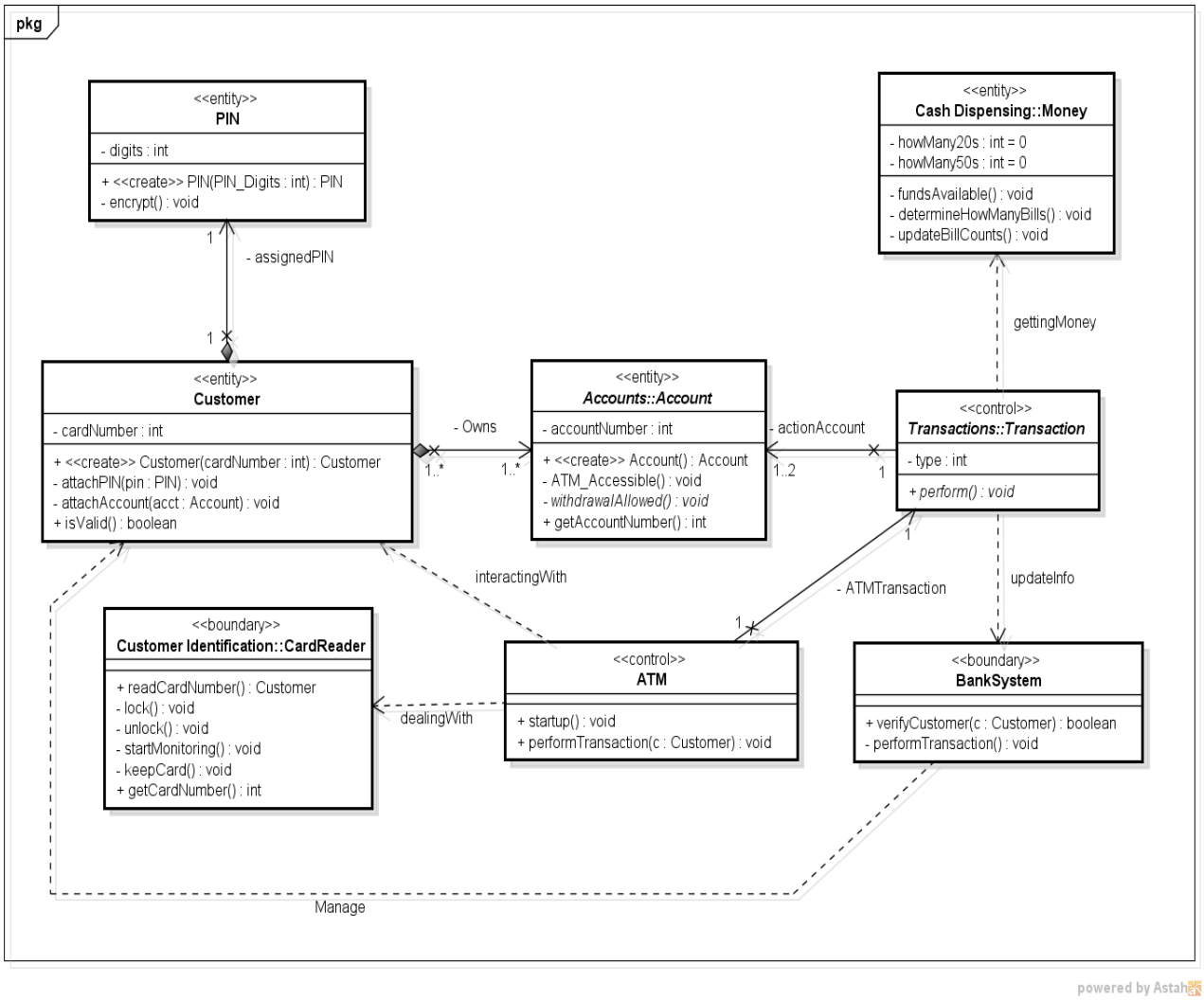


Figure 2.27 Modified ATM System Structure (Class Diagram)

2. DbC Applied to the ATM example.

Outlining an analogy using contracts in a business environment, the concept can be adapted to specify the communication mechanisms in an object-oriented software system (Meyer, 1991). In this case, there are two classes –Account and Money– which are going to be used in terms of contracts for the DbC approach.

The structure of these classes is shown in figure 2.28, with a new adaptation in the given result type of the operation in Money, *fundsAvailable* will return a Boolean data type.

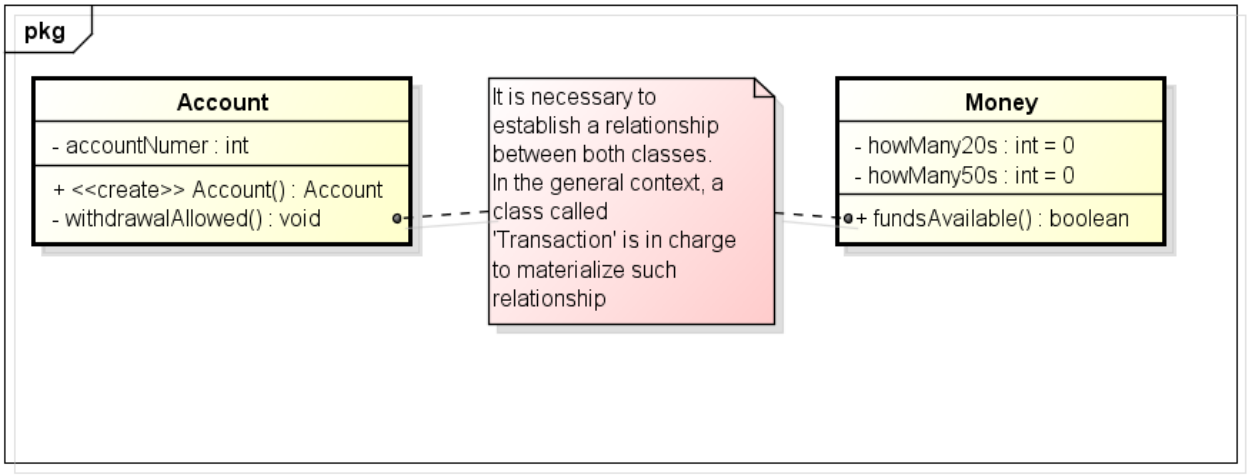


Figure 2.28 Brief class structure for Account and Money.

In this case, two operations will be looked at when working with the DbC criteria: *Account.withdrawalAllowed* and *Money.fundsAvailable*. The generated code in outlines for pre and post conditions are as follows:

```
using System;

public class Account
{
        private int accountNumber;
      //-----------------------------------
        public Account Account()
        {
        /*
       * This is the constructor method
       */
        }
```

```
        //-----------------------------------
            private void withdrawalAllowed()
            {
             /*
              * Pre:  Is there enough money?
              * Post: none
              */
              if (Money.fundsAvailable()) // a intermediate class is required due to the lack of a direct
relationship
                                          // between both Account and Money classes
             {
                /*
                 * allow withdrawal operation in this account
                 */
             }
            }
        //-----------------------------------
    }



using System;

public class Money
{
    private int howMany20s = 0;
    private int howMany50s = 0;
    //-----------------------------------
    public boolean fundsAvailable()
    {
        /*
        * Pre:  none
        * Post: Count how many bills are available.
        */
        return (true);
    }
    //-----------------------------------
}
```

2.2.8. Hand-In Assignment 8

When facing the construction of complex software systems, using an object-oriented approach has enormous advantages in terms of scalability, robustness, maintainability, and reliability, to name but a few. In this module about system analysis and design with an object-oriented approach, several aspects of the paradigm were covered, without using a strictly specific software process. It was therefore established that the usage of the object-oriented approach is applicable to any software process, regardless of which process is followed.

When it comes to standardization, the Unified Modeling Language –UML– is a graphical language widely used to define software systems from the analysis and design stages (Bennet, McRobb & Farmer, 2010: 77). Today, UML is composed of a series of diagrams, which are used in different phases of a software process. In particular, this module covered the generalities of requirement analysis and design (architecture and detailed design). A set of UML diagrams were used, including use case diagrams, class diagrams, activity diagrams, sequential diagrams, integration and collaboration diagrams, package diagrams, and component diagrams. An interesting point in this module was the employment of a non-UML technique used to identify both structure and behavior in classes, the use of CRC.

The main purpose of software design is to create a clear guide as to how a software should be built. A good design enables easy software maintenance and support; in addition, a good design is the basis of good quality software, where scalability is perhaps one of the most relevant aspects to take into account. On the other hand, all the aspects

of a software construction process must work in harmony. In the whole process, all the diagrams must be conceptually connected to a sound theoretical background for software construction.

Finally, software design is a 100% intellectual labor, involving a high degree of creativity. It is also important to note that computer programming is just one part of the entire software process. The trend in software construction points towards increasing skills in design, rather than implementation. It is striking how development platforms and languages have evolved, to the extent that design is now a key part of software construction. In the authors' experience, Microsoft .NET technologies are an example of this, where the languages themselves –such as Visual Basic, C#, C++, F#, and so on– are not the crux of the development of complex software solutions. Programming languages are becoming more and more trivial; today's development platforms have powerful code editors in which intelligent syntax is provided with refactoring functions and so on. In the end, software reuse is the jewel in the crown of software construction. Perhaps in the near future, the human element of computer programming will become obsolete, but Software Design will remain.

3. SOFTWARE ENGINEERING

3.1. REFLECTIONS

3.1.1. Three Specialized Software Tools at Universidad de Nariño

This case study is based on a small public university in southern Colombia, the University of Nariño. The authors' main occupation consists of teaching computer programming at undergraduate level to Computer Science students. Software development has always been an integral part of that teaching, and the authors have also been able to construct the software tools used by the university during their careers.

The university has an integrated information system whose core is academia. Taking into account the nature of the organization as a whole, several subsystems were developed in accordance with the university's many departments. These subsystems are shown in the following figure:

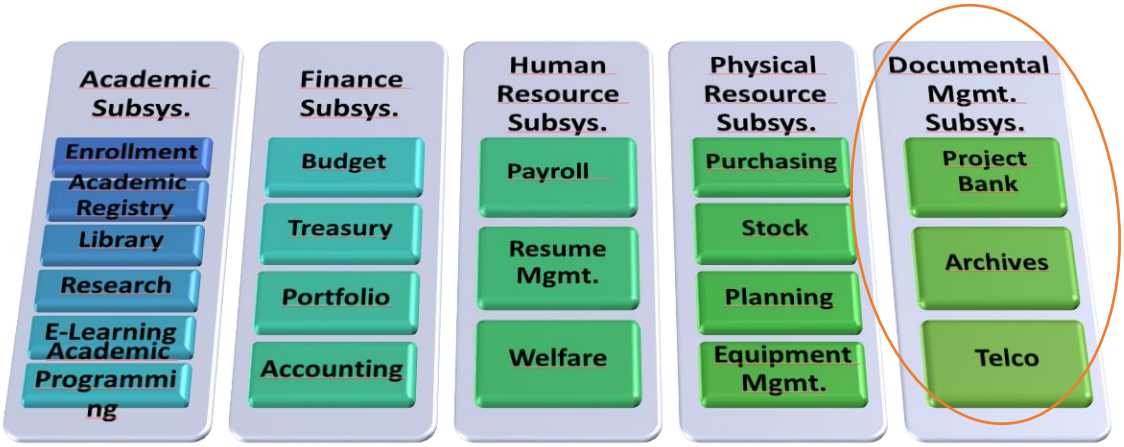


Figure 3.1 Subsystems at the University of Nariño

The structure above was proposed in 2009, and since then, the computing department of the university has been working to construct this complex system. Today, all of the subsystems are complete apart from the last: Documental Management Subsystem.

This subsystem is divided into three main modules: Project Bank, Archives and Telecommunication. This last subsystem was conceived as the support for processes between the transactional and the business intelligence levels. Both archives and telecommunication modules are responsible for managing the data warehouse, and carrying out knowledge discovery processes via data mining.

Taking into account that the data mining process looks for non-trivial information in complex relations, new knowledge about the organization is a priority when it comes to decision making. Data mining has three main features: making sense of content, handling large amounts of data, and working with mostly unsupervised data (Krzysztof, 2007: 4).

In the quest for new knowledge in the interests of the university, the three aforementioned modules could be considered software-based tools used to facilitate the decision making process. In this vein, these modules have a huge impact on the process of improving the university, not only in academic terms but in all its capacities.

Currently, the computing center at the University of Nariño is implementing software engineering techniques following a waterfall model. The documentation generated by this process is extensive and detailed. When constructing software, the computing center works with Oracle Database, and the development platform is supported by Microsoft Visual Studio and .NET technologies. Microsoft Visual Studio is

most commonly used as an architectural tool. Although Microsoft Visual Studio works well with Rapid Software Development via Agile Methodologies, it is also possible to support complex processes with sound documentation self-generated by this platform.

In conclusion, the three tools developed using Microsoft .Net Technologies are: Project Bank, Archives and Telecommunication. They are the starting point for data mining support processes with a profound impact on the reviewing of current software processes. All these tools (or modules) are part of the Documental Management Subsystem which is currently under construction.

3.1.2. A Secure Architecture for Client-Server Systems

Security is one of the most common concerns in software construction. Since the beginning of the computing era, authentication and authorization mechanisms have been implemented in order to protect information against possible attackers. Early information systems were based on a centralized approach, therefore the responsibility for the security of an entire system lay in one location: the server.

As software-based solutions and the new requirements and expectations of end users became more complex, client-server systems were powered through networking environments with better hardware infrastructure. In contrast with centralized systems, client-server systems enabled the use of clients' computing power; in this vein, the information processing was, and still is, carried out in both servers and clients. However, this approach does still raise some security concerns.

At least three levels of protection must be implemented when establishing the security criteria in a client-server systems' architecture: platform-level protection, application-level protection, and record-level protection. Platform-level protection is related to the user working with an information system; application-level protection is associated with a secure environment within the application itself; and record-level protection refers to data management (Sommerville, 2011: 377).

Following this outline, the layer-based architecture has been constructed as seen in the following figure. In this example, Microsoft Technology was used for its final implementation.

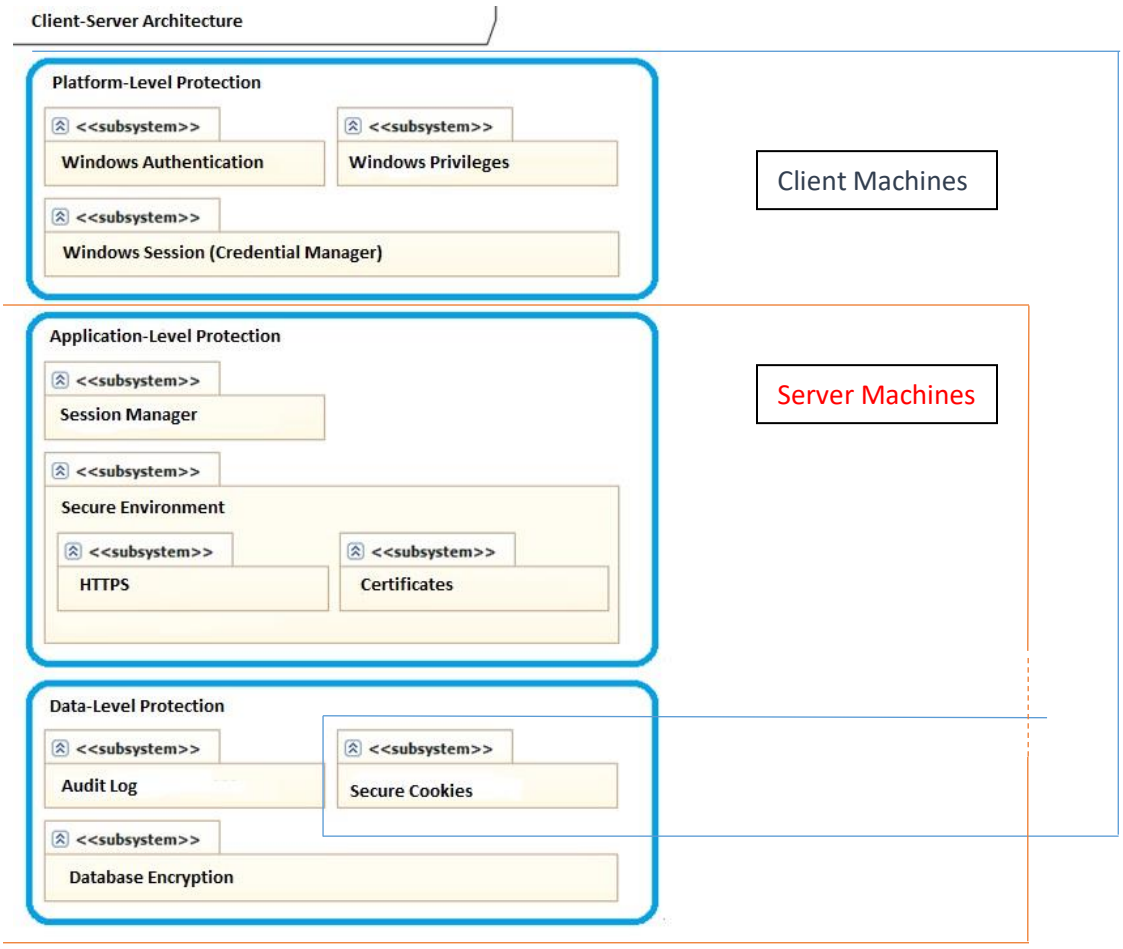


Figure 3.2 Client machines and Server Machines

The architecture above shows a set of client machines with a protection system based on their platform; in this example, Microsoft Windows Clients is responsible for the distribution of this architecture. The three subsystems (Windows Authentication, Windows Privileges, and Windows Session through Credential Manager) are supported by Microsoft Active Directory Technology. In addition, for web based applications, clients have secure cookies in order to protect the web session information in-client.

In the application-level protection, the architecture creates a secure environment based on HTTP over Secure Socket Layers (HTTPS) with the required certificates. In this example, this can be done by using Microsoft ASP.NET version 4. The session manager is part of the policy of authenticating and authorizing web users, including: Windows, Forms, and Passport (Microsoft, 2003).

Finally, the data-level protection is largely related to server environments with functioning databases. In this vein, database encryption is reached using inner techniques and tools available on the Microsoft SQL Server. This database engine provides an audit infrastructure based on log files.

It is important to note that the web application is based on compiled units of Microsoft ASP.NET technology. However, the complied assemblies run on the server as parallel processes. Partial client exposure takes place when, within the application, AJAX technologies are used. In this example, Microsoft AJAX Control Toolkit for Microsoft .NET Framework 4.0 was employed and as such, the validation of GUI interaction is fairly important.

3.1.3. Feasibility Studies of Software Production

In software engineering, a feasibility study is an activity carried out in the early stages of a software process, especially in the requirement engineering phase. In general terms, a feasibility study determines whether or not a software system is able to contribute to an organization; in addition, it whether or not a software system is viable, taking into account the organization's staff and IT infrastructure (Somerville, 2011: 100).

It is important to include all of the costs vs benefits of a software system in a feasibility study. Whether the software system will cover all requirements stated by the organization for instance, or whether the current hardware platform will support the final computing solution, and whether the current personnel in charge of the construction of that computing solution are fully competent. All the areas mentioned above are directly related to software, hardware and people.

These main considerations notwithstanding, there are other elements which must be taken into account when determining whether a software system is feasible, or not. There are two main environments in which feasibility studies take place, with different priorities for each: companies that produce software for the open market, and companies that produce software for their own needs. In the former, additional aspects of a feasibility study, present only when the software is a product (or service) for the market, are: market opportunity, risks, status of business competitors, and target customers, among others (Gall, Grechenig, & Bjerre, 2011: 17). In the latter, the additional elements of a feasibility study are related to production costs and final performance (Devaraj et al, 2011: 201).

It is important to take into account market opportunities when producing software for the market. Kamper states, that “software markets are defined as markets of intangible software programs, services, and applications” (Kamper, 2009: 29). These markets are influenced by people’s software consumption. The consumer can be classified in terms of integration, ubiquity, massive consumption, etc., and these concepts must be part of a feasibility study when a company is looking to place a software product/service on the market.

If a company wants to place its software on the global market, it must consider a competitive dynamics analysis that examines the moves that rivals make to reach advantageous market positions (Upson et al, 2012: 93). This is another essential criterion of the feasibility study in order to establish whether or not software productions will have an opportunity on the market.

Finally, in the case of a company which produces software for its own needs, the cost-benefit relationship is a key factor that impacts on a feasibility study. Questions are raised such as: Is it worth investing time and effort in building own software? Or, is it more convenient to outsource software development? In either case, the construction of software always incurs some kinds of costs.

3.1.4. Air Traffic Control System – An Object Model, Classes and Hierarchy

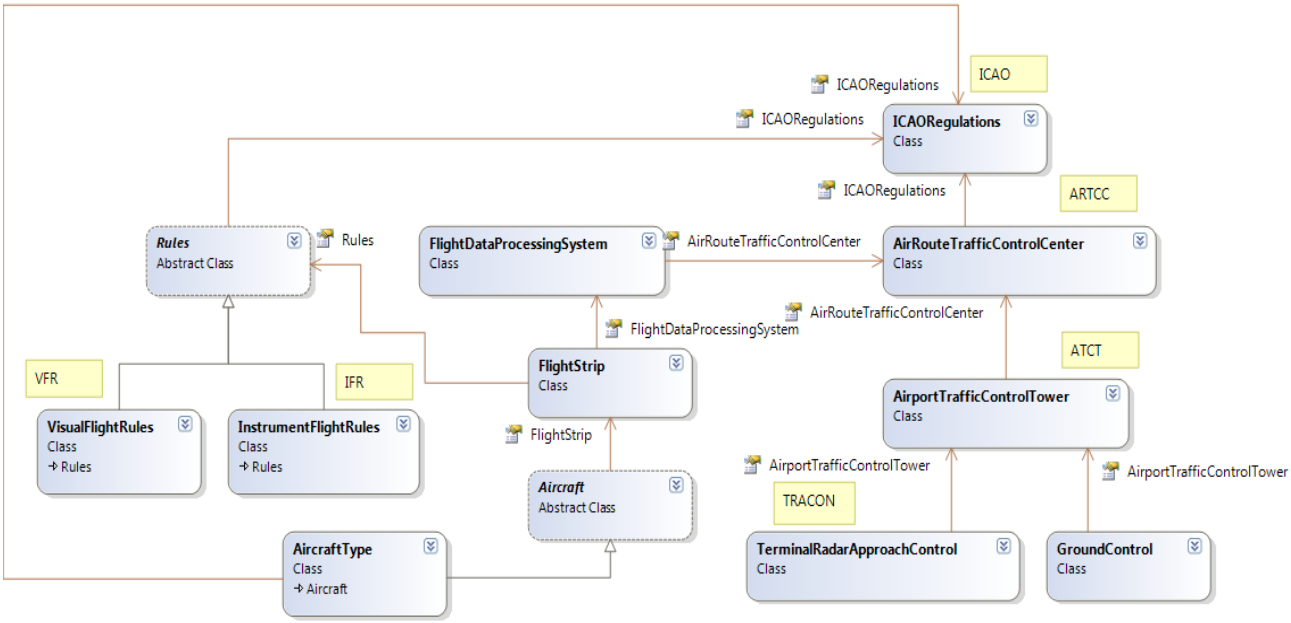


Figure 3.3. Class diagram

3.1.5. Cloud Computing and its Impact

In computing, creating logical entities has always been a necessary part of controlling physical devices. This is how the concepts of software and hardware arose. The divide between software and computer programs has been blurred and as such, the terms are often used interchangeably.

As computer-based solutions have become more complex in response to rising demands, creating software has become an increasingly hard task. This is the narrow field in which software engineering plays a transcendental role. By definition, software

engineering is a discipline within the field of engineering, that works with all aspects related to professional software production (Sommerville, 2011: 7).

The difference between writing code in computer programming, and constructing complex software systems following quality criteria should be fairly evident. Taking into account professional methodologies used to create software processes, the divide between writing computer programs and creating complex software-based solutions is not so blurred as it first appears. In fact, computer programming is merely a small part of the entire software process.

The evolution of software has reached such heights, that it no longer requires the installation of additional components, besides a web browser and some plug-ins. This is the perfect context for web-based solutions to reach their full potential. Web development has existed since the creation of the WWW in 1990s (Connolly, 2000). Over time, more functions have been added to the web, such as client-server scripting, asynchrony, data access, enhanced user interfaces, and security, among others.

Nowadays, web development uses the computing power of virtual environments: Cloud Computing. James Whittaker, engineering director at Google, stated that, “in the cloud, all the machines automatically work together; there’s monitoring software available, and one test case will run anywhere” (Shull, 2012: 5). Software development has undergone some changes in its implementation for use by cloud computing in virtualized environments.

Citing an example, Microsoft Corporation has its Cloud Computing services through Windows Azure. On this platform, how developers create their software solutions

differs a little; in essence, the paradigm of Software Plus Service (S+S) is maintained through Web development, specifically ASP.NET, XML Web Services, WCF Services, and RESTful support. Furthermore, Windows Azure infrastructure must be added as a complementary project in the solution. A given solution is implemented locally through emulators for processing and storage levels. Finally, the solution can be uploaded to a specific location in world of Microsoft's server farms, using a valid subscription.

Evidently, development and deployment of software solutions in Windows Azure is not difficult using Microsoft's tools and technologies. It directly impacts the way people create complex software systems based on cloud services. However, in spite of its benefits, Cloud Computing has been said to have security issues; some IT professionals see the action of moving mission-critical applications around a "remote place" which is controlled by someone else as risky (James, 2009: 3).

Cloud Computing allows customers to benefit from software solutions anywhere. Its impact can be seen in (almost) all aspects of daily life. Perhaps, developers of "traditional" applications should consider moving towards the Cloud to keep their place in the competitive market. Finally, an important question is raised: will everything eventually be built in the cloud? Only time will tell.

3.1.6. Windows® Internet Explorer 9 Configurations

Windows® Internet Explorer is a powerful, cutting-edge web browser created by Microsoft Corporation. The latest version is IE9, including new features which speed up the user's web browsing experience. These features include: new browser controls,

taskbar management, use of accelerators, fast searching, download manager, work with tabs, and enhanced security, among others (Microsoft Corp, n.d.).

The basic configuration of this web browser is quite simple. For end users, Windows Internet Explorer is easy to configure at home and at work. It includes parental control settings, tracking protection, InPrivate Browsing, SmartScreen Filters, the freedom to manage web pages' privacy policies, add-on management, and so on. Its interfaces are intuitive with strong online support.

The needs of most domestic end users are fairly straightforward. However, there are other users whose requirements are more advanced, those who work with Microsoft technology and infrastructure. They are the host administrators based on the Windows® Server operating system.

A server is a machine with management software for distributing services and resources to end users or customers. Nowadays, online services are fairly common, such as web-based applications, web services, FTP services, media streaming, distributed computing power, and storage, among others. In these various contexts, servers have well-defined tasks which must be carried out to offer online services. For security reasons, it is not advisable to browse the Web while in session as an administrator.

When an administrator in a server's desktop session tries to browse the Web using Windows® Internet Explorer, the following warning pops up, as shown in figure 3.4. This is due to the potential risks of browsing uncertified websites; malware can be unleashed by running hazardous scripts.

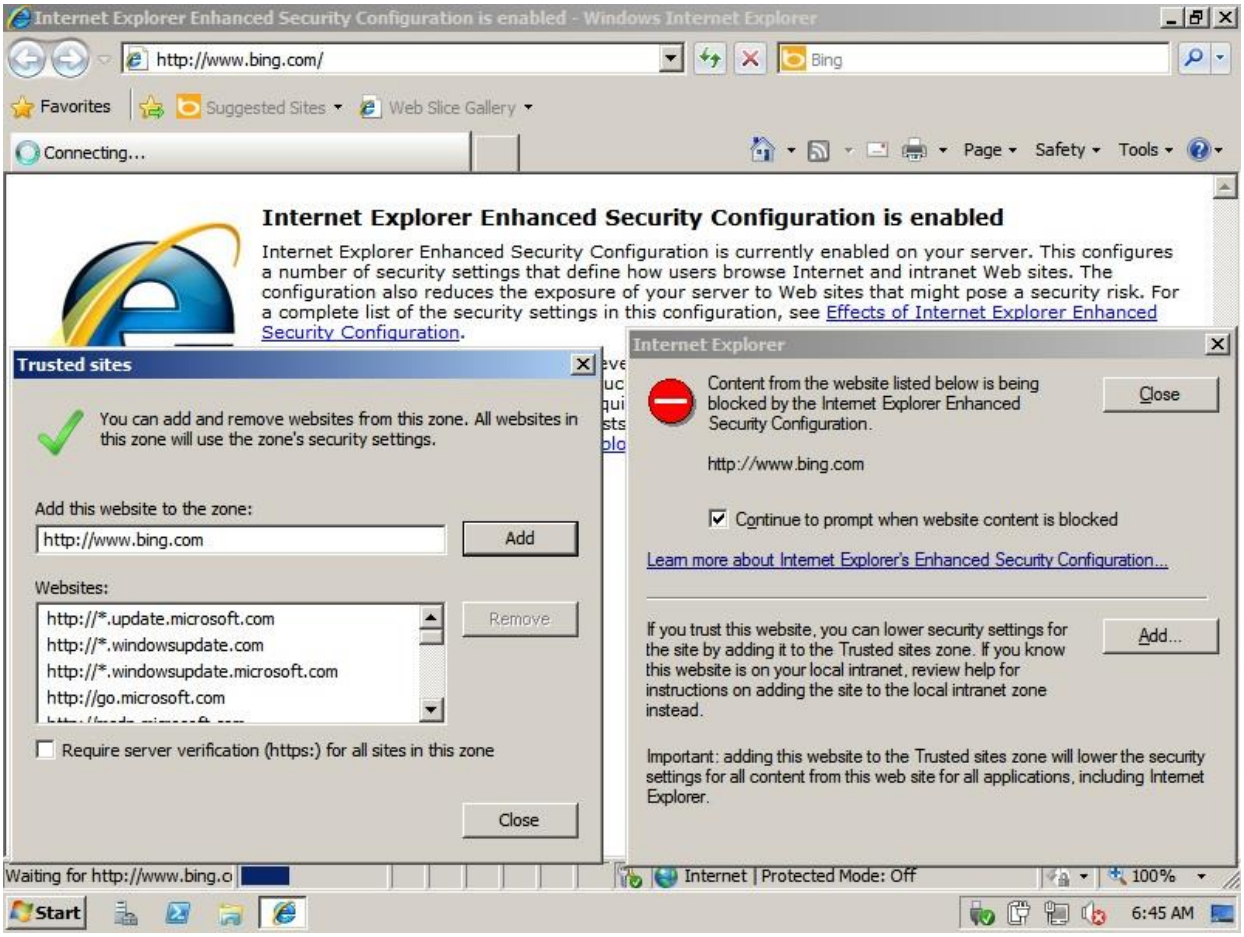


Figure 3.4 Default message concerning Internet Explorer Enhanced Security Configuration.

On a Windows® Server, if other users with no administrator privileges are using server's desktop sessions, they cannot browse the Web directly without seeing warning messages. For each new website opened, end users must do one of two things: register the new website within the zone as a trusted site, or cancel the validation process. This validation is required for each new website opened on Windows® Internet Explorer, seen by some as a tedious chore, but for others it is an essential security requirement to protect the server.

End users only can configure trusted sites for their sessions. This is done using the Internet Options available on Windows® Internet Explorer and creates what might be termed “intuitive” personalized online security (Microsoft Corp., 2010). In spite of the individual configuration for each user however, general internet security policy is ruled exclusively by the administrator; no one else can make alterations.

Only the server administrator can enable or disable these security settings which are a feature of Windows® Server operating systems. To change this configuration, the administrator has to enter the Server Manager, which is one of the Administrative Tools in the Control Panel. Then, the administrator can activate or deactivate the enhanced security configuration of Windows® Internet Explorer. As such, the Enhanced Security Configuration can be switched on and off for administrators and other users.

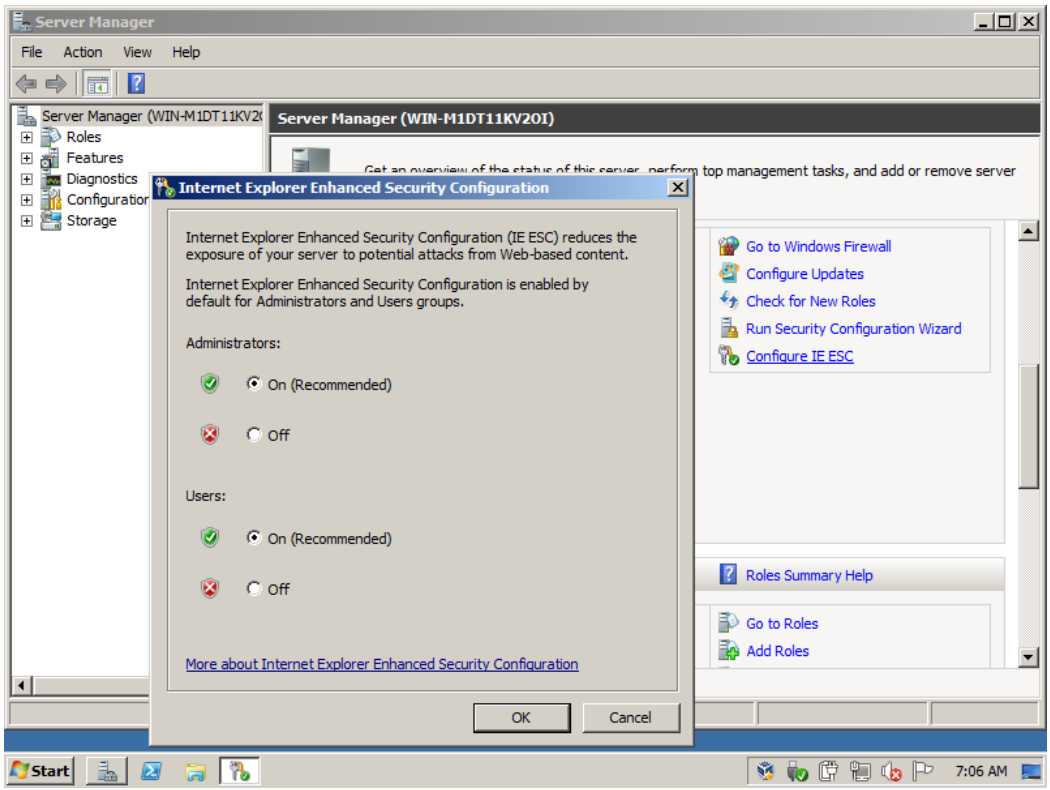


Figure 3.5 Internet Explorer enhanced security configuration

3.1.6. Costs related to Software Reuse

Currently, a lot of research is being done into software reuse, although its definition varies, Charles W. Krueger for example, defines software reuse as the discipline responsible for creating software systems from pre-existing software (Krueger, 1992: 136). Ian Sommerville states, that reuse is the process of creating software systems from existing software parts, rather than building them from scratch (Sommerville, 2011: 426). Considering the ambiguities surrounding the term ‘reusability’, some standards from various quarters have been put forward.

One of the standards developed by IEEE Computer Society, which was coded 1517, covers all the relevant aspects to be considered when establishing the life cycle of a piece of software from a reuse-oriented perspective (IEEE-CS, 2010: 1). In general terms, the structure of the IEEE Computer Society’s proposal encompasses the following content: references and definitions, application of a standard, reuse in the life cycle of a system, and reuse in the life cycle of software. This has been summarized in the table below:

Reuse in Software Life Cycle	
Stages	**Processes**
Software	Software Requirements Analysis
Implementation	Software Architectural Design
	Software Detailed Design
	Software Construction
	Software Integration and Software Qualification Testing

Software Support	Software Documentation Management
	Software Configuration Management
	Software Quality Assurance, Software Validation and Software Verification
	Software Review, Software Audit and Problem Resolution
Software Reuse	Domain Analysis and Design
	Reusable Assets Construction

Table 3.1 Integration of reuse into software-specific life cycle processes. (IEEE-CS, 2010: 21)

In this IEEE-CS 1517 standard, reusability can conceivably be part of every stage of each process. Ergo, a reusable asset may not always be a software entity; indeed, it is possible to reuse documents, designs, templates, etc. As a matter of fact, the context which best lends itself to reusability is the construction of software itself; this means the reuse of software entities in other environments, including: application system reuse, component reuse, and object/function reuse (Sommerville, 2011: 426).

The main advantages associated with software reuse lie in the benefits that organizations that incorporate it can reap. Quality improvement and reduction of effort for example; which in turn result in the increased productivity of the development team.

In spite of these obvious benefits, software reuse has several requirements that should be taken into account, such as: domain analysis, increased documentation, maintenance and enhancement of reuse artifacts in terms of documents and components, royalties and licenses for externally acquired components, creation (or acquisition) and operation of a reuse repository, and training of personnel in design for reuse and designs

with reuse, among others. This lengthy list makes evident that software reuse has other implicit costs beyond the cost of the software component itself (Vasantha et al, 2008: 953).

In conclusion, there is no direct relationship between the costs of development/distribution of a reusable component and its size. Other factors affect the estimation of costs as previously mentioned: deployment, maintenance, and training, among others. The specific costs associated with software reuse are “cataloging reusable assets and including them in the reuse repository. Searching for reusable assets also contributes to the costs” (Nazareth & Rothenberger, 2004: 248). As such, the implementation and adaptation of reusable software is costlier than the component itself.

3.1.7. The Development View in Detail

While there is a wide range of symbols used to illustrate system architectures, graphical symbols like boxes and arrows can have several interpretations, depending on the reader. Boxes can be programs, application modules, hardware, pieces of source code, components, etc. In the same way, arrows could represent dependency, flow control, sequentially, and so on. These symbols are in fact used to represent a bit of everything (Kruchten, 1995: 42).

In order to “organize” this mixture of approaches to interpretation, a general perspective was proposed in 1995 by Philippe Kruchten. It covered all aspects related to the creation of system architectures, at the time of course, and it was called: the 4+1 View Model of Architecture. This model is shown the Figure 3.6 as below:

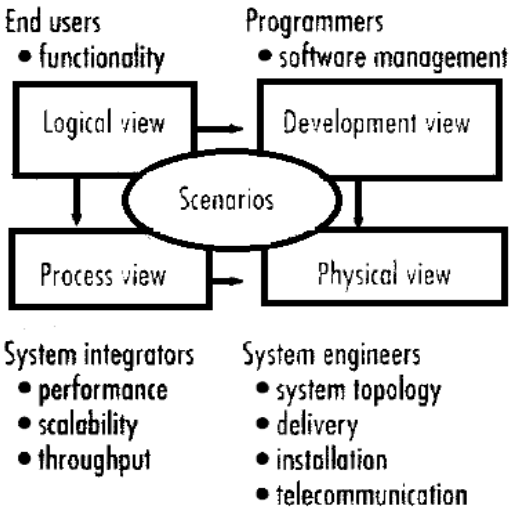


Figure 3.6 The 4+1 View Model (Kruchten, 1995: 43).

In accordance with the 4+1 View Model, all integrated scenarios have their own views and actors. In software management, programmers find the Development View a good starting point. In this field, the author of such a view model focuses on the organization of software modules within complex software-based systems; these modules can be called libraries or subsystems. Based on Development View concepts, a complex software-based system is broken down into several components which are built by programmers (Sommerville, 2011: 154).

Today, one of the most common ways to represent software systems is through the Unified Modeling Language, UML. This language was created using the combined efforts of several authors of previous techniques, including the Booch method, OMT, and OOSE (covering OOA and OOD), among others. Nowadays, its current specification is UML 2.x (2.4.1 for infrastructure and superstructure) and it is supported and maintained by OMG, The Object Management Group.

Using this “new” perspective in software modeling, the “classic” development lifecycle can be graphically represented, including the implementation phase, using UML and the principles of the Rational Unified Process (in particular, the Software Development Life Cycle). The strong relationship between the construction and implementation phases can be observed in Figure 3.7:

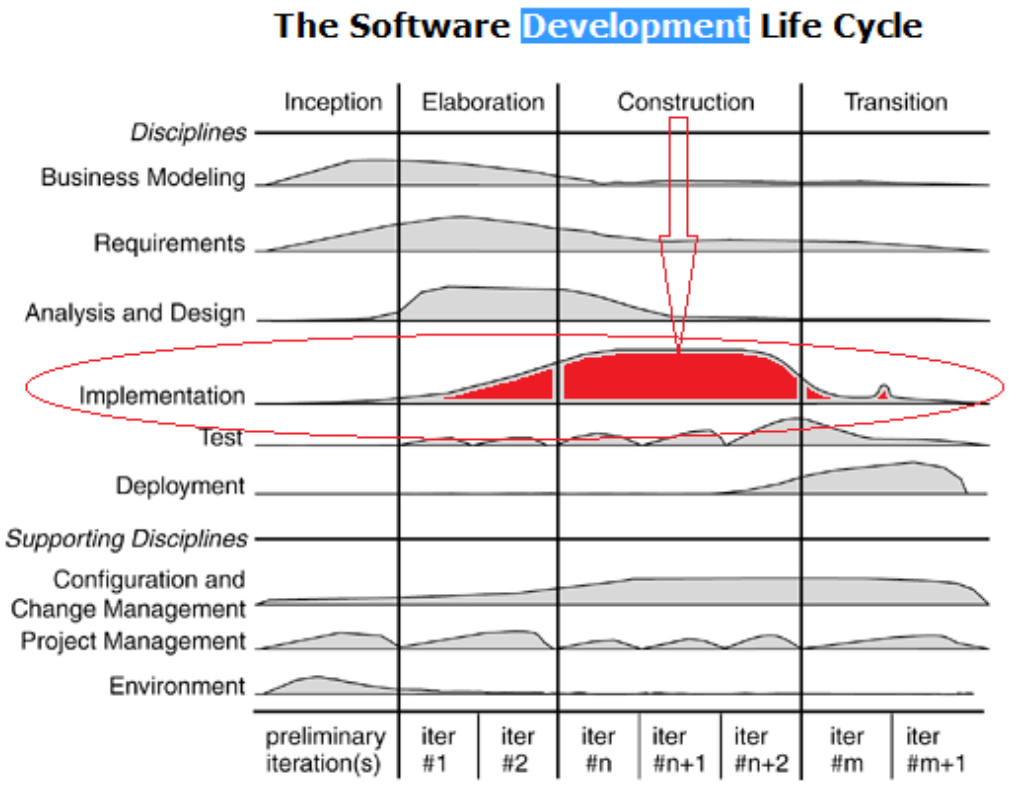


Figure 3.7 The Software Development Life Cycle (Booch, Rumbaugh, & Jacobson, 2005: Appendix B).

The implementation phase is the point at which the software system is assembled, involving the construction of components and artifacts to produce a functioning system. In the implementation phase, “the dynamic aspects of this view are captured in interaction

diagrams, state diagrams, and activity diagrams". Its composite structure can also be represented in class diagrams (Booch, Rumbaugh, & Jacobson, 2005: 33).

The advantages of using UML as part of the implementation and development phases –as part of the architectural construction– are associated with independence of construction platforms and computing languages. This feature is particularly beneficial when a given architectural design in the development scenario is to be constructed in different platforms and languages, in order to reach a wider range of customers.

In contrast, a possible disadvantage is related to the necessity for thorough documentation throughout implementation and development. This situation could arise, depending on the type of software production methodology that is applied. For instance, in plan-driven methods, the documentation related to the development phase should be detailed and well-defined; however, those using Agile methodologies could be more flexible about the rigorousness of their documentation.

As with all aspects of human life, everything is a matter of habit. Perhaps, some people find it difficult to adapt to a new methodology, and for others it is no problem at all, but the ability to adapt makes all the difference when it comes to changing methodology.

3.1.8. Experiences with Stakeholders

The identification of stakeholders is the starting point when it comes to collecting software requirements at the early stages of a software process. A stakeholder is a person with a direct interest in a software-based solution. Stakeholders are involved in the usage,

and sometimes even in the construction, of software; in addition, they are considered the most important source of information when it comes to establishing requirements (Sommerville, 2011: 103).

Unfortunately, some software development projects fail due to stakeholders, as lack of knowledge, time, or interest for involvement in the project on the part of a stakeholder can cause a project to flop (Lim, Quercia, & Finkelstein, 2010: 295). Such a case is detailed below based on personal experience.

The authors have been professors of computer science for over 15 years in a small public university in southern Colombia. Parallel to their work in the field of education and research, they have had the opportunity to participate in software construction projects in cooperation with the computing center of the university. As such, they have been involved in the process of choosing stakeholders many times.

For instance, in the construction of the academic integrated information system, the identification of stakeholders was driven by the rector of the university's requirements. Taking into account the politically complex –even chaotic– environment which constitutes a public university in Latin America, university rectors have supporters and contractors (as well as critics and opponents) depending on their ideas and political affiliations. With this in mind, a university rector's demands are welcomed in some quarters, but sometimes rejected in others.

In the University of Nariño, there is a Student Affairs Division who are responsible for programs and services that focus on students and their academic experience. In addition, this department liaises with the faculty, staff, parents, alumni and the wider

community in general. Five years ago, the director of the Student Affairs Division was working on the construction of the user requirements needed in order to create an integrated information system about academic data. This task had been assigned to him by a previous university rector who was a close friend of his, but who had unfortunately died suddenly. His replacement, a divisive and combative person, was in charge of the university and as a result, the work stopped.

An important factor when involving stakeholders in a project is their attitude and the atmosphere in the workplace. Interpersonal relationships are an important ingredient when facing processes which involve the participation of several people.

3.1.9. Facing the inevitable in Software Construction: Changes

Today, software is present in (almost) all aspects of our lives; in this vein, software should effectively respond to the dynamics of human life. Therefore, the first contact (or relationship) between users and the team in charge of the construction of specific software, takes place in the system requirements phase, no matter which software process is being used (Sommerville, 2011: 43).

In this context, changes following “business logic” can lead to alterations in the final software based products. Establishing a clear strategy to face the evolution process of software is the best way to minimize the negative effects of external changes. Unfortunately, this is easier said than done, and unexpected changes will always generate costs and waste time.

The aforementioned strategy should be based on the anticipation of change under certain criteria, such as change avoidance and change tolerance. However, a clear vision in the design and development phases is essential in order to ensure adaptability. In consequence, Sommerville recommends two strategies to minimize the changes' negative effect: using prototypes and incremental delivery (Sommerville, 2011: 44).

Another factor that generates changes in the software design and development phases, is the promotion of new approaches and techniques in design and programming. For instance, the difference between facing the challenge of constructing a "traditional" simple Web site and constructing a complex Web site based on an MVC 3 approach. In the following example, the team in charge of software construction faces modifications in terms of design and implementation, depending whether they use a traditional model or MVC 3.

According to the authors' experience in software construction, technological changes (in the field of design and implementation) are not so hard to deal with if the team is using the latest design/development tools and platforms. The main problem arises when system requirements are altered. In the first situation, it is preferable to work with a Component-based Software Development approach with strong tech support, such as Microsoft .NET or Oracle J2EE.

An interesting model for flexibility has been proposed by Smith, who claims it is possible, "to make changes in the product being developed or in the process by which it is developed, even relatively late in development, without being too disruptive" (Smith, 2008: 35). The key point is to work with the Agile method. The Agile manifesto promotes

the development of software in loops by verifying software requirements at the end of each iteration, with the possibility of re-planning them. In this context, a set of small teams of programmers work in collaboration, and the use of catalogs for software reuse is an important element.

Finally, success when facing changes to a product depends on the experience of each of the team's members. Experience in the field of software construction and the implementation of solutions for real cases is a relevant factor when dealing with unexpected changes, in particular those which are related to business logic. Taking into account the recommendations given by Smith and the strategies proposed by Sommerville, it is clear that incremental development is a good way to alleviate the negative impact of external changes.

3.1.10. Front Controller Design Pattern for Data Mining

The case of MCP's Entry Order Module

Pattern Name:

Font Controller

Description:

A design pattern is widely seen as the solution for certain problems related to software production (Sommerville, 2011: 190). It is suggested that web applications use this pattern, which requires all requests made of the application to go through a *Front Controller Servlet*. This servlet receives the user's requests in order to process the

information via a *Page Controller* which uses a certain predefined model; this model produces output in the form of graphical representations, illustrated by user controls in a web browser.

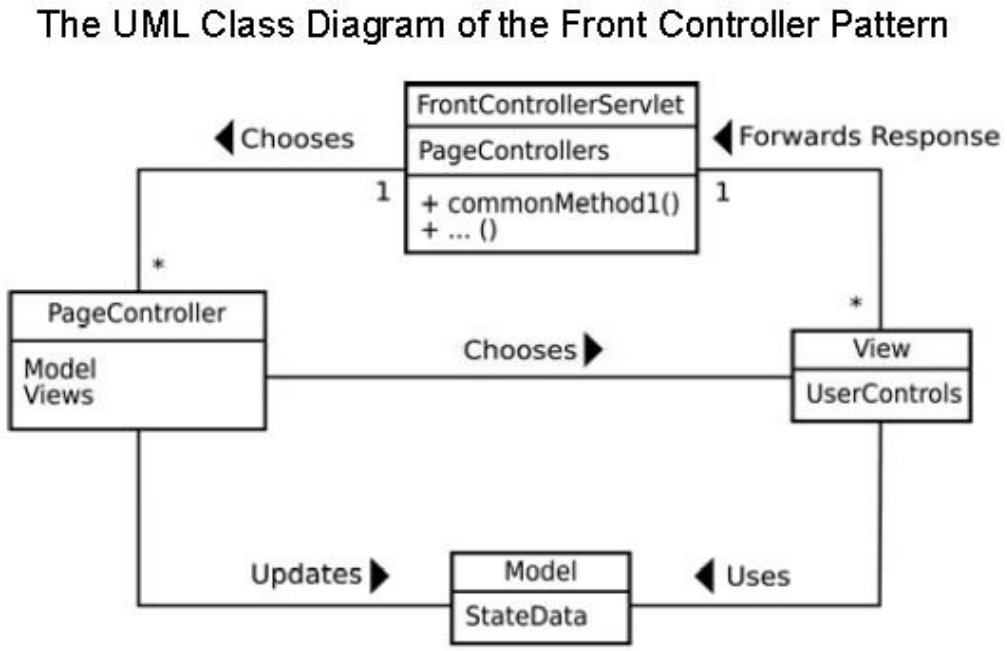


Figure 3.8 The UML Class Diagram of the Front Controller Pattern (Mouratidou et al, 2010: 29).

Problem Description:

In the particular case of using data mining techniques to find non-trivial information in the data warehouse, the business intelligence layer has a group of tools to represent the information retrieved in a way that is intelligible to the user.

The logic of data mining lies in complex processing units which act as searchers; indeed, the core of the data mining technique is based on good algorithms for association, clustering, decision trees, mathematical regressions, time series, Bayesian networks, and

neural networks, among others. These algorithms represent a huge effort in terms of computing power and lots of data are required.

This is the essence of the problem, because end users are “light” clients within the information system, and the majority use mobile devices when consuming data mining resources. As such, end users are often unable to process information on a large scale. Obviously this scenario is placed in a client-server model.

Solution Description:

This solution can be addressed by a re-interpretation of the actors and classes that play a role in the data mining process. In a simplified format, the Front Controller Design Pattern can be interpreted in a simplified way as follows:

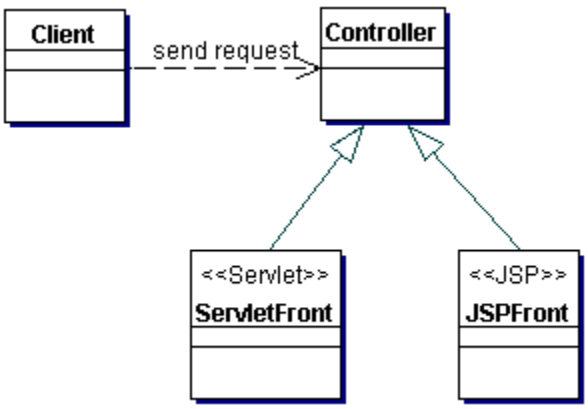


Figure 3.9 Simplified representation of a Front Controller Design Pattern

According to the above design, a Controller delegates the hard work to the class *ServletFront*. This class is responsible for the interaction with the DBMS engine using the Data Mining Services. It is important to note that a servlet is an application that runs on a server consuming its computing resources (processing time, memory allocation, storage,

etc.). In this way, a request for non-trivial information from a client can be processed in accordance with the following sequence diagram:

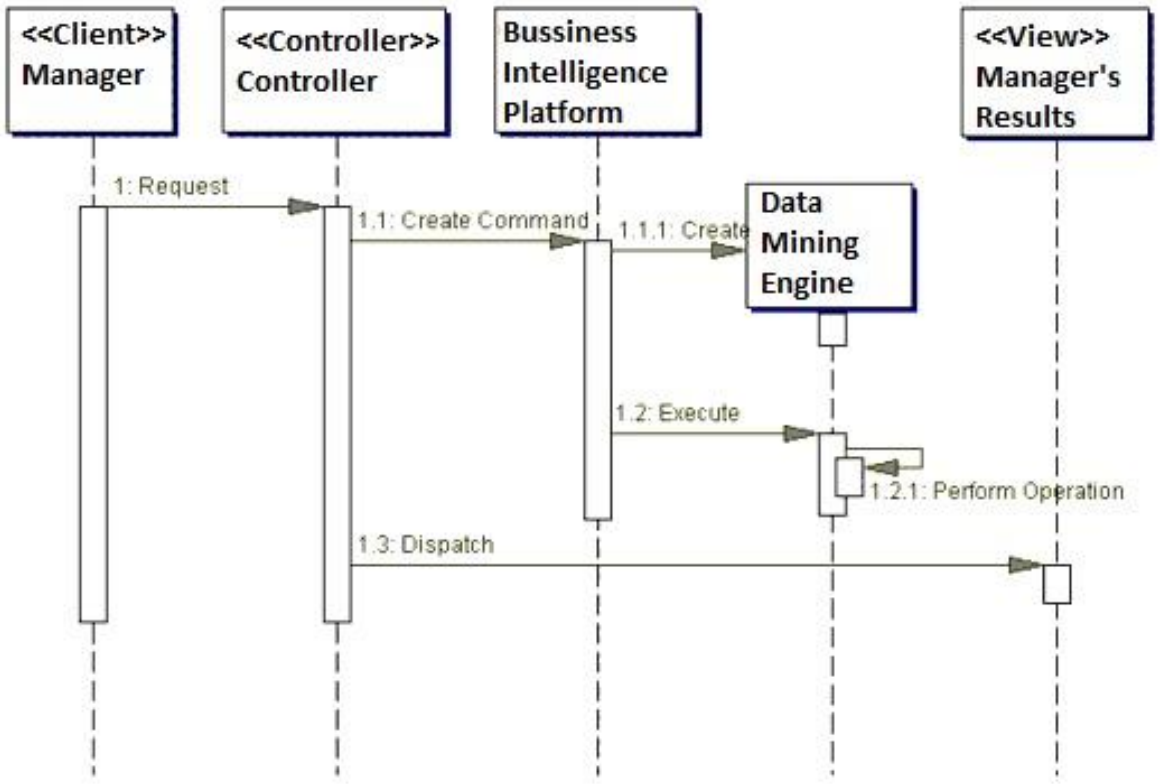


Figure 3.10 Solution based on the Front Controller Pattern's sequence diagram

Consequences:

In this way, the scenarios are independent, the places for requesting and representing information for end users, and the place where all this information is constructed, are clearly defined. There are separate entities which can be located within the three tier paradigm: presentation, business logic, and data access.

The success of this implementation is based on the excellence of the design and deployment functions at the data base (and data warehouse) level. The platform for data

mining must work perfectly, with the highest standards in terms of quality and performance.

3.1.11. Measuring a Software Process

Positivism, the school of thought relating to empirical sciences, is entirely objective. One of the main features of the positivist approach lies in the quantification of knowledge, objects of study and the systematization of their research methodologies (i.e. the usage of the scientific method with a quantitative approach). Within this paradigm are the “exact” sciences and disciplines of knowledge that are related to applied science, like engineering.

Therefore, disciplines related to software engineering use the objective approach in order to measure all data associated with software process quantitatively (Sommerville, 2011: 711). The end goal is to determine how efficient a software process is in accordance with the quality of its products versus the customers’ expectations. Several factors are considered in the measurement process, including: time, costs, resources, people efforts, technical parameters, and quality indicators, among others.

The measurement of a software process generates evidence concerning how experienced an organization is in constructing software-based solutions (Sommerville, 2011: 714). A common way of obtaining the information required is by using questionnaires and interviews, in conjunction with general guidance from a sound methodology, such as SEI’s CMMI. A point of interest is that CMMI and ISO 9001 can be

used in unison when goals are defined using the ISO method, and are measured using CMMI (Baldassarre et al, 2012: 309). This combination would give a more thorough understanding of processes and products' quality.

Reliable measurement techniques are available in both CMMI and ISO models, specifically in CMMI-DEV and ISO/IEC 15504. However, new approaches such as A^2QPM (Assessment Approach for Quantitative Process Management) incorporate most of the essential parts of previous approaches, from both a process and measurement perspective (Tarhan & Demirors, 2012: 79).

The Assessment Approach for Quantitative Process Management (A^2QPM) is a brand new approach that helps to identify software processes' measurements from a quantitative perspective, in order to classify various elements under clearly defined terms, such as: purposes, processes, measures, and efforts. A^2QPM looks for relevant evidence in the software process, based on CMMI and ISO models and in addition, A^2QPM's measurement perspective is based on identifying, and data management in terms of existence, verifiability, dependability, normalizability, and integrality (Tarhan & Demirors, 2012: 79).

According to the authors, this new approach is easier to implement in companies than its predecessors, due to its simple language; as stated by them, "the effort required to utilize the A2QPM is relatively small" (Tarhan & Demirors, 2012: 77). In general, this new approach is for any company that develops software, no matter the size or methodology. In fact, most of them use Agile methodologies to produce their software-based solutions. The process involved in applying it is shown in the figure below:

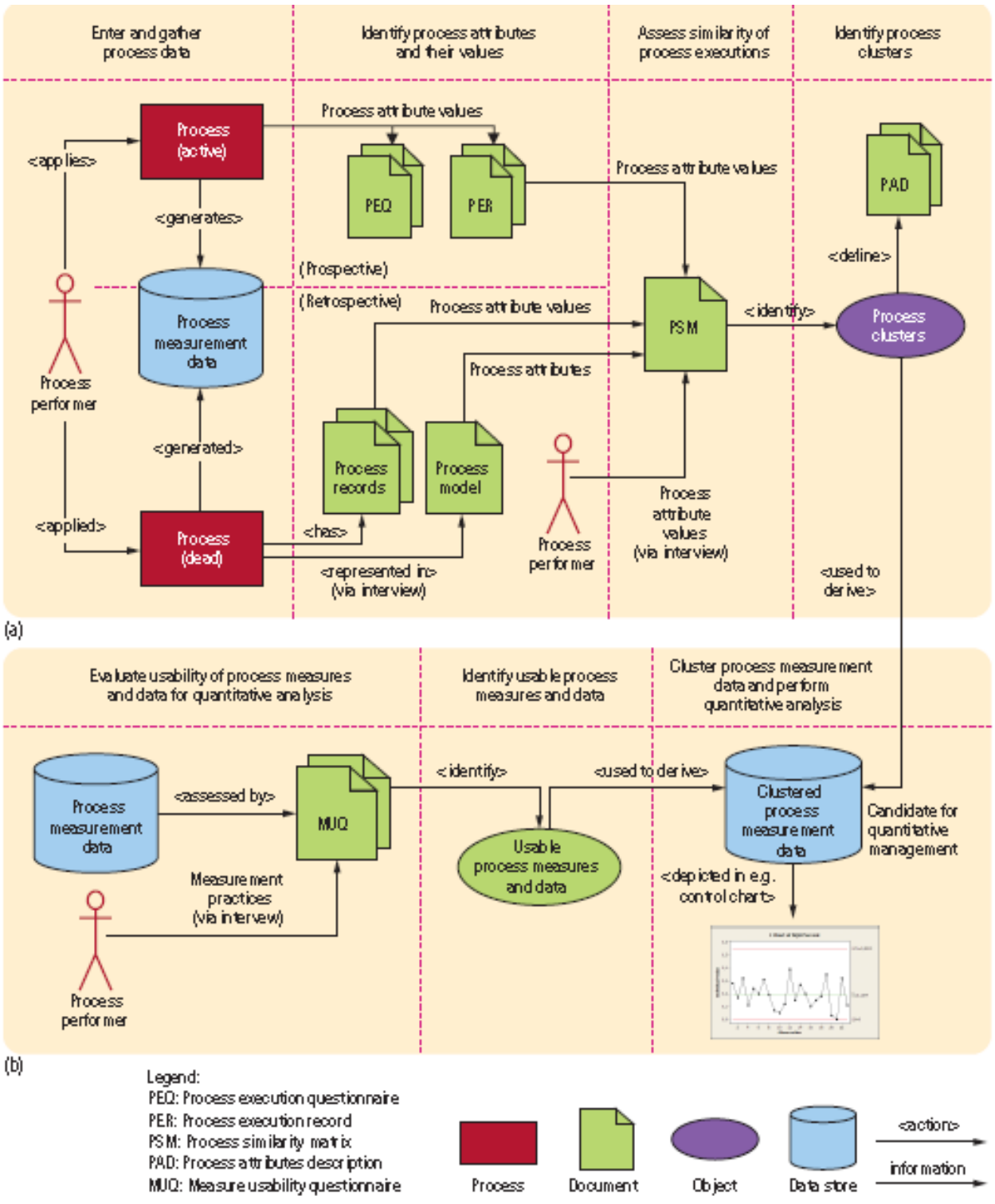


Figure 3.11 A^{2}QPM Assessment model (Tarhan & Demirors, 2012: 78).

In conclusion, new approaches to measuring the current status of a company in terms of their software production are welcome, since those approaches focus on identifying the potential of these companies. In spite of sound infrastructure, good practices and reliable methodologies in the software construction field, what still matters most is skilled staff and a good atmosphere. This key factor is not so easy to measure, as a qualitative approach is required.

3.1.12. Programmers' Skills

Good practice in computer programming is a key factor in the construction of dependable software (Sommerville, 2011: 356). It is just as essential as a sound technical background in the field; however, another, more specific but equally essential asset is experience in constructing complex software solutions.

Having good programming practices is essential to constructing complex software, because it is difficult, perhaps impossible, to face the challenges associated therein without proper programming techniques. Therefore, complex programming requires many of the same skills essential throughout the technical arena; nonetheless, there are additional abilities that should be taken into consideration.

The additional programming skills needed in order to create complex and dependable software go beyond those of the technical field. These assets are related to attitude, values and the promotion of a healthy work environment. These skills can be categorized as technical, business, and soft skills (Bailey & Mitchell, 2007: 28).

In this integrated scenario, Bailey and Mitchell identify a list of skills and their respective types (TS as technical skills, BS as business skills, and SS as soft skills) as seen in the table below:

Rank	Description	Type
1	Ability to read, understand and modify programs written by others	TS
2	Ability to code programs	TS
3	Ability to debug software	TS
4	Listening skills	SS
5	Problem-solving process (decision tree, problem identification & analysis)	SS
6	Team work skills (long term)	SS
7	Knowledge of structured programming fundamentals	TS
8	Ability to implement programs	TS
9	Knowledge of multiple programming languages	TS
10	Ability to visualize/conceptualize	SS
11	Time management skills	SS
12	Adaptability to new technologies & languages	SS
13	Ability to apply knowledge	SS
14	Ability to read design specifications for conversion to code	TS
15	Verbal communication skills	SS
16	Ability to read technical documentation	TS
17	Ability to design software programs	TS
18	Ability to write clear documentation	TS
19	Knowledge of design methodologies	TS
20	Ability to design user friendly applications	TS
21	Ability to give and receive constructive criticism	SS
22	Ability to research language syntax	TS
23	Ability to multitask	SS

Table 3.2 Integrated Programmers' skills (Bailey & Mitchell, 2007: 31).

In this section, the focus is on skills related to attitudes, behaviors and values that are important in a work environment, the "Soft Skills". It is well known that the team work abilities are a common denominator. This aptitude incorporates listening capabilities,

attitudes to problem solving, time management, good communication in the workplace, respect, and the ability to accept constructive criticism.

These soft skills can be summarized under three main headings: good labor relations, commitment, and sense of belonging. As good interpersonal relations are the basis of any society, it follows that the establishment of good communication channels in the workplace is the first step to achieving a positive work environment. This is an important point, as the construction of complex software generally implies the participation of more than one programmer.

To create the right climate for constructing complex software, the programmers' commitment and their sense of belonging to the company or organization are paramount. Commitment might be interpreted as the degree of input given by the programmer to his/her project or job. In this case, the ideal situation that in which people really enjoy what they do and are therefore willing to product their best work. Unfortunately, this is not always the case.

A sense of belonging is related to the construction of a collective identity in business contexts; this forms part of many organizational theories for creating a good work environment. A sense of belonging can be seen as a feature that transforms people from simple employees to valuable human resources. Without doubt, motivation is key in both of the aforementioned scenarios.

In conclusion, the idea of looking for integrated skills when hiring programmers to construct complex software solutions is a sound one. These skills can be summarized as technical skills, good labor relations, commitment, and sense of belonging. The presence

of all the cited skills can be verified though well-designed interviews; however, strong preference must be given to proven experience in the field.

3.1.13. Software Engineering in the 21st Century

As with all applied sciences, computing and its related disciplines have evolved quickly; and as such, people expect faster, more robust, and more scalable applications to solve specific problems. This requires the constant upgrade of hardware and software (Andriole, 2008: 27). Firstly, Software Engineering is seen as a discipline in the field of engineering that incorporates all aspects of professional software production (Sommerville, 2011: 7).

Since the beginning of computing, the concept of software has been associated with computer programs. The construction of software was a difficult task which could only have been carried out by a few people (experts and gurus at that time). Obviously, the increasing complexity in software construction has led to serious reflections about how to construct software in response to current and future needs. Thus, Software Engineering was born.

First, software creators were concerned with the implementation phase; for long time therefore, the main role in software creation was that of the computer programmers. In the 1970s and early 1980s, computer programmers measured their work by counting how many code lines they wrote; this was the Structured Programming paradigm. Over time, system requirements became more complex, as did people's expectations of new

software products. In terms of design and programming, the Object-Oriented paradigm then began to emerge.

At the same time, multicore architectures in hardware processors, increasing storage capabilities, and the large main memory are features that can and must be utilized by software development suites. Development environments which employ powerful programming languages such as Java, C#, and Ruby are the new ingredients in the construction of complex solutions for universal use (Blackburn et al, 2008: 83).

Through the "relatively new" approaches associated with the design/implementation phases of software construction, such as patterns and frameworks, computer programming is becoming an increasingly trivial task. For instance: in the past, computer programmers had to write entire code lines for every single device, including the graphic interface, the pointing module (i.e. a mouse), the network interface, etc. Today's programmers just invoke reusable objects in a pre-conceived graphical scenario (nobody is interested in how to program a mouse pointer, as it simply already exists in a graphic user interface).

As such, the curricula of computer science have experienced drastic changes in computer programming. Not long ago, lots of code lines were required to construct software; today, design skills are more relevant than the programming abilities. There is now a significant difference between creating complex software systems and writing lines of code; Software Engineering explores the former environment, whilst computer programming is just a phase in a software process.

If it seems that computer programming, once so revered, is becoming trivial, what does the future hold for Software Engineering in the 21st century? Leon Osterweil, professor at the Department of Computer Science at the University of Massachusetts Amherst, stated that there is a pressure in the field of Software Configuration Management, Software Product Lines, Software Test Automation, and Software Development Environments, to construct more and more complex systems with billions of implicit code lines for reuse; especially as computing is becoming ever more omnipresent (IEEE Computer Society, 2007: 6).

It seems that the future challenges in software construction could be confronted with strengths in the field of software design; the use of reusable assets will lead us to the construction of complex software systems. Entire models as software templates will be adapted to produce dynamic replications of computer-based solutions. In addition, the main concern about this "forecast" is related to security, scalability, and the final product's quality (IEEE Computer Society, 2007: 9).

Furthermore, another interesting aspect associated with the Software Engineering discipline is the contribution of Middleware, both as technology and methodology. Valerie Issarny, Mauro Caporuscio, and Nikolaos Georgantas explain how the design of Middleware –as a software layer in the integration of networked systems, applications, and even people– promotes pervasive computing models. Their proposal is based on the concept of SOM (Service-Oriented Middleware) for spaces in grid, cloud, and ubiquitous computing (IEEE Computer Society, 2007: 244).

At the same time, Alan Dearle –professor of Computer Science at The University of St Andrews– stated that current technologies such as Java and .Net, and models such as OMG, CCM and D & C will evolve to face future challenges; the trend in computing is one of the virtualization of environments and service-oriented paradigms. Finally, he supports the idea that Middleware is a key factor in the integration of complex systems in ubiquitous scenarios (IEEE Computer Society, 2007: 273).

In conclusion, Software Engineering will be centered on models and designs; future professionals in the field will have specific skills in this matter, rather than a simple skills in computer programming.

3.1.14. Risks to take into account in Software Projects

As with all projects in any area or field of study, software construction projects have risks, and as consequence, processes and activities associated with software engineering must take into account the factors that may affect the completion of planned objectives. This is where risk management for software construction comes in. Commonly, studies concerning potential risks are done by project managers. The essence of these studies lies in the ability to anticipate situations which might alter the expected course of software construction. Risks represent a threat to software projects, and can be classified in three main categories: project risks, product risks, and business risks (Sommerville, 2011: 596).

As an example, three situations have been detailed within the context of a software construction project which are considered risks: requirements change, size underestimate, and staff turnover. These risks are shown in the following table.

Risk	Description and Impacts	Control actions
Requirements change	This risk is present when new requirements are demanded by users, or when the current requirements are altered. This risk has direct incidence on the overall schedule depending on the complexity of those changes.	The use of the best practices in requirements engineering is a key factor in anticipating possible changes. The phase is associated with requirements identification and their analysis should generate clear documentation with flexible design models; taking into account future additions and modifications.
Size underestimate	This risk is present when there are evident failures in the requirement identification and analysis phase. In this phase, the people in charge of designing and developing software solutions could face more complex implementation processes than expected because some features were not taken into account at the beginning, giving a false idea about the real size of the solution. This situation can generate extreme changes in both the design and implementation phases, which imply costs and delays.	Thorough documentation in the requirements phases derived from exhaustive analysis is the suggested solution. The application of criteria such as “Divide and Conquer” in the early stages of a software process can be useful at the moment of determining the real size of a computing solution.

Staff turnover	Everyone is different, and those participating in the construction of complex software systems can leave their mark, depending on their skills and knowledge. Changes to staff can also generate costs and delays in programmed schedules. It is difficult to continue with the “logic and habits” of someone else in the middle of a software construction process.	A clear policy concerning contracts for the staff could help avoid changes. The ideal situation is when the entire staff is present from the beginning to the end.

Table 3.3 Risks, impacts and actions

In conclusion, the above are some examples of risks that can occur in software construction. Those risks must be prevented if possible, and controlled if they happen. This is feasible if the team is adaptable and ready to address unexpected situations.

3.1.15. Software Requirements Document: A spell-checker

Preface:

This document describes the general requirements of a spell-checker available in a software component (plug-in). This component shall be embedded in web browsers to enable spell-checking tasks. This is the first version of the software component, with the possibility of an upgrade to a new version each year.

Introduction:

This spell-check plug-in is a software component to complement writing functions in web browsers to improve the users' grammar and syntax. This plug-in is embedded in a web browser as a software extension; as such, the spell-check function is available whilst a user is navigating online through his/her web browser. Any text field is capable of being verified so long as it includes written information. In addition, this plug-in includes multi-culture and globalization features, which are important for international users who browse in multiple languages.

Glossary:

Plug-in: software component that is embedded in a complex software solution to assist specific functions.

Web Browser: End User's application that allows navigation in the World Wide Web service.

Context Menu: An alternative set of options that are available by using the menu generated by right clicking the mouse.

User Requirement Definition:

1. Users download the plug-in according to their OS and web browser.
2. Users install the plug-in as an extension of their web browser.
3. Users repair the installation of the plug-in.
4. Users uninstall the plug-in.
5. Users select and download their own language.

6. Users feed their local lexicon.
7. Users add auto-correction rules.
8. Users check spelling and grammar.
9. Users provide feedback.

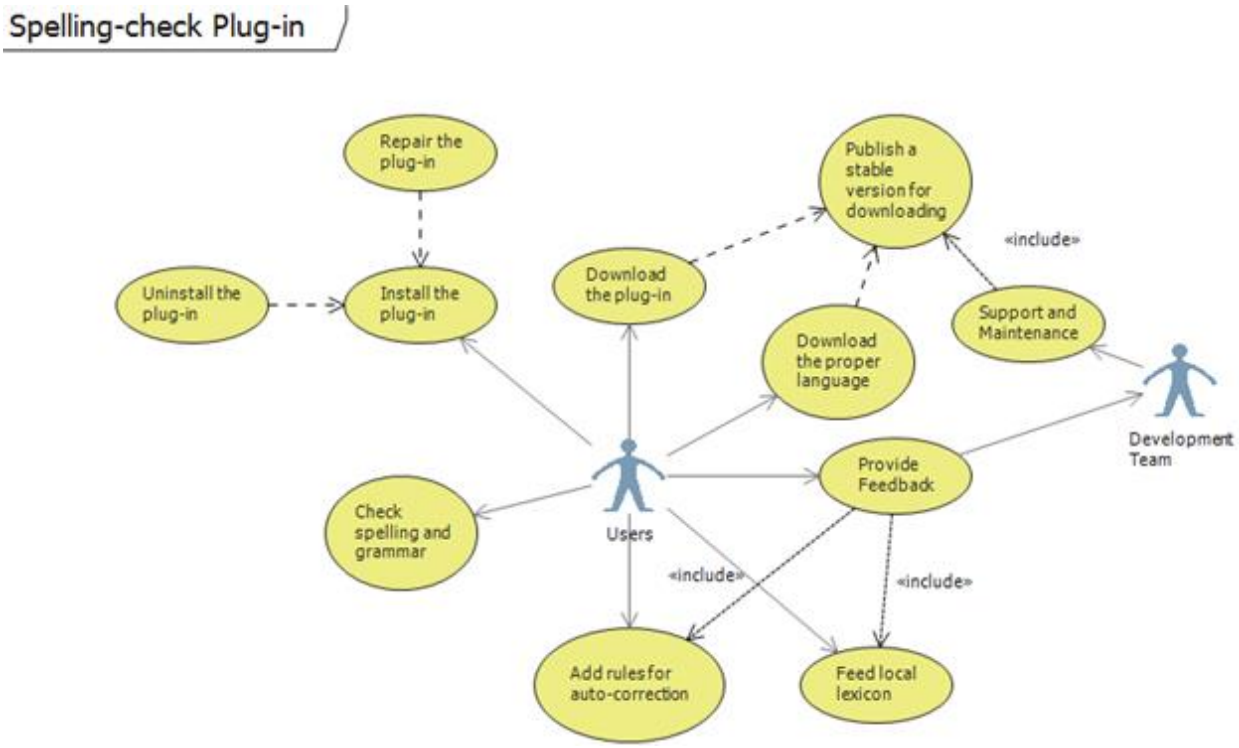


Figure 3.12 Use case for spelling-check Plug-in

System Requirements Specification:

Functional Requirements	Non-functional Requirements
Download, Install, Repair, and Uninstall the plug-in. The software component shall be available on the Internet for download; as well as a variety of languages chosen by the end user. In a local machine, end users can configure the plug-ins' behavior in a specific operating system and web browser.	• Speed: The plug-in shall work with a response-time of milliseconds; this means that the spell-check functionality works whilst users type sentences within a web browser's content (a web page). • Size: the plug-in is not larger than 10Mb. The grammar rules and basic lexicon according to the language are consumed through XML web services • Easy to Use: this plug-in is very intuitive; wizards are available at the moment of installation. It is located in a context menu on screen. • Reliability: There is an official web site for online support. • Portability: the plug-in is available for several OSs and web browsers, running in almost 50 languages.
The plug-in will recognize all written text in a web-based interface; according to the local language, it will check the syntax and grammar of written text within the web browser. The user can add some auto-correction rules and feed the local lexicon in order to enrich the vocabulary. Then, those actions related with the system upgrade can be associated with feedback processes for the development team.	

Table 3.4 System requirements specification

3.1.16. Systems without SOA

Within the context of distributed systems, Service-Oriented Architecture (SOA) is a way of constructing information systems using a distributed approach (Sommerville, 2011: 509). Each service runs independently of the entire system, and can be consumed by software with different types of implementation. SOA is based on XML and is used to envelop data using standard protocols such as SOAP, WSDL, and UDDI, among others (W3C, 2004).

SOA has several benefits, including easy adaptation for businesses' needs, portability in information transport, and scalability in its capacity for growth (Bartholomew, 2007: 23). SOA exploits the potential of network-based solutions where web services are reusable software units within a complex software construction.

It stands to reason that SOA has a strong dependency on network connection conditions; indeed, according to the World Wide Web Consortium, the deployment of SOA protocols needs HTTP. The main protocol, called SOAP, is a layer that works over the HTTP infrastructure. In this vein, Internet access is required.

In some scenarios, this dependency on a reliable Internet connection is an advantage, as it largely promotes the consumption of web services, in environments where the proper infrastructure is available. However, not all contexts have the right conditions. Consequently, some systems are not suitable for SOA due to their Internet connection. Two examples are outlined below.

A system for monitoring a natural phenomenon in extreme conditions. In this case, some systems need to capture data about a natural phenomenon such as volcanic activity, population monitoring for species in the middle of a jungle, and the tide's behavior in the middle of the ocean. For these situations, technological adaptations are based on embedded systems within units specially designed to work in extreme conditions; in fact, it is common to use local storage mechanisms in order to register data for further analysis according to the monitoring parameters, due to the lack of reliable Internet connection.

A system that needs to work in real-time. In spite of the availability of a reliable Internet connection, the "postback" phenomenon is ever present, based on the duality of

request-response. When an application consumes a web service within the parameters of SOA, the application must send a request to the web service provider; then that provider receives the request in order for it to process it, and for it to be transmitted as a response to the application. This communication requires time (sometimes seconds or fractions of seconds). For some systems which do not require an immediate response, this “postback” phenomenon is acceptable. However, for other systems which need to work in real time, a second is a long time. In this particular; for some systems, real time means lapses of time measured in milliseconds or even in microseconds. For instance, those systems which enable medical diagnosis or serve mechanic actuators.

3.1.17. The Evolution of Legacy Systems

For Ian Sommerville (2011: 525), a legacy system is old software that runs in an organization. In computing scenarios, the term “old” in terms of software development can be defined as the classic period of monolithic developments, when the number of code lines ruled all. Programming paradigms, such as procedural development, were used to construct giant software units within organizations.

Software produced in the aforementioned circumstances has not been easy to maintain (Sommerville, 2011: 526). It has also been difficult to improve, due to the resistance to change of the end users, and the lack of updates on theoretical and practical knowledge on software development.

Some end users reject new developments; they are familiar with the old system's user interfaces and have mechanized its usage. Perhaps there is a fear that changing the way a system works will have negative consequences. It is possible that these people feel threatened in their jobs as using new user interfaces can lead to lower productivity while users become accustomed to the new format.

On the other hand, lack of experience in new software development approaches is another factor that contributes to problems in the software development team. Those in charge of the software construction do not always update their knowledge and programming skills. This is of critical importance, because these individuals should be the source of motivation and initiative when it comes to improving the system.

It is fairly difficult to establish a satisfactory solution because the context implies attitudinal and behavioral aspects. However, education is always a possible solution.

3.1.18. The Replacement of Software Components

Component-Based Software Development (CBSD) is designed to produce scalable software systems through the reuse of components built explicitly for that purpose. In this context, CBSD's purposes can be summarized as certain concepts, especially the component concept. A component is a reusable unit of deployment and composition. The common view, is that a component is closely related to an object, and the CBSD is therefore an extension of object-oriented development (Crnkovic et al, 2002).

One of the main features of CBSD is related to component independence. In this regard, independence is defined as the quality of components which work within a well-

defined environment of collaboration between software units. Communication between software components takes place via parameterized interfaces (Sommerville, 2011: 453).

When there is component independence, it is possible to replace one component with another without altering the functionality of the whole system. This is the ideal from a theoretical perspective; however, in real life this can only be achieved in certain circumstances.

In general terms, a software component is seen as a “black box” that solves a specific problem. The interfaces that connect to the “black box” work under well-defined parameters and must be well documented, to facilitate the consumption of a component in reuse environments. This can be a problem, depending on how the component was developed, and whether reusability was then taken into account.

Based on the principles of reusability, a component can be developed independently. This is a threat to the proper functionality of a component for reuse, because this situation generates failures in three major areas: parameter incompatibility, operation incompatibility, and operation incompleteness (Sommerville, 2011: 469).

When corporations or companies are using CBSD, the construction of a brand new software-based solution involves the use of prefabricated pieces. However, it is possible to find development teams –even individuals– that work at different times, interacting with different people or stakeholders, and perhaps with different ideas on their minds (Iribarne, Troya, & Vallecillo, 2004: 342). This can cause some degree of incompatibility in developed software components.

In order to avoid the problems associated with component incompatibility, some practices have been suggested by Floch, among others. These good practices include: well-defined component interfaces and compositions by using a UML-based approach, use an ontology-based approach for specifying the components' goals, and the usage of strong incremental validations techniques to reduce issues related with incompatibility (Floch et al, 2010: 1760).

In conclusion, according to the evidence presented, it appears that the usage of Component-Based Software Development requires a clearly defined software process, which is supported by solid documentation. The question is how Agile methodologies can integrate CBSD, because Agile methodologies look for rapid software development, but without extensive documentation.

3.1.19. The software process at the University of Nariño

The information system in a public university is quite complex. Strategic information systems change goals, operations, products and relationships within the environment of the institution to help them gain a competitive edge. Often information systems change within the institution and as do the products, services, and procedures which lead to new patterns of behavior. Such changes often require new administrators, a new labor force and a much closer relationship with customers and suppliers.

The existing applications at the University of Nariño provide a flow of information about activities at an operational level, transport and storage of data, and generation of information by way of reports. The applications developed are part of five major

subsystems: Academic subsystem, Financial subsystem, Human Resource Subsystem, Physical Resources Subsystem, and Document Management Subsystem. The configuration is shown in figure 3.13.

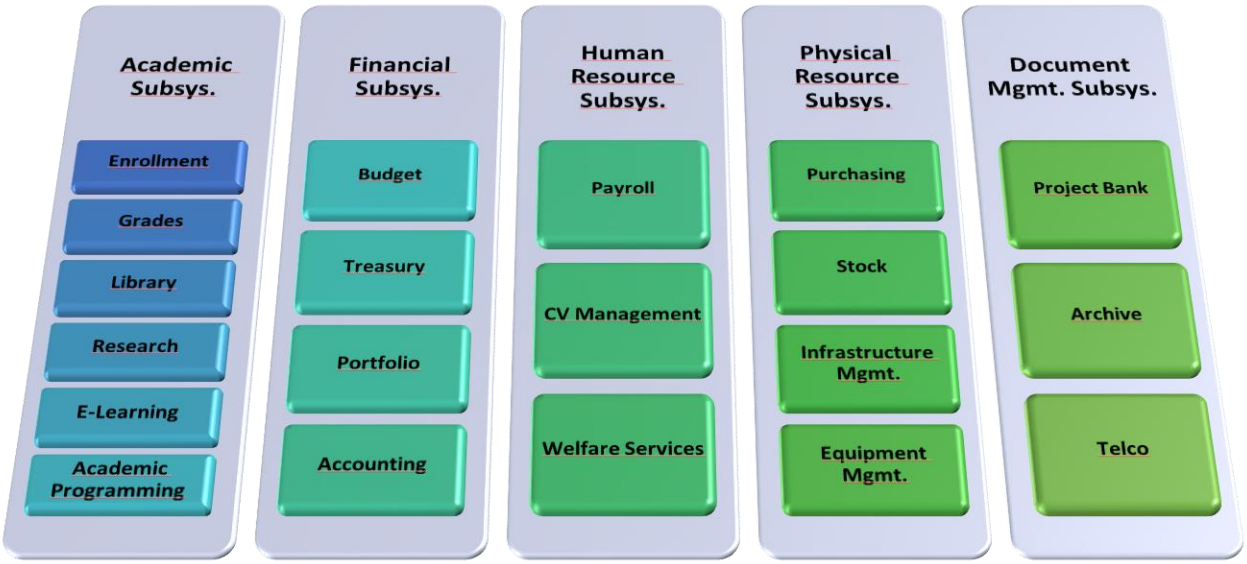


Figure 3.13 Integrated Information System of Universidad de Nariño (Insuasti et al, 2008).

In today's organization there is a growing interdependence between strategy, rules, business procedures and information systems (software, hardware, databases and telecommunications). A change to any one of these components often requires changes in others. This relationship becomes critical when management plans for the future. What an organization would like to be doing in five years often depends on what their systems can do. In this sense, and taking into account the process of university reform, the proposal suggests the implementation of some changes to the administrative and

academic structure of the University of Nariño in favor of enhancing the Integrated Information System and its relations with the academic mission system of the Alma Mater. Thus, presented below are three main points to be addressed: 1. The creation of the Office of Information Systems. 2. The Computer Services Unit for the Academy. 3. The New Systems Committee.

As waterfall models are so rigorous (Sommerville, 2007), all software processes are based on this approach. However, it is important to mention that some documentation is always missed. In this vein, all the strict stages of the software process are present in the construction of software application at the university. However, the greatest weakness is the lack of standards at the moment of documenting the process.

3.1.20. Open Source Development

Software construction has been around since the beginning of computing itself. Paradoxically, early computing solutions based on the mainframe approach distributed source code; over time, computer programs became more complex and their code was closed progressively. This is how proprietary code was born.

Today, both free software and open source movements have a strong presence in the computing world. Indeed, it was necessary to create a brand new software process in accordance with the open source philosophy. Open source development is seen as an approach to software construction in which final products conform to principles, such as: free redistribution of source code without restrictions, and free use of source code without charge, among others (Open Source Initiative, n.d.).

Unfortunately, the terms “free software” and “open source initiative” are sometimes used indistinctly. According to the president of the Free Software Foundation: “The two terms describe almost the same category of software, but they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement” (Stallman, n.d.). Stallman defines open source as a software development approach. It stands to reason that any software development methodology has its own pros and cons, and open source development is no exception.

One of the main concerns about open source development is related to its reliability, due to the massive participation of different contributors in software construction. The wide participation of developers is done via the Internet (Sommerville, 2011: 198). This situation generates some trust issues, a common question, as stated by Paul B. de Laat, is “how do people handle the problem of trusting that those strangers have good intentions and adequate competence?” (Laat, 2010: 327). In order to answer that question, the author identified some mechanisms used to handle the problem of trust, as show in in figure 3.14.

	Open-source software
Assumption of trust (substantial trust)	
Inference of trust (weak form)	Hacker ethic
Inference of trust (strong form)	Past performance
	Entry examinations
Substitution of trust (from a small to a large extent)	Design (from high-discretion to low-discretion)

Figure 3.14 Mechanisms for handling the problem of trusting developers (Laat, 2010: 340)

According to the findings of Laat's research, there is no formal mechanism for gaining substantial trust in software construction. In other words, good intentions and adequate competence are always uncertain. The concept associated with "substantial trust" is based on the assumption of deep knowledge and the presence of the strong experience in the field. The truth is that in these scenarios, which count on the participation of several (even hundreds) of invisible developers in an open source project, the first step is to start in the assumption of good faith (Laat, 2010: 333).

However, it is important to note that some "radical" developers work under the open source approach, they often call themselves hackers. In this case, Laat raises the following question: What kind of ethics do these hackers have?

Ken Thompson, in his famous Turing Award lecture, *Reflection on Trusting Trust*, stated controversially that, "you can't trust code that you did not totally create yourself". In spite of the context in which he made his point, in fact, it was in 1983, it has a strong impact on the open source initiative, as more was discovered about backdoor attacks (Kamp, 2011: 1). This is clearly another factor that creates an uncertain environment when using an open source approach.

Nowadays, academics and scientists recognize open source initiative as a good way to create spaces for sustainable software development. The initiative has sound benefits including low production costs, freedom, inclusion, and non-discrimination policies, among others. However, this does not negate the fact that using open source products is a matter of trust, which is entirely personal.

3.1.21. Timing Requirements in Real-Time Systems

Real-time systems are able to complete tasks in “real time”, they require no response time, and are present in almost all aspects of people’s daily lives; they are in the inner parts of aircrafts, mobile phones, automobiles, medical equipment, entertainment systems, and home appliances, to name but a few (Sommerville, 2011: 538). In order to interact “instantaneously” with the physical world real-time systems integrate several advanced engineering systems, which perform control functions, which gives them a close relationship with embedded systems.

However, real-time is a relative concept, depending on the system requirements and the user’s needs. For some systems, real-time could mean a response time of a couple of seconds; this occurs in systems in which a reasonable delay is acceptable. For instance, a vehicle tracking system via GPS does not need to establish the position of a vehicle several times per second. In this case, a new position signal, giving the real location of a moving vehicle, can be received after a few seconds delay without affecting the functionality of the entire system.

In contrast, other systems have an instantaneous response time. For instance, the control systems in an aircraft must capture data, process them, and act accordingly in milliseconds or microseconds. Generally speaking, applications which are directly responsible for people’s welfare or safety are based on embedded systems working in real-time, as missing a deadline could create a system failure or threaten human life (Lee et al, 2007: 135). Entertainment and communication systems also use systems with

instant response times; fast responses are essential, especially when multimedia is involved.

Schedulability analysis examines the timing requirements of real-time systems (Lopez, Cuevas & Drake, 2012: 257). Schedulability analysis can be represented using UML sequence diagrams, which are useful when illustrating timelines and delays. UML sequence diagrams represent time in rows in pointed lines, an example is shown in figure 3.15.

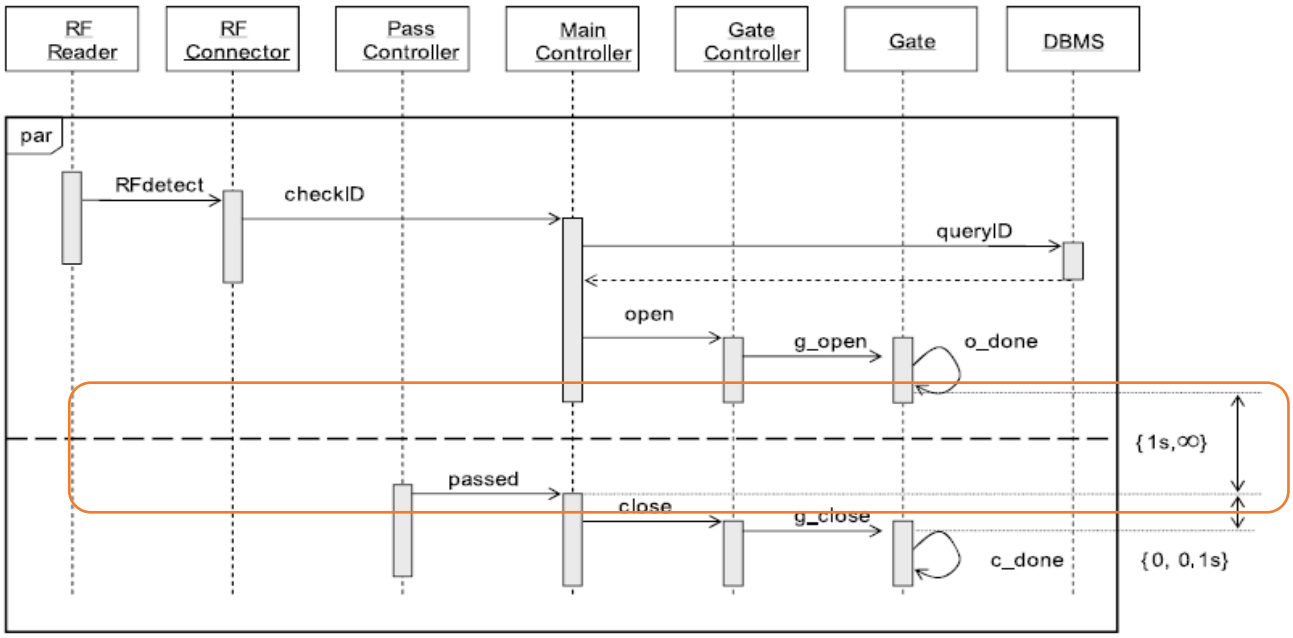


Figure 3.15 Example of a real-time system –automatic door in parking– (Lee et al, 2007: 142).

In terms of systems design, the set of UML sequence diagrams is translated into MTER nets. An MTER (Modular TER) net is a formal extension of a TER (Time Environmental Relationship) net which is a useful tool for showing a system's response

time (Lee et al, 2007: 137). These nets are based on petri nets oriented to timing constraints. In this vein, the transformation process from UML sequence diagrams to MTER nets means the entire timing consumption in a real-time system can be established. Such a transformation is shown in figure 3.16.

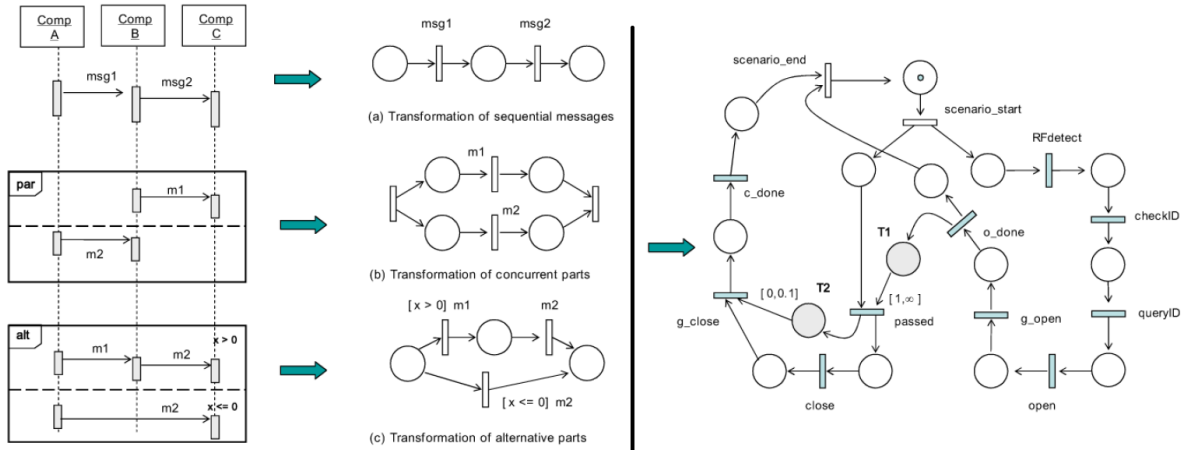


Figure 3.16 Transformation process from UML sequence diagrams to MTER nets (Lee et al, 2007: 146).

As seen above, there are tools and resources available to accurately illustrate timing in software/systems scenarios. The critical question is how can time parameters be fixed when creating deadlines. In engineering terms, time parameters are set after a comprehensive requirement analysis which includes a profound study of the physical phenomenon to control or measure. Time parameters are provided within the disciplinary knowledge of the phenomenon or event to be measured or controlled.

3.1.22. Two Sides of the Coin in Cloud Computing

Software engineering is defined as a discipline of engineering that involves all aspects of professional software production (Sommerville, 2011: 7). Under this definition, it stands to reason that professional software construction includes different phases, regardless of which software process or methodology is employed. An emerging technology used specifically in the programming and deployment phases is gathering more followers: this is Cloud Computing.

By definition, Cloud Computing is a complex composite technology and infrastructure that allows the deployment of mission-critical, scalable and robust applications within virtualized environments (Baun et al, 2011: 3). Although this technology is not entirely new, current software production is led by SaaS, PaaS, and IaaS approaches (Software as a Service, Platform as a Service, and Infrastructure as a Service approaches) in which virtualization plays a transcendental role. It has been said that, “resource virtualization is at the heart of most cloud architectures” (Baun et al, 2011: 5).

Looking first at the benefits of cloud computing, its impact has been extensive in several areas: performance, application management, computing resource management, data distribution, and scalability, among others (IEEE, 2011). These benefits are explained briefly as follows:

Performance on the Cloud: The ability to use simultaneous computing resources, such as CPU processing, memory allocation and storage, means high performance. There are of course certain variables: number of concurrent users, size of data to be

processed, and the good practice in application development, among others. Nonetheless, it makes cloud computing particularly attractive to those who are looking for super-computation environments, as well as companies with thousands or millions of users.

Application management on the Cloud: Given that cloud computing environments are related to SaaS, it is easier to deploy applications designed and programmed for web-based scenarios. In order to control the behavior of a deployed application, some extra elements such as language descriptors based on RIF –Rule Interchange Format– may be needed (Moran, Vaquero & Galan, 2011: 89).

Computing resources management/Scalability on the Cloud: This is associated with the IaaS approach; Virtual Machines are created and configured automatically as required. In virtualized environments, this feature allows resource pool management to be automated. By applying resource reservation methods, it is possible to increase workload performance (Ye et al, 2011: 267). In this way, depending on the physical infrastructure, virtualized environments can be highly scalable as well as their respective applications.

Data Distribution on the Cloud: Within the SaaS paradigm, SOA leads on design and development of web-based applications which become web services. An application –independent of its nature– can consume data from different sources according to the web services. This is the holy grail of ubiquitous computing, in which data are distributed along the cloud, and are available anytime, anywhere.

Above are only some of the benefits of using cloud computing technology. There are many other benefits associated with Cloud Computing solutions, such as lower energy consumption and less space required, due to virtualization.

However, there are concerns surrounding cloud computing, associated with ownership and control. There is the idea that a cloud-based solution is not a good idea because of the risks associated with not having control over the administration of the deployment infrastructure. In particular, this concern arises when people are dealing with mission-critical software systems, when software systems work with sensitive information.

In conclusion, cloud computing will go arguably continue to develop with new features and services. There could be an increase in its use for new generations of software designers and programmers. However, all technologies have requirements, and a major requisite of cloud computing is a good and reliable Internet connection; and unfortunately, that is not yet universally available.

3.2. WORKSHOPS

3.2.1. Workshop 1

1. Classes of Telephone, DVR, Printer, and Bank Account

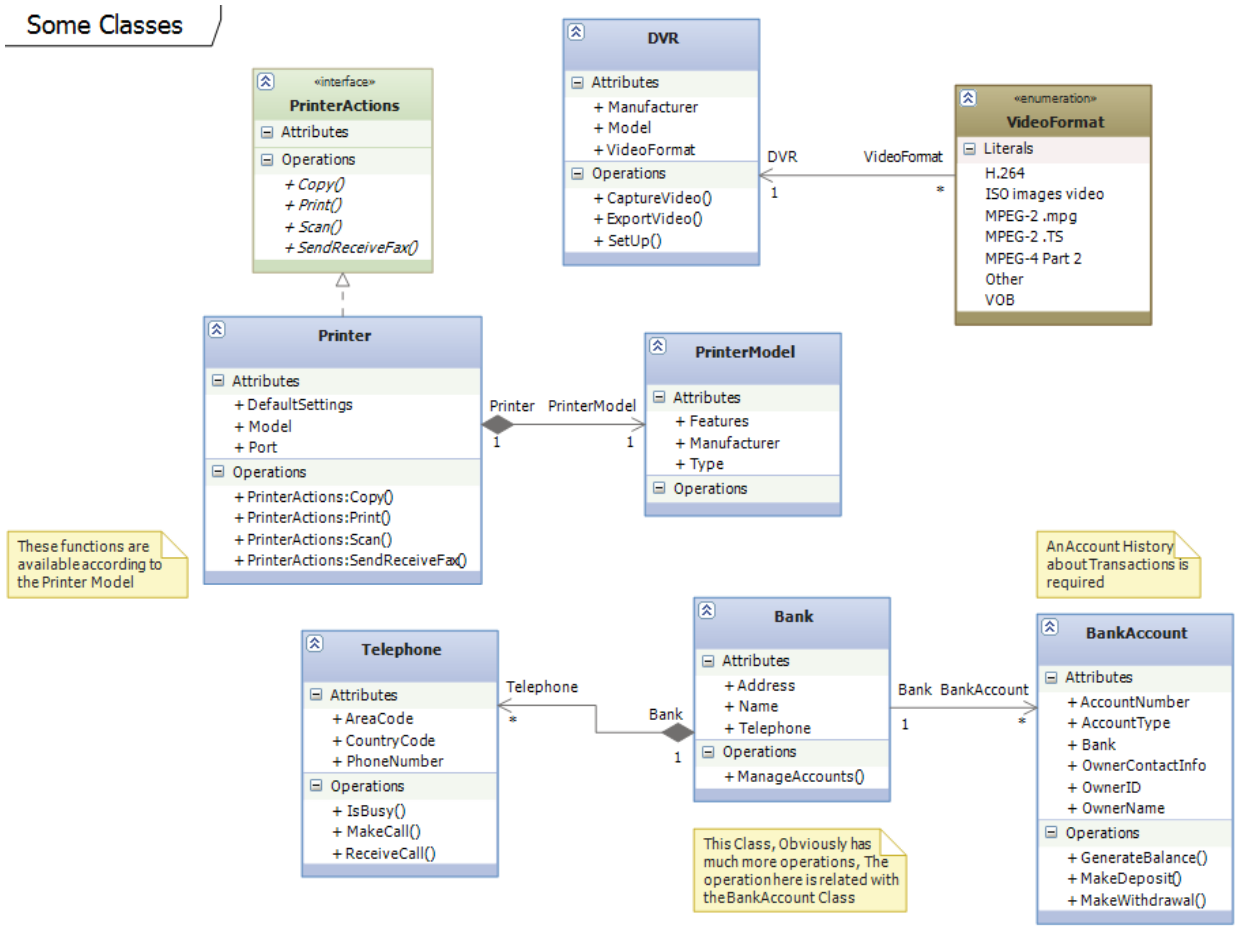


Figure 3.17 Some classes

2. Risks related with a medical system based on radiation

Regarding complex solutions, risks are classified in three categories: project risks, product risks, and business risks (Sommerville, 2011: 596). Taking as example a medical

system that works with radiation to treat tumors, the system is controlled by specialized software with the following possible product risks:

- Miscalculation of dose in delivering radiation. Too much radiation worsens the condition of the tumor and affects the patient's general health; whereas, too little radiation would not have any bearing on treatment.
- Sensitive information can be exposed due to security failures. Information related to people's health status must be protected under the principles of confidentiality and privacy (United Nations, n.d.). Security failures allow unauthorized people to get access to private information.

Medical system software must have strong mechanisms to safeguard the above risks, because people's lives and privacy are at stake. These mechanisms include:

- Real-time Audit with a warning messaging system. This feature will allow constant monitoring of the system's functionality. In addition, the system will have the ability to self-adjust the radiation doses in real-time; these actions must be reported via instant messaging systems to the system's operators.
- Rigorous security policies. The operation of the system, and the management of the information generated are possible using reliable authentication and authorization techniques. All communication channels should be secured to avoid data leaking. In addition, information must be strongly encrypted in order to guarantee confidentiality.

3.2.2. Workshop 2 (Software requirements Template)

2xxx

X System-Y Subsystem

Software Requirements Document

Your Name Here

Your Name Here
Date Here

Preface

Version History

Version	Date	Author	Sections Worked On

Distribution List

Name	Position	Role

Intended Audiences

References

Table of Contents

BIBLIOGRAPHY

- Adobe Systems Inc. (2012). 'Adobe Dreamweaver CS5.5', *Adobe Home Page*, [Online] Available at: http://www.adobe.com/products/dreamweaver.html, (Retrieved on: February 4th, 2012)
- Ambler, S (2005). A Manager's Introduction to The Rational Unified Process (RUP), Ambysoft, Toronto, Canada.
- Andriole, S. (2008). 'Technology Curriculum for the Early 21st Century', *Communications of the ACM*, July 2008, 51(7), pp.27-30.
- Apple, Inc. (2011). 'HTTP Live Streaming', [Online] Available at: http://tools.ietf.org/html/draft-pantos-http-live-streaming-07, (Retrieved on January 7th, 2012)
- Bailey, J., & Mitchell, R. (2007). 'Industry Perceptions of the Competencies needed by Computer Programmers: Technical, Business, and Soft Skills', *Journal of Computer Information Systems*, Winter 2007, 47(2), pp28-33.
- Baldassarre, M. et al (2012). 'Harmonization of ISO/IEC 9001:2000 and CMMI-DEV: from a theoretical comparison to a real case application', *Software Quality Journal*, June 2012, 20(2), pp.309-335.
- Bartholomew, D. (2007). 'THE PROS AND CONS OF SOA', *Baseline*, November 2007, 78(1), pp.22-24.
- Baun, C. et al. (2011). Cloud computing [electronic book]: web-based dynamic IT services, Springer, New York, USA.

- Behr, J., & Jung, J. (2010). 'X3DOM: Getting declarative (X) 3D into HTML' ,*TPAC 2010 - W3C Technical Plenary / Advisory Committee Meetings Week*, Web 3D Consortium, [Online] Available at: www.w3.org/2010/11/TPAC/HTML5-X3D-Graphics-Demo.pdf, (Retrieved on: January 28th, 2012)
- Bender, J., & McWherter, J. (2011). Professional test-driven development with C#: developing real world applications with TDD, Indianapolis, Ind.: Wiley, 327 p.
- Bennet, S., McRobb, S., & Farmer, R. (2010). Object-Oriented Systems Analysis and Design Using UML. 4th Edition, McGraw-Hill Higher Education, UK.
- Bentley, R. (2009). 'Put yourself on the map', *Caterer & Hotelkeeper*, July 2009, 199(4588), p30-31, 2p.
- Berners-Lee, T., & Connolly, D. (1993). 'Hypertext Markup Language (HTML): A Representation of Textual Information and MetaInformation for Retrieval and Interchange, 1st Draft', *CERN*, Genève, Switzerland, [Online] Available at: http://www.w3.org/MarkUp /draft-ietf-iiir-html-01.txt, (Retrieved on: February 11st, 2012).
- Bertolino, A., & Mirandola, R. (2003). 'Towards Component-Based Software Performance Engineering', In (Eds.) Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction,
- Blackburn, S. et al (2008). 'Wake Up and Smell the Coffee: Evaluation Methodology for the 21st Century', *Communications of the ACM*, August 2008, 51(8), pp.83-89, doi:10.1145/1378704.1378723.

- Bohem, B., & Papaccio, P. (1988). 'Understanding and Controlling Software Costs', *IEEE Transactions on Software Engineering - TSE*, 14(10), pp. 1462-1477, DOI: 10.1109/32.6191
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide – 2^{dn} Edition, Pearson Education, Upper Saddle River (NJ), USA.
- Bothelo, S. (2011). 'HTML5 vs Native Apps: Where Should Publisher Focus?', *Folio the Magazine*, December 2011, p.9.
- Braverman, B. (2006). 'The Filmmaker's Codec', *Digital Content Producer*, August 2006, 32(8), pp.14-18.
- Briton, C., & Doake, J. (2005). "6: Identifying functionality. CRC cards and interaction diagrams", *Chapter 6 In: A Student Guide To Object-Oriented Development*, pp. 147-180, ScienceDirect, EBSCOhost, DOI: 10.1016/B978-075066123-2/50006-2 , (Retrieved on: October 20th, 2012).
- Cappel, J., & Zhenyu, H. (2007). 'A USABILITY ANALYSIS OF COMPANY WEBSITES', *Journal of Computer Information Systems*, Fall 2007, 48(1), p117-123.
- Carnaval de Negros y Blancos' corporation – CORPOCARNAVAL (2009). Nomination for inscription on the Representative List in 2009, Some Cultural Expressions, Pasto Colombia.
- Castrianni, L. et al (2010). 'High resolution satellite ortho-images for archaeological research: different methods and experiences in the Near and Middle East', Advances in Geosciences 2010, 24, p97-110.

- Channel 5 Broadcasting Ltd, (n.d.). 'Colour in Snails', [Online] Available at: http://milkshakecdn.channel5.com/assets/image_file/000/001/311/standard_Fifi_background.jpg?1320925702, (Retrieved on January 16th, 2012)
- Chapman N., & Chapman, J. (2009). Digital Multimedia 3rd Edition, John Wiley and Sons Ltd., USA.
- Chapman, N. & Chapman, J. (2009). Digital Multimedia, 3rd Edition, John Wiley & Sons, Ltd, England.
- Chapman, N., & Chapman, J. (2009). Digital Multimedia 3rd Edition, John Wiley and Sons Ltd., UK.
- Chapman, N., & Chapman, J. (2009). Digital Multimedia 3rd Edition, John Wiley and Sons Ltd., USA.
- Chapman, N., & Chapman, J. (2009). Digital Multimedia, John Wiley and Sons Ltd. England.
- Clune, R., et al. (2012). "An object-oriented architecture for extensible structural design software", *Computers and Structures*, 100-101, pp. 1-17, ScienceDirect, EBSCOhost, doi:10.1016/j.compstruc.2012.02.002 (Retrieved on: October 24th, 2012).
- Connolly, D. (2000). 'A Little History of the World Wide Web', *W3C*, [Online] Available at: http://www.w3.org/History.html, (Retrieved on: May 5th, 2012).
- CORPOCARNAVAL (2012). 'Carnaval de Negros y Blancos', *CORPOCARNAVAL home page*, [Online] Available at: http://www.carnavaldepasto.org/sitio/, (Retrieved on: February 29th, 2012)

- Crnkovic, I. et al (2002). Building reliable component-based software systems, Artech House Computing Library, Norwood (MA), USA.
- Cugnini, A. (2011). 'Video Compression Technology: Web browsing and broadcasting are closer than ever', *Transition to Digital – Digital Handbook (Broadcast Engineering)*, pp.22-23.
- Devaraj, E. et al (2011). 'Predicting the software performance during feasibility study', *IET Software*, April 2011, 5(2), pp.201-215, DOI: 10.1049/iet-sen.2010.0075.
- Eggleston, R. (n.d.). For the Birds, Pixar Animation Studios, [Online] Available at: http://www.youtube.com/watch?v=MOiyD26cJ2A&feature=related, (Retrieved on: January 28th, 2012)
- Egypt Sons (n.d.). South West Road, [Online] Available at: http://shannonalexandraspier. blogspot.com/2011/05/paved-roads.html, (Retrieved on January 20th, 2012)
- Eiffel Software (n.d.). 'The Power of Design by Contact', Eiffel Software's Home Page, [Online] Available at: http://designbycontract.com/developers/design_by_contract.html, (Retrieved on: November 24th, 2012).
- Floch, J. et al (2010). 'A comprehensive engineering framework for guaranteeing component compatibility', *The Journal of Systems & Software*, 2010, 83(10), pp.1759-1779, DOI: 10.1016/j.jss.2010.04.075.
- Fowler, M. (1999). Refactoring: Improving the Design of Existing Code, Addison-Wesley Longman Inc., Westford (MA), USA.

- Gall, M., Grechenig, T., & Bjerre, M. (2011). 'Assessing the Feasibility of Developing a Federated ERP System', *International Journal of Managing Information Technology (IJMIT)*, August 2011, 3(3), pp.16-26.
- Google (2012). 'Adding map features', Google Maps home page, [Online] Available at: http://support.google.com/maps/bin/answer.py?hl=en&answer=144363&ctx=cb&src=cb&cbid=-lgpq07qphie0, (Retrieved on: February 24th, 2012)
- Harmes, R., & Diaz, D. (2008). "Encapsulation and Information Hiding", In: Pro JavaScript Design Patterns, doi: 10.1007/978-1-4302-0496-1_3, p. 25.
- Havaldar, P., & Medioni, G. (2009). Multimedia Systems: Algorithms, Standards, and Industry Practices, Course Technology, 1st Edition, Boston (MA), USA.
- Hvam, L., Riis, J., & Hansen, B. L. (2003). "CRC cards for product modelling", *Computers in Industry*, 50(1), January 2003, pp. 57-70, DOI: 10.1016/S0166-3615(02)00143-4.
- IANA (2007). 'MIME Media Types', *Internet Assigned Numbers Authority*, [Online] Available at: http://www.iana.org/assignments/media-types/index.html, (Retrieved on January 7th, 2012).
- IBM (2007). RUP for Asset-Based Development V3.0 and Asset-Based Development Governance Plug-in V1.0, IBM Developer Works, [Online] Available at: http://www.ibm.com/developerworks/rational/downloads/07/rup_abd_gov/, (Retrieved on October 23th, 2011).

- IEEE (2011). 4th IEEE International Conference on Cloud Computing (CLOUD), [electronic book] Piscataway: Institute of Electrical and Electronics Engineers, 10.1109/CLOUD.2011.2, Danvers (MA), USA.
- IEEE Computer Society (2007). Future of Software Engineering, 2007. FOSE '07 [electronic book] Piscataway: Institute of Electrical and Electronics Engineers, Minneapolis (MN), USA.
- IEEE Computer Society (2010). IEEE Standard for Information Technology – System and Software Life Cycle Processes – Reuse Processes, IEEE Standard 1517™ - 2010, New York, USA.
- Insuasti, J. et al (2008). 'Propuesta del Sistema de Información Integrado', Universidad de Nariño Press, Language: Spanish, Pasto, Colombia.
- Insuasti, J., Jaramillo, N., Rodriguez, H., & Timaran, R. (2009). 'Integrated Information System', *Universidad de Nariño – Reform 2009-2020*, Udenar Press, Pasto - Colombia
- Iribarne, L., Troya, J.., & Vallecillo, A. (2004). 'A Trading Service for COTS Components', *Computer Journal*, 2004, 47(3), pp.342-357
- James, J. (2009). 'Cloud Computing: Silver lining or storm clouds ahead? ', *Windows IT Pro*, January 2009, 15(1), pp.3-3.
- Kaur, K., & Singh, H. (2012). 'An Investigation of Design Level Class Cohesion Metrics', *International Arab Journal of Information Technology (IAJIT)*, January 2012, 9(1), pp.66-73.
- Kay, R. (2006). 'MIME'. *Computerworld*, June 2006, 40(24), p42-42.

- Kemper, A. (2009). Valuation of network effects in software markets [electronic book] : a complex networks approach / Heidelberg : Physica, DOI: 10.1007/978-3-7908-2367-7.
- Kolakowski, N. (2011). 'Microsoft's Bing Adds Airport Maps', *eWeek*, October 2011, 28(17), p7-7.
- Kruchten, P. (1995). 'The 4+1 View Model of Architecture', *IEEE Software*, November 1995, 12(6), pp.42-50, DOI:10.1109/52.469759.
- Kruchten, P. (2004). The Rational Unified Process 3rd Edition: An Introduction. Reading, MA: Addison-Wesley Longman, Inc.
- Krueger, C. (1992). 'Software Reuse', ACM Computing Surveys, June 1992, 24(2), pp.131-183.
- Krzysztof J. (2007). Data Mining [electronic book]: A Knowledge Discovery Approach. Springer Science+Business Media, Boston (MA), USA.
- Laakso, S. (2003). 'User Interface Design Patterns', User Interface Design Patterns' Home Page, [Online] Available at: http://www.cs.helsinki.fi/u/salaakso/patterns/, (Retrieved on: November 9th, 2012)
- Laudon, K., Laudon, J. (2002). Sistema de Información Gerencial. Editorial Prentice-Hall, 6th Edition México D.F., México.
- Laureate Online Education (n.d.). 'Subject: System Analysis', *Material of the course (week4) Systems Analysis and Design using an Object-Oriented Approach*, [Online] Available at: https://our.ohecampus.com/secure/login.aspx, (The University of Liverpool – Laureate Online Education). (Retrieved on: October 25th, 2012) pp. 1-21.

- Lee, H. et al (2007). 'Specification and analysis of timing requirements for real-time systems in the CBD approach', *REAL-TIME SYSTEMS*; July 2007, 36(1-2), pp.135-158, DOI: 10.1007/s11241-007-9017.
- Lim, S. Quercia, D., & Finkelstein, A. (2010). 'StakeNet: Using Social Networks to Analyse the Stakeholders of Large-Scale Software Projects', *ICSE: International Conference on Software Engineering*, pp.295-304.
- Lin, P., Lee, J., & Chang, C. (2011). 'Protecting the Content Integrity of Digital Imagery with Fidelity Preservation', *ACM Transactions on Multimedia Computing, Communications & Applications*, August 2011, 7(3), p15:1-15:20, 20p; DOI: 10.1145/2000486.2000489
- Lopez, P., Cuevas, C., & Drake, J. (2012). 'Compositional real-time models', *Journal of Systems Architecture*, June-August 2012, 58(6-7), pp.257-276, DOI: 10.1016/j.sysarc.2012.04.001.
- Madeyski, L. (2010). Test-Driven Development: An Empirical Evaluation of Agile Practice, Heidelberg; Springer-Verlag, The University of Liverpool Catalogue, EBSCOhost, viewed 24 November 2012.
- Mat, R. et al. (2011). 'Online 3D Terrain Visualization of GIS Data: A Comparison between Three Different Web Servers', *Pertanika Journal of Science & Technology*, October 2011 Supplement, 19, p31-39
- Mathew, D. (2010). HTML5: Designing Rich Web Applications, *Elsevier Inc.*, USA, [Online] Available at: http://www.sciencedirect.com.ezproxy.liv.ac.uk/science/book/9780240813288, (Retrieved on: February 11st, 2012).

- Matisse Software Inc. (2012). 'Matisse: The post-relational SQL database', Matisse's Home Page, [Online] Available at: http://www.matisse.com/, (Retrieved on: November 10th, 2012).
- Meyer, B. (1991). 'Design by Contract', in *Advances on Object-Oriented Software Engineering*, Mandoli, D., & Meyer, B. (Eds), London: Prentice-Hall.
- Microsoft Corp. (2003). 'ASP.NET Authentication', Microsoft Developer Network MSDN, [Online] Available at: http://msdn.microsoft.com/en-us/library/aa291347(v=VS.71).aspx, (Retrieved on: June 19th, 2012).
- Microsoft Corp. (2011). '.NET Windows Azure Introduction', *Microsoft's Channel 9 – Microsoft Developer Network*, [Online] Available at: http://channel9.msdn.com/Blogs/Windows-Azure-Developer-Experience-Videos/NET-Windows-Azure-Introduction, (Retrieved on: May 5th, 2012).
- Microsoft Corp. (2012). 'Bing Maps Silverlight Control SDK', Download Center – Microsoft Home Page, [Online] Available at: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=2949, (Retrieved on: February 24th, 2012)
- Microsoft Corp. (2012). 'Microsoft Expression Web 4', *Microsoft Home Page*, [Online] Available at: http://www.microsoft.com/expression/products/Web_Overview.aspx, (Retrieved on: February 4th, 2012)
- Microsoft Corp. (2012). 'Microsoft Live Movie Maker 2011', *Microsoft Home Page*, [Online] Available at: http://explore.live.com/windows-live-essentials-movie-maker-get-started?os=other, (Retrieved on: February 1st, 2012)

- Miyake, Y., Suzaki, K., & Araya, S. (2009). 'A Web page that provides map-based interfaces for VRML/X3D', *Electronics & Communications in Japan*, February 2009, 92(2), p28-37, DOI: 10.1002/ecj.10017.
- Moran, D., Vaquero, L., & Galan, F. (2011). 'Elastically Ruling the Cloud: Specifying Application's Behavior in Federated Clouds', *4th IEEE International Conference on Cloud Computing (CLOUD)*, pp.89–96, DOI: 10.1109/CLOUD.2011.53.
- Mouratidou, M. et al (2010). 'An Assessment of Design Patterns' Influence on a Java-based E-Commerce Application', Journal *of Theoretical & Applied Electronic Commerce Research*, 2010, 5(1), pp.25-38.
- MSDN (n.d.). 'Model-View-Controller', *Microsoft Developer Network's Home Page – Patterns and Practices Developer Center*, [Online] Available at: http://msdn.microsoft.com/en-us/library/ff649643.aspx, (Retrieved on: November 14th, 2012)
- MSDN (n.d.). 'Supported Media Formats, Protocols, and Log Fields', Microsoft Development Network, [Online] Available at: http://msdn.microsoft.com/en-us/library/cc189080(v=vs.95).aspx, (Retrieved on: February 15th, 2012)
- Nazareth, D., & Rothenberger, M. (2004). 'Assessing the cost-effectiveness of software reuse: A model for planned reuse', In Applications of statistics in software engineering, *The Journal of Systems & Software*. 2004, 73(2), pp.245-255 DOI: 10.1016/S0164-1212(03)00248-6
- ODBMS.org (n.d.). 'Introduction to ODBMS', Object Database management Systems' Home Page, [Online] Available at: http://www.odbms.org/Introduction/, (Retrieved on: November 10th, 2012).

- O'Leary, L. (2007). 'Silverlight Brings Speedy Video to the Web', *PC World*, October 2007, 25(10), pp.28-28.
- Pathak, P. (2010). 'Image Compression Algorithms for Fingerprint System', *International Journal of Computer Science Issues* (IJCSI), July 2010, 7(4), p45-50.
- Pau, V., Mihailescu, M., & Stanescu, O. (2010). 'Model View Presenter Design Pattern', *Journal of Computer Science & Control Systems*, 3(1), pp.173-176.
- Rajesh, K. (2011). 'Location Intelligence Mashup Using Open Source Software and Google Maps API', *IUP Journal of Information Technology*, March 2011, 7(1), p35-46, 12p.
- Rajput, H., & Singh, L. (2011). 'Improvement of Software Quality Attributes in Object Oriented Analysis and Design Phase Using Goal-Question-Metric Paradigm', *Journal of Software Engineering & Applications*, June 2011, 4(6), pp.345-349, DOI: 10.4236/jsea.2011.46039.
- Reeds, K. (2004). 'When the botanist can't draw: the case of Linnaeus' *Interdisciplinary Science Reviews*, September 2004, 29(3), p248-258, DOI: 10.1179/030801804225018882.
- Roelofs, G. (2009). 'A Basic Introduction to PNG Features', *The LibPNG*, [Online] Available at: http://www.libpng.org/pub/png/pngintro.html, (Retrieved on January 7th, 2012).
- Rossel, G., & Manna, A. (2003). "Design by Contract: constructing reliable software", Revista Digital Universitaria, 5(5). UNAM, Mexico.

- Seffah, A., & Gaffar, A. (2006). 'Model-based user interface engineering with design patterns', *The Journal of Systems & Software*, 80(8), pp.1408-1422, DOI: 10.1016/j.jss.2006.10.037.
- Shull, F. (2012). 'A Brave New World of Testing? An Interview with Google's James Whittaker', *IEEE Software*, March 2012, 29(2), pp4-7, DOI: 10.1109/MS.2012.23.
- Silverstein, D., & Farrell, J. (2004). 'The Relationship between Image Fidelity and Image Quality', *Proceedings of IEEE International Conference on Image Processing*, [Online] Available at: http://scien.stanford.edu/jfsite/Papers/ImageQuality/FidQual.pdf, (Retrieved on January 21th, 2012).
- Smith, P. (2008). 'CHANGE: EMBRACE IT, DON'T DENY IT', *Research Technology Management*, July/August 2008, 51(4), pp.34-40.
- Sommerville, I. (2011). Software Engineering 9th Edition, Pearson/Addison-Wesley, Boston (MA), USA.
- Sommerville, I. (2011). Software Engineering, 9th Edition, Pearson Education / Addison-Wesley, Boston (MA), USA.
- Spanbauer, S. (2000). 'Audio Wars: WMA Tops MP3...Sometimes', *PC World*, October 2000, 18(10), p235, 2p.
- Steinmetz, R., & Nahrstedt, C. (2010). Multimedia Systems, Springer, Germany.
- Tarhan, A., & Demirors, O. (2012). 'Apply Quantitative Management Now', *IEEE Software*, May 2012, 29(3), pp.77-85, doi: 10.1109/MS.2011.91.

- Ullrich S. et al (2011). 'Quantizing the void: extending Web3D for space-filling haptic meshes', *Studies In Health Technology And Informatics [Stud Health Technol Inform],* IOS Press, 163, pp. 670-6; DOI:10.3233/978-1-60750-706-2-670.
- Unicode, Inc. (2012). 'What is Unicode', *Unicode Web Site*, [Online] Available at: http://www.unicode.org/standard/WhatIsUnicode.html, (Retrieved on January 7th, 2012).
- United Nations (2011). Disability and the Millennium Development Goals: A Review of the MDG Process and Strategies for Inclusion of Disability Issues in Millennium Development Goal Efforts, USA, [Online] Available at: http://www.un.org/disabilities/documents /review_of_disability_and_the_mdgs.pdf, (Retrieved on: February 17th, 2012)
- United Nations (n.d.). 'The Millennium Development Goals (MDGs) and Disability', *UN Enable – The MDG and Disability Portal*, [Online] Available at: http://www.un.org/disabilities/default.asp?id=1470#panel, (Retrieved on: February 17th, 2012)
- United Nations (n.d.). 'The Universal Declaration of Human Rights', *United Nations' Home Page, Article 12*, [Online] Available at: http://www.un.org/en/documents/udhr/. (Retrieved on: May 28th, 2012)
- Upson, J. et al (2012). 'Competitor Analysis and Foothold Moves", *Academy of Management Journal*, February 2012, 55(1), pp.93-110, DOI: 10.5465/amj.2008.0330
- Vasantha, R. et al (2008). 'Cost Estimation Model for Reuse Based Software Products', International MultiConference of Engineers & Computer Scientists 2008, pp.951-954.

- W3C (2004). 'Web Services Architecture', World Wide Web Consortium, [Online] Available at: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, (Retrieved on: June 15th, 2012).
- W3C (2011). 'Designing for Inclusion', Web Accessibility Initiative Home Page, [Online] Available at: http://www.w3.org/WAI/users/Overview.html, (Retrieved on: February 17th, 2012)
- W3C (2011). 'HTML5: A vocabulary and associated APIs for HTML and XHTML', W3C Working Draft 25 May 2011, World Wide Web Consortium Working Draft, [Online] Available at: http://www.w3.org/TR/html5/, (Retrieved on: February 11st, 2012).
- Webber, T., & Weins, B. (2009). 'Put Your Business on the Map With Google Maps', *Journal of Accountancy*, June 2009, 207(6), p44-48, 4p.
- Wells, D. (2009). "Extreme Programming: A Gentle Introduction", [Online] Available at: http://www.extremeprogramming.org/ , (Retrieved on: October 9th, 2012).
- Wirfs-Broke, R., Wilkerson, B., & Wiener, L. (1990). Designing Object-Oriented Software, Englewood Cliffs, NJ: Prentice-Hall International.
- Ye, K. et al (2011). 'Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments', *4th IEEE International Conference on Cloud Computing (CLOUD)*, pp.267-274, DOI: 10.1109/CLOUD.2011.69.