PROYECTO FIN DE CARRERA

Universidad Carlos III de Madrid

INGENIERÍA EN INFORMÁTICA

LA EVOLUCIÓN DE LOS PRODUCTOS SOFTWARE: APLICACIÓN DE REINGENIERÍA EN SISTEMAS DE INFORMACIÓN CORPORATIVOS

Autor:
D. Luis Miguel Esteban Santiago

Director:
Dr. D. Ricardo Colomo Palacios
(Departamento de Informática)

Fecha:
Abril, 2009

Yo creo en la suerte, y a lo largo de los años he aprendido que cuánto más trabajo, más suerte tengo.

Un ingeniero no es una copia, es original, y se atreve a cambiar una realidad, no importa el tiempo o el espacio, todo es posible mientras crea que es así.

Agradecimientos

En mi caso, no existe otra manera de comenzar este apartado sino dando las gracias a mi padre, mi madre y mi hermano. Sin su apoyo incondicional me hubiese resultado muy difícil conseguir las metas que me propuse. Siempre me enseñaron que el camino al éxito comienza con el recorrido de nuestro propio esfuerzo. Y gracias a ellos, he aprendido a esforzarme al máximo en cada momento para conseguir mi objetivo, lo que me lleva cada día a tratar de mejorar tanto personal, como laboralmente. Papá, Mamá, Rober, si soy como soy, es gracias a vosotros.

Y como no mencionar a mi novia, la cual ha sabido siempre escucharme y soportar mis desavenencias y duros momentos sufridos en ciertas etapas de mi carrera. Gracias a sus ideas y a su apoyo moral aportado hacia mi persona, aprendí que en este mundo triunfa quien se encarga de encontrar la solución a los problemas, no quien espera a que otros se los resuelvan. Debido a ello, mi modo de trabajo es cada día más ambicioso y animado, convirtiendo dicho trabajo en un reto, no en una obligación. Cris, gracias por ser como eres.

Tampoco podría olvidarme de mis amigos y mis compañeros de carrera, con los que tantas y tantas horas he pasado junto a ellos, realizando prácticas y estudiando. En muchas ocasiones nos ha tocado sufrir, pero eso nos ha hecho fuertes y ha creado entre nosotros lazos de amistad inquebrantables. Hemos aprendido a trabajar en equipo y, para mí lo más importante, no existe ocasión en la que no nos hayamos ayudado los unos a los otros. Gracias al sentimiento de saber que en todo momento podía contar con vuestra ayuda, por supuesto al igual que vosotros con la mía, habéis convertido mis logros en vuestros propios logros. Es una extensa lista la que tendría que nombrar aquí, pero sé que no es necesario, vosotros sabéis bien quienes sois porque todos continuáis a mi lado.

También quisiera reservar un apartado especial para mi tutor, Ricardo, porque además de abrirme las puertas al mundo laboral, desde el comienzo de este proyecto siempre ha estado disponible para atender mis dudas y me ha ayudado en todo momento. Conmigo se ha portado no sólo como un tutor, sino como un amigo, prestándome su cooperación tanto académicamente como a la hora de tomar decisiones laborales. Gracias a su experiencia y sus ideas, creo que aún podré seguir aprendiendo mucho de él.

Por último, en este apartado dedicado a los agradecimientos, no se me olvidará mencionar a mis compañeros de trabajo, con los que cuántas más horas paso, más me doy cuenta de que son como una gran familia para mí, con nuestros momentos de agobio y nuestros momentos de diversión, que no son pocos. Siempre dispuestos a arrimar el hombro los unos a los otros para que el resultado final de nuestro trabajo más que bueno, sea perfecto. Gracias por haberme acogido tan bien y depositar vuestra confianza en mí.

Resumen

A medida que las nuevas tecnologías evolucionan en una era basada en la informática y las telecomunicaciones, la lógica de negocio de las compañías corre el riesgo de quedarse obsoleta debido a las preferencias del mercado.

Los productos o servicios ofertados deben orientarse y especializarse cada vez más en función del tipo de cliente que requiere de ellos. Los clientes ya no se conforman con lo que encuentran, ya que actualmente tienen múltiples opciones para satisfacer sus necesidades. Hoy en día, la competencia, además de ser muy numerosa, es de clases muy distintas, por lo que muchas empresas necesitan realizar un cambio de mentalidad y dar un giro al modo de llevar su negocio.

La necesidad de las empresas de adaptarse a los cambios continuos en la forma de trabajo, así como de hacer frente a la presión ejercida por la competencia, otorga una importancia significativa al fenómeno de reingeniería.

En el presente proyecto se pretende mostrar como la reingeniería ayuda al desarrollo y evolución de los productos software de una empresa, sobre todo si se trata de un sistema amplio con gran complejidad. Para ello, se mostrará cómo aplicar un proceso de reingeniería del software en un sistema de información, basado en una migración del software. De este modo se obtendrá un sistema adaptado a las nuevas tecnologías que pueda enfrentarse a la competencia.

Abstract

As new technologies evolve in an era based on informatics and telecommunications, the business logic of the companies run the risk of becoming obsolete due to market preferences.

The products or services offered should be increasingly designed and specialized depending on the type of customer that requires them. Customers are no longer satisfied with what they find, as they currently have multiple options to meet their needs. Nowadays, rival companies, as well as being many, are very different kinds, so many companies need to change their mind and vary the way they run their business.

The need for companies to adapt themselves to continuous changes in the way of work and to cope with the pressure of competition, gives a significant importance to the phenomenon of reengineering.

The present project aims to show how reengineering helps to develop any company's software products, especially if it is a comprehensive system with great complexity. To do this, will show how putting into practice a software reengineering process in the system based on a technology migration plan. This will provide us with a system adapted to new technologies that may face competition.

Índice general

PROYECTO FIN DE CARRERA.....1

Agradecimientos 5

Resumen..... 7

Abstract 9

Índice general.....11

Índice de ilustraciones 15

Índice de tablas 17

Capítulo I. Introducción y objetivos. 19

1. Introducción. 21

1.1. Descripción del ámbito de estudio. 22

1.2. Problemática.....23

1.3. Delimitación de la solución. 23

1.4. Estructura de la memoria. 24

2. Objetivos.....27

Capítulo II. Estado del arte.....29

3. Reingeniería. 31

3.1. Introducción a la reingeniería.....31

3.2. Historia de la reingeniería.....31

3.3. Definición de reingeniería. 32

3.4. El porqué de la reingeniería. 34

4. Reingeniería del software. 37

4.1. Sistemas de información *legacy*. 38

4.2. Definición de reingeniería del software. 38

4.3. Procesos involucrados en la reingeniería del software. 39

4.4. Marco para la reingeniería del software. 42

4.5. Planificación de la migración de software. 44
4.6. Actividades para la planificación de la migración de software. 45
4.7. Directrices para la migración de software. 48
4.8. Conclusiones. 55
5. Sistemas de información corporativos. 57
5.1. El concepto de gestión de la información. 57
5.2. Definición de un sistema de información. 58
5.3. Sistemas de información en la empresa. 60
5.4. Sistemas de información gerenciales. 62
5.5. Sistemas para la gestión de recursos humanos. 63
5.6. Costes asociados a los sistemas de información. 65
5.7. Conclusiones. 67
Capítulo III. Descripción del problema. 69
6. Situación previa. 71
6.1. Consideraciones de la plataforma propiedad de la compañía. 71
6.2. Consideraciones de las etapas previas a este proyecto. 72
6.3. Consideraciones de la presente etapa de este proyecto. 74
7. Problemas identificados. 79
Capítulo IV. Descripción de la solución. 83
8. Gestión de la solución. 85
8.1. Framework adoptado. 85
9. Aspectos relacionados. 87
9.1. Calendario. 87
9.2. Presupuesto. 94
9.3. Tecnología. 95
9.4. Herramientas. 99
10. Desarrollo de la solución. 107
10.1. ¿Qué es WPF? 107
10.2. ¿Qué es XAML? 108

10.3. XAML y .NET. 110
10.4. El code-behind. 110
10.5. Ventajas de la combinación XAML y WPF. 111
10.6. Detalles previos a la solución. 115
10.7. Aspectos generales de la solución. 119
10.8. La solución: migración de la tecnología. 123
10.9. La solución: ejemplos de migración. 136
10.10. La solución: peculiaridades. 144

Capítulo V. Conclusiones y líneas futuras. 147

11. Conclusiones. 149
11.1. Conclusiones generales. 149
11.2. Conclusiones a los problemas identificados. 151
12. Líneas futuras. 155

Capítulo VI. Apéndices. 157

I. Apéndice A. URD. 159
II. Apéndice B. SSD. 181
III. Apéndice C. Plan de administración del proyecto software. 223
IV. Apéndice D. Manual de usuario. 249

Capítulo VII. Bibliografía. 265

Bibliografía. 267
Recursos electrónicos. 271

Índice de ilustraciones

Ilustración 1 - Modelo cíclico de la reingeniería del software....40
Ilustración 2 - Marco para la reingeniería....43
Ilustración 3 - Actividades para la planificación de la migración de software....45
Ilustración 4 - Funciones de un sistema de información....59
Ilustración 5 - Arquitectura de Información de la Organización....60
Ilustración 6 - Situación de los HRIS en los Sistemas de Información....64
Ilustración 7 - Framework para la reingeniería....86
Ilustración 8 - Planificación (I)....88
Ilustración 9 - Planificación (II)....89
Ilustración 10 - Planificación (III)....90
Ilustración 11 - Planificación (IV)....90
Ilustración 12 - Diagrama de Gantt....91
Ilustración 13 - Diagrama de Gantt (I)....91
Ilustración 14 - Diagrama de Gantt (II)....92
Ilustración 15 - Diagrama de Gantt (III)....92
Ilustración 16 - Diagrama de Gantt (IV)....93
Ilustración 17 - Estimación del uso de los distintos sistemas operativos....99
Ilustración 18 – XAML del WPF Application inicial....111
Ilustración 19 - C# del WPF Application inicial....112
Ilustración 20 - XAML del WPF Application con un botón....113
Ilustración 21 - C# del WPF Application con un botón....114
Ilustración 22 - Definición del control personalizado M4Button....117
Ilustración 23 - Definición del control personalizado M4Textbox....117
Ilustración 24 - Descripción de la operativa de la solución desechada....120
Ilustración 25 - Descripción de la operativa de la solución actual....121
Ilustración 26 - Solución del proyecto de migración....124
Ilustración 27 - Interfaz gráfica del componente TestInitAppWpf....125
Ilustración 28 - PRESENTATION_UC3M_BOTON....137
Ilustración 29 - PRESENTATION_UC3M_BOTON tras pulsar el botón....137
Ilustración 30 - PRESENTATION_UC3M_TAB....139
Ilustración 31 - PRESENTATION_UC3M_TAB migrada....139
Ilustración 32 - PRESENTATION_UC3M_GENDER....140
Ilustración 33 - PRESENTATION_UC3M_GENDER migrada....141
Ilustración 34 - PRESENTATION_UC3M_FRAMEWORK....142
Ilustración 35 - PRESENTATION_UC3M_FRAMEWORK migrada....143

Ilustración 36 - PRESENTATION_UC3M_FRAMEWORK sin estilos visuales.....155
Ilustración 37 - PRESENTATION_UC3M_FRAMEWORK con estilos visuales.....156

Índice de tablas

Tabla 1 - Tipos de sistemas de información 61
Tabla 2 - Coste temporal 94
Tabla 3 - Coste monetario 94
Tabla 4 - Explicación del método Migrate 130

CAPÍTULO I. INTRODUCCIÓN Y OBJETIVOS.

1. Introducción.

Desde finales del siglo pasado, hemos podido presenciar el tránsito de una era basada en el papel, a una era en la que el uso de las nuevas tecnologías guía el curso de la mayoría de las compañías. En un siglo en el que la tecnología está presente en todos los aspectos de nuestro entorno y nuestra vida cotidiana, las grandes empresas no deben estancarse en los mismos productos ofertados desde hace años. Los clientes exigen que la calidad de dichos productos sea cada vez mayor, y la competencia aumenta cada día en número y variedad. Por ello, muchas compañías se replantean si sería conveniente realizar una labor de rediseño de los procesos y los productos de su organización, con el fin de obtener beneficios con el cambio y aumentar su ventaja competitiva.

El trabajo desarrollado en este proyecto se basa en la aplicación de un proceso de reingeniería de software, consistente en un proyecto de migración de un producto software en una empresa productora de software. Para dicho proceso de migración, se desarrolló anteriormente un marco de trabajo con el fin de que el proceso de reingeniería diese los resultados esperados (*Cabezas, PFC, Junio 2008*) (*Colomo-Palacios, García-Crespo & Ruano-Mayoral. "EuroSPI", 2008*). Dicho marco de trabajo ha podido ser adoptado ya que, entre otros motivos, ofrece cierta flexibilidad que facilita la adaptación a la situación específica del proyecto desarrollado.

El presente apartado de introducción se compone de cuatro subapartados, que se explican brevemente a continuación:

- **Descripción del ámbito de estudio**. En qué consiste este proyecto y las etapas en las que está dividido.
- **Problemática**. Descripción del problema a resolver con el proyecto emprendido.
- **Delimitación de la solución**. Límites adoptados para la consecución de este proyecto.
- **Estructura de la memoria**. Organización en capítulos y apartados del presente Proyecto Fin de Carrera.

1.1. Descripción del ámbito de estudio.

El presente Proyecto Fin de Carrera mostrará cómo debe aplicarse un proceso de reingeniería de software a un proyecto de migración de un sistema de información desarrollado en una empresa cuya sede central se encuentra en Madrid, España.

Como será expuesto más ampliamente en el capítulo dos del presente proyecto, la reingeniería de software está adquiriendo gran importancia en grandes compañías, debido al deseo de éstas de evolucionar sus productos software, ya sea desde el punto de vista de su comercialización como de su uso interno. Gracias al rediseño de los procesos de construcción de dichos productos, se consigue adaptarlos a un determinado cambio, mejorando el producto final. Para ello es necesario no caer en la tentación de seguir el camino fácil, ya que muchas empresas deciden mejorar los procesos existentes debido a que están más familiarizados con ellos. Pero esto es una de las causas de mayor fracaso en la aplicación de un proceso de reingeniería en las empresas. Así, la reingeniería propone empezar de cero, pero sin pretender cambiar la funcionalidad del programa en cuestión, para que el trabajo de reingeniería de software que nos compete se efectúe de manera exitosa.

Este proyecto se basa en la aplicación de la reingeniería en sistemas de información corporativos, por ello se quiere mencionar la importancia con la que cuentan los sistemas de información en las organizaciones. Cada vez son más las empresas que destinan recursos al estudio del análisis y tratamiento de la información, ya que los resultados obtenidos tras dicho estudio aportan una información de salida extremadamente valiosa en cuanto a la toma de decisiones de la empresa se refiere. De este modo, la gestión de la compañía se ve mejorada, pudiendo hacer frente de un modo más agresivo a las empresas competidoras en un mercado de carácter cambiante.

Anteriormente al comienzo del proyecto que nos ocupa se realizaron dos tipos de acciones. En primer lugar, se llevó a cabo una primera etapa completamente teórica acerca del trabajo a realizar y los fines que se pretendían con el proyecto de migración. Posteriormente, en la segunda etapa fue necesario definir un marco de trabajo que marcaba las pautas a seguir durante el desarrollo del mismo para que el resultado final fuese el esperado. Dicho framework o marco de trabajo se definió de un modo flexible con el fin de poderse aplicar aunque el proyecto de migración se viese alterado en algún sentido. De este modo, actualmente nos encontramos en la tercera etapa del proceso de migración del sistema de información, con un framework ya definido y testado en un entorno productivo similar al que se aborda

en el presente proyecto (*Cabezas, PFC, Junio 2008*) (*Colomo-Palacios, García-Crespo & Ruano-Mayoral. "EuroSPI", 2008*).

1.2. Problemática.

Hoy en día son muchas las empresas que cuentan con procesos de reingeniería de software. Pero no existen metodologías concretas que las empresas puedan seguir para desarrollar dicho proceso.

Los procesos de reingeniería de software basados en procesos de migración de sistemas de información son bastante complejos. En concreto, el proyecto que nos ocupa tiene un ciclo de vida estimado en cuatro años. Este proyecto se enmarca en el tercer año de vida del proyecto.

Como se puso de manifiesto anteriormente, nos encontramos en la tercera etapa del proyecto de migración, con lo que se plantea un problema añadido a la labor del propio proceso de migración. Este problema consiste en analizar si el trabajo de migración realizado previamente se estaba ejecutando de manera correcta.

Después de ser analizado por el equipo de trabajo, se llegó a la conclusión de que la solución técnica adoptada para la migración de los componentes software en la anterior etapa no era correcta (esto será explicado con más detalle en el capítulo tres del presente Proyecto Fin de Carrera). Por este motivo, a la labor del proceso de migración, se plantea un problema que hay que resolver previamente, consistente en definir la tecnología y los pasos que se han de emprender para obtener el sistema migrado. Sin embargo, también se quiere hacer notar que el proceso definido en la anterior etapa ha sido considerado válido para emprender el tercer año del proceso de reingeniería y el presente Proyecto de Fin de Carrera.

1.3. Delimitación de la solución.

Disponiendo de un marco de trabajo definido que aporta soporte metodológico al proyecto de migración (*Cabezas, PFC, Junio 2008*) (*Colomo-Palacios, García-Crespo & Ruano-Mayoral. "EuroSPI", 2008*), la primera tarea del presente proyecto consiste en aplicar un proceso de reingeniería del software con el fin de ajustar la solución técnica para el proceso de migración del sistema de información. Dicho ajuste de la solución técnica es llevada a cabo por los integrantes del equipo de

trabajo, después de haber comprobado los fallos existentes en la anterior etapa. Tras realizar un estudio acerca del producto que se quiere obtener y acerca de cómo se quiere llegar a obtener dicho producto, será entonces posible ajustar el proceso de migración a seguir incluyendo las nuevas soluciones técnicas para el proceso.

En el presente proyecto serán mostrados cuáles eran los problemas existentes en la etapa previa, y la solución que se ha decidido ajustar y aplicar. De esta manera se podrá hacer frente a la segunda tarea, consistente en ejecutar la propia migración de la tecnología.

Se considera conveniente mencionar que esta tercera etapa en la que se ha desarrollado el presente proyecto, consta de una duración de cinco meses, con un equipo de trabajo de tres personas. En este sentido, se entiende que la migración de la tecnología existente en la empresa no se ha podido realizar en su totalidad, ya que como se dijo anteriormente, el proyecto de migración que nos compete tiene un ciclo de vida estimado en cuatro años.

1.4. Estructura de la memoria.

El presente Proyecto Fin de Carrera está dividido en siete capítulos.

El **primer capítulo** consiste en una introducción acerca del caso que se atiende, y en unos objetivos que deben cumplirse para cumplir con la consecución del trabajo para llegar a la solución esperada.

El **segundo capítulo** incluye un estudio del estado del arte. En él se muestra ampliamente en qué consiste la reingeniería del software, así como la migración de software y los sistemas de información corporativos en la empresa. Este capítulo es fundamental para entender claramente el resto del trabajo desarrollado.

En el **tercer capítulo**, se describe el problema al que nos enfrentamos, incluyendo una exposición de la situación previa, así como los problemas que se han identificado con respecto a dicha situación anterior, y los problemas que se plantean en esta etapa.

En el **cuarto capítulo**, se detalla el modo de solucionar los problemas identificados previamente, describiendo las características y el proceso de desarrollo de la solución propuesta en este proyecto. Esto incluye una descripción del framework adoptado, así como la planificación, el presupuesto y la tecnología y herramientas necesarias para llegar a la solución planteada.

En el **quinto capítulo**, se exponen las conclusiones extraídas de la realización de este proyecto, a la vez que se indican las posibles líneas de trabajo que se pueden llevar a cabo en un futuro.

El **sexto capítulo** está formado por los apéndices, que consisten en presentaciones (interfaces de usuario) utilizadas para la realización de este proyecto, y también se recogen aquellos documentos de importancia que se han obtenido como resultado de la realización de este proyecto.

Por último, el **séptimo capítulo** consiste en la bibliografía y recursos electrónicos empleados.

2. Objetivos.

El objetivo fundamental de este proyecto consiste en la aplicación de un proceso de reingeniería de una aplicación software comercial, en un sistema de información comercial de una empresa con sede central en Madrid, España.

Dicho proceso de reingeniería consistirá básicamente en un proceso de migración de las presentaciones (interfaces de usuario) creadas con la tecnología utilizada por dicha empresa, a otras presentaciones que representen fielmente a las primeras mencionadas pero en el entorno tecnológico .NET, de tal modo que se obtenga una presentación equivalente en Windows Presentation Fundation (WPF).

WPF es una de las novedosas tecnologías de Microsoft y uno de los pilares de Windows Vista, aunque su uso no es exclusivo de este sistema operativo ya que puede ser utilizado también sobre Windows XP. WPF encierra un enorme trabajo que parte de la gran experiencia acumulada en el desarrollo de interfaces de usuario que ha evolucionado desde la aparición de Windows.

Esta tarea de reingeniería planteada forma parte de una tercera etapa del proyecto de migración. La primera etapa fue completamente teórica. Ya en la segunda etapa se definió un framework para facilitar el desarrollo del proyecto y se efectuó el primer proceso de reingeniería para dicho proyecto de migración. Dicho framework se diseñó lo suficientemente flexible como para poder sufrir determinados cambios y adaptarse a los cambios a su vez sufridos en el proceso de migración del sistema de información.

Para que la migración de la tecnología tenga éxito, será necesario cumplir con otro objetivo, que consiste en realizar una labor de ajuste del proceso de migración del sistema de información. Para realizar un buen trabajo de reingeniería, será conveniente no basarse en un absoluto conjunto de ideas preconcebidas adoptadas en la etapa anterior, sino realizar un estudio acerca del producto que se quiere obtener y acerca de cómo se quiere llegar a obtener dicho producto. Será entonces cuando sea posible definir el proceso de migración a seguir y cumplir así el primero de los objetivos. Hay que tener claro que se trata de un proceso de reingeniería del software, por lo que no se desecha totalmente el producto antiguo, sino que se mantendrá su funcionalidad y se aprovecharán las ventajas que éste ofrecía.

CAPÍTULO II. ESTADO DEL ARTE.

3. Reingeniería.

En este apartado se pretende exponer en qué consiste un proceso de reingeniería y por qué es conveniente su utilización. Se dividirá dicho apartado en subapartados que tratarán de mostrar en primer lugar una breve introducción e historia de la reingeniería, para posteriormente pasar a explicar en qué consiste la reingeniería de procesos y el porqué de su uso, ya que para poder hablar de la reingeniería del software, es necesario conocer el origen de esta actividad. Finalmente, tras mostrar en qué consiste la reingeniería de software, se exponen las conclusiones extraídas de este apartado.

3.1. Introducción a la reingeniería.

El fenómeno de reingeniería surge con la intención de dar un giro al orden de los acontecimientos de la compañía, con el fin de mejorar el objetivo a cumplir. Cuando se trata de una compañía que emplea sistemas de información corporativos, la reingeniería del software aparece como la solución esperada para la renovación y mejora del producto software de la empresa.

En este capítulo se explicará principalmente en qué consiste un proceso de reingeniería, cómo llevarlo a cabo, y la necesidad de una compañía para realizar un proceso de reingeniería, prestando especial atención a la reingeniería de software y a las mejoras aportadas tras su utilización.

3.2. Historia de la reingeniería.

A finales del siglo XX, grandes organizaciones comenzaron a adoptar el fenómeno de reingeniería. Ya en el siglo XXI, la rápida expansión de este fenómeno, pasando muy rápidamente de la fase emergente a la fase de alto impacto en el mundo empresarial, está provocando cambios sustanciales en las organizaciones. El deseo de estas organizaciones es que, gracias a la experiencia acumulada, se disminuya el riesgo de fracaso en su aplicación.

Los principales promotores de la reingeniería en el área empresarial fueron, allá por los años 90, Michael Hammer y James Champy, gracias al libro "*Reengineering The Corporation"*. Ellos mismos criticaron la concepción inicial de la reingeniería, al igual

que muchos investigadores universitarios, consultores y ejecutivos, que mediante experiencias acumuladas observaron las limitaciones ofrecidas por este enfoque inicial. El pensamiento que había que adoptar no era otro que rediseñar, revolucionando radicalmente la forma en que se diseñó en el pasado.

La reingeniería, en su actual acepción, tuvo su origen en Occidente como una reacción de las empresas estadounidenses a sus problemas de competitividad frente a las compañías niponas. Estas últimas venían trabajando desde hacía mucho tiempo en la mejora continua, logrando de tal forma ir sacando continuas e importantes ventajas frente a las organizaciones occidentales. Así dadas las circunstancias, la única forma que tenían las empresas americanas era dar un salto que las reposicionara frente a sus competidores. Era menester destruir los viejos conceptos que las limitaban e impedían el desarrollo, evolución y puesta en práctica de nuevos conceptos tanto en materia de productos, como de procesos. Entre las más expuestas de las industrias se encontraban las automotrices, las cuales generaban productos que ya no satisfacían las demandas y necesidades del consumidor. Sus procesos, tanto de diseño como de producción, superaban en gran medida en plazo a los de sus competidores japoneses, además de contar con altos costos y bajos niveles de calidad, sobre todo si se la comparaba con sus rivales.

Así surgió la primera aplicación de la reingeniería de procesos como una forma de dar alcance a los competidores. Posteriormente, gracias a la aplicación de la reingeniería de procesos, surgió la reingeniería de software, la cual nos compete en el presente proyecto.

3.3. Definición de reingeniería.

En pocas palabras, la reingeniería consiste en el rediseño de los procesos de una organización con el fin de lograr la optimización de los flujos de trabajo y la productividad de dicha organización, entendiendo proceso como una sucesión de acciones interrelacionadas continuas y regulares, que ocurren o se llevan a cabo de una forma definida, y que llevan al cumplimiento de algún resultado. La reingeniería, de acuerdo a Hammer y Stanton, consiste en repensar de manera fundamental los procesos de negocios y rediseñarlos radicalmente, con el fin de obtener dramáticos logros en el desempeño. Los factores clave del concepto de reingeniería son: la orientación hacia los procesos, el cambio radical y la gran magnitud de los resultados esperados. En esta medida se entiende que no son los

productos los que llevan al éxito a una empresa, sino los procesos que los crean. Por ello, las compañías tienen que organizarse en torno al proceso.

La definición más aceptada actualmente para reingeniería es la siguiente: "La Reingeniería es el replanteamiento fundamental y el rediseño radical de los procesos del negocio para lograr mejoras dramáticas dentro de medidas críticas y contemporáneas de desempeño, tales como costo, calidad, servicio y rapidez". (*Hammer y Champy, Reengineering The Corporation, 1994*).

Se dice que durante una reingeniería, los procesos son objeto de una revisión fundamental, ya que es necesario realizarse las preguntas básicas sobre la compañía y cómo funciona sin dar nada por sentado. La reingeniería determina primero *qué* debe hacer una compañía para después determinar *cómo* debe hacerlo. Se olvida por completo de lo que es y se concentra en lo que debe ser.

Es considerable puntualizar que reingeniería no consiste en mejorar lo existente, sino en desechar en cierta medida los procesos actuales para recomenzar de nuevo. Todos los negocios están llenos de reglas de diseño de trabajo basadas en viejas suposiciones acerca de los objetivos de la tecnología, gente y organizaciones. Por ello, es necesario realizar una labor de ingenieros reinventando la manera de abordar el trabajo a realizar, asumiendo los riesgos que esto conlleva, pero teniendo en cuenta los beneficios que supondrá en el futuro.

"Tenga en consideración cualquier producto de tecnología que haya adquirido. Lo ve con regularidad, pero está envejeciendo. Se rompe con frecuencia, tarda en repararse y ya no representa la última tecnología. Si el producto es de hardware, probablemente lo tirará y se comprará uno nuevo. Pero si es un software personalizado, no dispondrá la opción de tirarlo. Necesitará reconstruirlo. Creará un producto con una funcionalidad nueva, un mejor rendimiento y fiabilidad, y un mantenimiento mejorado. Eso es lo que llamamos reingeniería." (*Roger S. Presuman, Ingeniería de Software, un enfoque práctico, 5ª ed.*).

3.4. El porqué de la reingeniería.

Se necesita reingeniería en una empresa básicamente:

- Cuando el rendimiento de la organización está por detrás de la competencia.
- Cuando la organización está en crisis, como una caída en el mercado.
- Cuando las condiciones del mercado cambian; como por ejemplo tecnología.
- Cuando se quiere obtener una posición de líder del mercado.
- Cuando hay que responder a una competencia agresiva.
- Cuando la empresa es líder y sabe que debe seguir mejorando para mantener el liderazgo.

Debido a las características del entorno, muchas empresas se ven obligadas a encontrar formas diferentes a las tradicionales para enfrentarse a la competencia y poder mantenerse en un mercado altamente competitivo. El mundo al que se enfrentan las compañías es denominado por muchos como las tres C: Clientes, Cambio y Competencia.

Cuando se emprende la reingeniería, existen dos áreas importantes a las que se ataca primero. La primera y más común es la relacionada con el servicio al cliente. La otra consiste simplemente en atacar la que esté funcionando peor, que suelen ser la financiera o la de manufactura.

Los clientes se han colocado en posición ventajosa, en parte por el acceso a mayor información. Los clientes asumen el mando, ya no tiene vigencia el concepto de "el cliente", sino que ahora es "este cliente", debido a que el mercado masivo hoy está dividido en segmentos, algunos tan pequeños como un solo cliente. Los clientes ya no se conforman con lo que encuentran, ya que actualmente tienen múltiples opciones para satisfacer sus necesidades.

En cuanto a la competencia, antes era más sencilla: la compañía que lograba salir al mercado con un producto o servicio aceptable y al mejor precio, realizaba una venta. Sin embargo ahora hay mucha más competencia y de clases muy distintas. En parte debido a la globalización, que trae consigo la caída de las barreras comerciales y ninguna compañía tiene su territorio protegido de la competencia extranjera. Ser grande ya no es ser invulnerable, y todas las compañías existentes tienen que tener la agudeza para descubrir las nuevas compañías del mercado. Las compañías nuevas no siguen las reglas conocidas y hacen nuevas reglas para

manejar sus negocios, y este es el motivo de que puedan llegar a ser "peligrosas", en cuanto a la competencia se refiere.

Y con respecto al cambio, éste se vuelve una constante, debido en gran medida a la rapidez del cambio tecnológico, que promueve la innovación. Los ciclos de vida de los productos han pasado de años a meses. Ha disminuido el tiempo disponible para desarrollar nuevos productos e introducirlos. Hoy las empresas tienen que moverse más rápidamente, o pronto quedarán totalmente paralizadas. Los ejecutivos creen que sus compañías están equipadas con radares eficientes para detectar el cambio, pero la mayor parte de ellas no lo está, lo que detectan son sólo los cambios que ellas mismas esperan. Pero los cambios que pueden hacer fracasar a una compañía son los que ocurren fuera de sus expectativas.

En este sentido se entiende que el cliente es el factor más importante para cualquier organización, y es necesario tratar de ofrecer servicios o productos que traten no sólo de alcanzar el nivel ofrecido por la competencia, sino mejorándolo, realizando para ello los cambios oportunos en la gestión de dicha organización, buscando alternativas a las prácticas tradicionales.

4. Reingeniería del software.

Llegados a este punto en el que se ha comprendido en qué consiste la reingeniería, se procede a explicar con más detalle la reingeniería del software.

Hoy en día disponemos de hardware mucho más potente y barato que unas décadas atrás, lo que permite una mejora en calidad y eficiencia de los productos software. En el momento en el que el hardware dejó de ser un impedimento para el desarrollo de la informática, se produjo la denominada "crisis del software", caracterizada por la imprecisión en la planificación de los proyectos y estimación de los costes, la baja calidad del software y la dificultad de mantenimiento de programas con un diseño poco estructurado. A raíz de esta crisis se comprendió la necesidad de crear estándares de desarrollo del software, lo que dio lugar a lo que hoy llamamos "Ingeniería del software", basada en el establecimiento y uso de principios de la ingeniería con el fin de obtener software fiable y que funcione eficientemente.

A pesar de la creación de estos estándares, muchos sistemas siguieron siendo desarrollados y mantenidos sin aplicar ninguna práctica de ingeniería de software por lo que hoy en día, muchas organizaciones se ven obligadas a seguir viviendo en esta crisis, dado que sus sistemas son vitales para el funcionamiento de dichas organizaciones.

La reingeniería del software surge como solución, en muchas ocasiones como única solución, a dichas organizaciones. La reingeniería del software pretende dejar morir esos sistemas imposibles de mantener, no sin antes extraer de ellos conocimientos que permitan crear un nuevo sistema fiable, eficiente y de fácil mantenimiento, basándose en las nuevas tecnologías pero sin dejar escapar la esencia de la compañía en la que se esté aplicando el proceso de reingeniería.

Previamente a abordar la definición formal de reingeniería del software, se ha considerado necesario explicar en qué consisten los sistemas de información *legacy*.

4.1. Sistemas de información *legacy*.

Los sistemas de información *legacy* (LIS) generalmente son la columna vertebral del flujo de información de las empresas y la principal forma de agruparla. Se tratan de sistemas de información que significativamente se resisten a la modificación y evolución. Casi siempre son ejecutados sobre hardware obsoleto, que son lentos y caros de mantener. Suelen carecer de la documentación necesaria para el entendimiento de los detalles del sistema, por lo que el mantenimiento del software es costoso. Además son difíciles de ampliar y la integración de los LIS con otros sistemas es una tarea difícil, en parte debido a la falta de interfaces limpias.

Una vez entendido qué es un sistema de información *legacy*, se procede a definir de manera formal la reingeniería del software.

4.2. Definición de reingeniería del software.

La Reingeniería del Software es un modo de modernización para mejorar las capacidades y/o mantenibilidad de los sistemas de información *legacy* mediante la aplicación de tecnologías y prácticas modernas. La Reingeniería del Software ofrece una disciplina de preparación para migrar un sistema de información *legacy* hacia un sistema evolucionable. En el presente proyecto se mostrará cómo es posible aplicar dicho proceso de reingeniería en una compañía existente.

El propósito de la reingeniería del software es que los sistemas existentes utilicen las ventajas aportadas por las nuevas tecnologías y habilitar el nuevo esfuerzo de desarrollo para aprovechar las ventajas de reutilizar sistemas existentes. Por ello, no se pretende cambiar la funcionalidad del programa, sino la forma del mismo, así como la documentación existente para él. En determinados casos, la reingeniería del software no sólo se centra en la forma del programa, sino que va más allá e incluye rediseñar para cambiar la funcionalidad del programa, buscando con ello mejoras para el usuario.

Una de las ventajas tras la aplicación de la reingeniería del software es que se perfecciona nuestro entendimiento del software, además de perfeccionar el software en sí mismo, haciendo más fácil la tarea de mantenimiento, evolución y reutilización.

La reingeniería del software es el puente desde viejas tecnologías hacía nuevas tecnologías que las organizaciones deben usar en la actualidad para responder al

cambio de requerimientos del negocio. Aunque la reingeniería se usa principalmente durante el mantenimiento del software, va mas allá de una simple ayuda para el mantenimiento.

Cuando una organización se encuentre con casos en que la fiabilidad del sistema sea cuestionable, aparezcan problemas de rendimiento, la tecnología se quede obsoleta, haya problemas de integración del sistema, el código sea de calidad pobre, el mantenimiento sea caro, y diferentes dificultades, se debe considerar la aplicación de la reingeniería del software en el sistema de información *legacy*.

Un hecho que se debe tener en cuenta es que la reingeniería de software requiere tiempo y dinero, y absorbe recursos que de otro modo se podrían emplear en preocupaciones más inmediatas. Por todo esto, se entiende que la reingeniería de software no se lleva a cabo en unos pocos meses, ni siquiera en unos pocos años, si se desea obtener un buen trabajo de reingeniería. Y en este sentido, es necesario mentalizarse, por parte de los altos directivos de la organización y de los propios empleados que trabajan para ésta, que la reingeniería de sistemas de información es una actividad que absorberá recursos de las tecnologías de la información durante bastantes años.

4.3. Procesos involucrados en la reingeniería del software.

La reingeniería del software debe ser entendida como un proceso mediante el cual se mejora un software existente haciendo uso de técnicas de ingeniería inversa y reestructuración de código.

Existen varios métodos y modelos de la reingeniería del software con el fin de estructurar el proceso de reingeniería. A continuación se muestra un modelo elegido que nos permitirá entender de una manera teórica cómo llevar a cabo la reingeniería del software. Se trata del denominado *Modelo Cíclico*. Este modelo define seis actividades que, en algunas ocasiones, se producen de forma secuencial y lineal, pero esto no siempre es así. Por ejemplo, puede ser que la ingeniería inversa (la comprensión del funcionamiento interno de un programa) tenga que producirse antes de que pueda comenzar la reestructuración de documentos (actividad supuestamente anterior a la ingeniería inversa).

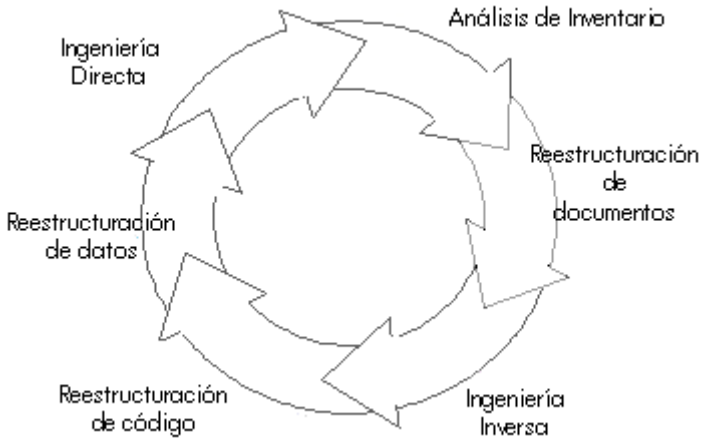


Ilustración 1 - Modelo cíclico de la reingeniería del software

Una ventaja del modelo propuesto es que, al tratarse de un ciclo, cada una de las actividades presentadas como parte del paradigma puede repetirse en otras ocasiones. Para un ciclo en particular, el proceso puede terminar después de cualquiera de estas actividades.

Análisis de inventario.

Se trata de un inventario de todas las aplicaciones de una organización de software, que proporcione una descripción detallada de dichas aplicaciones activas. Al ordenar esta información en base a su importancia para la empresa, será entonces posible asignar recursos a las aplicaciones candidatas para el trabajo de reingeniería. Dicho inventario deberá ser revisado con regularidad.

Reestructuración de documentos.

En primer lugar se desea explicar a qué se refiere reestructuración en este ámbito. "*Reestructuración del software: cambio de representación de un producto software, pero dentro del mismo nivel de abstracción*" (*M.A. Sicilia, artículo electrónico sobre Técnicas del mantenimiento del software – http://cnx.org*).

La documentación es uno de los puntos débiles de muchos sistemas de información *legacy*. Existen distintas opciones para la creación de la documentación que las organizaciones del software deberán seleccionar según resulte más adecuada para

cada caso. En muchos casos la creación de documentación supondría un elevado consumo de tiempo, por lo que las organizaciones deciden sobrevivir con la poca documentación de que dispongan. En otras ocasiones se considera conveniente la actualización de la documentación, pero se dispone de recursos limitados, por lo que se procede a documentar sólo aquellas partes que sufran determinados cambios. También se puede dar el caso de que el sistema sea fundamental para el negocio, por lo que es preciso volver a documentarlo por completo. Incluso en este caso, un enfoque inteligente consiste en reducir la documentación al mínimo esencial, sin prescindir de la documentación básica.

Ingeniería inversa.

La Ingeniería inversa es un proceso de recuperación de diseño. Con las herramientas de la ingeniería inversa se extraerá del programa existente información del diseño arquitectónico y de proceso, e información de los datos. La ingeniería inversa del software es el proceso de análisis de un programa con el fin de crear una representación del programa con un nivel de abstracción más elevado que el código fuente. En esencia, una ingeniería inversa con éxito precede de una o más especificaciones de diseño y fabricación para el producto, mediante el examen de ejemplos reales de ese producto.

Reestructuración de código.

El tipo más común de reingeniería es la reestructuración del código. Algunos sistemas de información *legacy* tienen una arquitectura de programa relativamente sólida, pero los módulos individuales han sido codificados de una forma que hace difícil comprenderlos, comprobarlos y mantenerlos. En estos casos, se puede reestructurar el código ubicado dentro de los módulos sospechosos.

Reestructuración de datos.

Un programa que posea una estructura de datos débil será difícil de adaptar y de mejorar. De hecho, para muchas aplicaciones, la arquitectura de datos tiene más que ver con la viabilidad a largo plazo del programa que el propio código fuente.

La reestructuración de código se produce a un nivel relativamente bajo de abstracción, mientras que la reestructuración de datos es una actividad de reingeniería a gran escala. En la mayoría de los casos, la reestructuración de datos comienza por una actividad de ingeniería inversa.

Ingeniería directa.

La ingeniería directa se ocupa de la recuperación de la información de diseño de un software ya existente y, además, utiliza esta información con el fin de mejorar su calidad global. En la mayoría de los casos, el software procedente de una reingeniería vuelve a implementar la funcionalidad del sistema existente, y añade además nuevas funciones y/o mejora el rendimiento global. Por ello, la tarea de la ingeniería directa consiste en una labor de reespecificación, rediseño, reimplementación y puesta en marcha.

Una vez explicado esto, lo cual nos sirve como base teórica para la comprensión de la reingeniería del software, se procede a explicar en un sentido ya más práctico el marco adoptado en nuestro caso para la realización de la reingeniería del software.

4.4. Marco para la reingeniería del software.

La figura mostrada posteriormente representa un marco para la reingeniería del software (*EuroSPI. Colomo-Palacios, García-Crespo & Ruano-Mayoral, 2008*). El marco, o framework, propuesto se basa fundamentalmente en dos premisas. La primera es la necesidad de un marco para la mejora de la reingeniería de software, no sólo en el proyecto de migración en la que se despliega, sino en el conjunto de la organización. La segunda es la voluntad de obtener un marco flexible, sin una rígida estructura que dificulte su adaptación a la situación específica de las organizaciones en que se despliegue.

Como se mostrará en posteriores capítulos del presente Proyecto Fin de Carrera, estas dos premisas son fundamentales en cuanto a su cumplimiento. La primera, ya que como se demostró en la anterior etapa del proyecto de migración, era absolutamente necesario definir un framework para el correcto desarrollo del proceso de migración y documentación. La segunda, ya que como se ha demostrado en esta etapa, el framework era lo suficientemente flexible como para poder alterar el proyecto sin necesidad de tener que adoptar un marco de trabajo distinto.

Ilustración 2 - Marco para la reingeniería

Dicho framework estará dividido en tres partes.

I. **Marco para el plan de reingeniería.** Es una fase anterior al propio plan de reingeniería, y tiene por objetivo elaborar el diseño del marco que guía el proceso en su conjunto.

II. **Ejecución del plan de reingeniería.** Durante la fase de ejecución se realizan las tareas de migración. Para ello será necesario desarrollar la definición del plan de reingeniería, la definición del proceso de reingeniería y la evaluación del plan de reingeniería.

III. **Plan de reingeniería post mórtem.** El objetivo de esta etapa es proporcionar la mayor cantidad de información posible al marco de reingeniería para introducir mejoras, tanto en su definición como en su

calidad. Se adopta un esquema de mejora continua para enriquecer el modelo con calibraciones y personalizaciones. Se propone que la fase de post mórtem sirva para llevar a cabo la comunicación de los resultados en el entorno operativo de la empresa, y, además, extender algunas de las características que han sido mejoradas en el proceso para contribuir a la madurez del proceso del software de la organización en la que se ha aplicado.

4.5. Planificación de la migración de software.

Tal y como se ha explicado a lo largo de este capítulo, cada vez son más las organizaciones que se enfrentan en la actualidad a la necesidad de modernizar sus sistemas software a través de su migración a sistemas más adecuados, lo que supone un complejo problema de reingeniería. Para que un proyecto de migración de software tenga éxito, se necesita un plan de desarrollo y un plan de migración adecuados con el fin de conseguir los resultados esperados.

El plan de desarrollo se centrará en la selección de los procesos software, métodos, herramientas y plataformas hardware y software apropiados.

El plan de migración debe sopesar y dar prioridad a los aspectos programáticos y técnicos para el desarrollo del sistema, antes que a las prioridades del cliente. Además, el plan de migración implica la necesidad de conseguir un balance entre costes, planificación, riesgos y recursos. Un plan de migración creado paralelamente y en conjunto al plan de desarrollo resulta necesario para asegurar una correcta transición funcional al nuevo sistema.

Para establecer con mayor claridad las bases de un plan de migración de software, en el siguiente apartado se detalla la planificación de un proceso de migración de software para el Departamento de defensa de los Estados Unidos (*Bergey, O'Brien and Smith, DoD Software Migration Planning 2001*), que identifica factores influyentes, remarca un conjunto de actividades para la planificación de la migración y ofrece unas directrices para el proceso de planificación de la migración.

4.6. Actividades para la planificación de la migración de software.

La siguiente figura muestra las actividades de las que se compone un plan de migración de software (*Bergey, O'Brien and Smith, DoD Software Migration Planning 2001*):

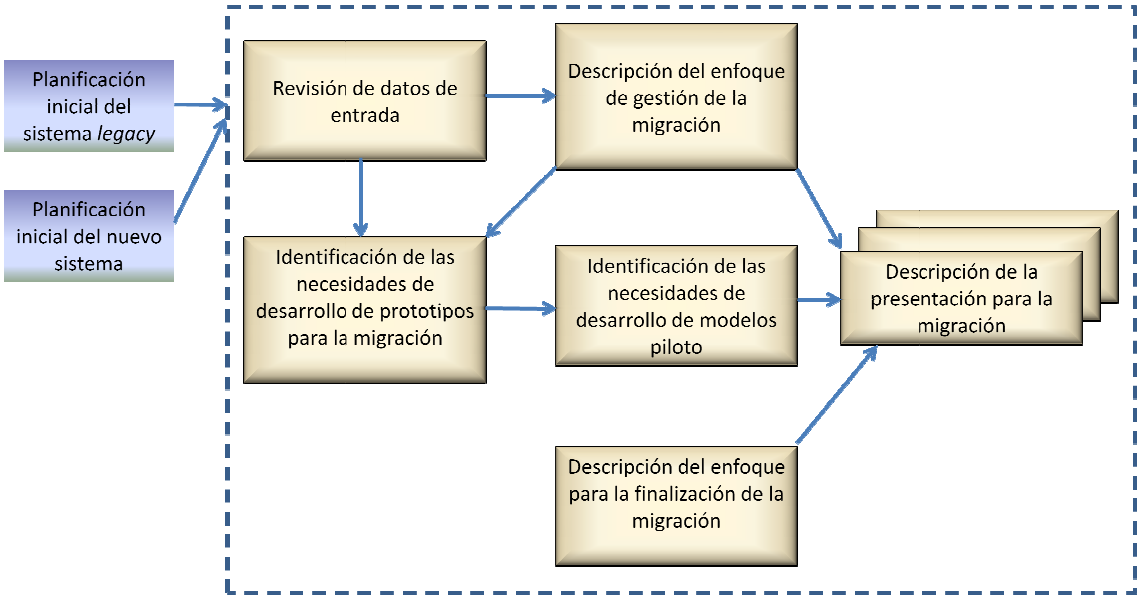


Ilustración 3 - Actividades para la planificación de la migración de software

A continuación se explica en qué consisten cada una de las actividades para la planificación de la migración de software mostradas en la figura de arriba:

1. Revisión de datos de entrada.

Se sostiene que estos datos de entrada proporcionan información básica acerca del estado actual del sistema de información *legacy*, información básica acerca del nuevo sistema, y el concepto de operaciones que describe al sistema final desde un punto de vista orientado al usuario. La revisión de estos datos suministra una aproximación inicial al grado de compatibilidad entre el sistema actual y el nuevo.

2. Descripción del enfoque de gestión de la migración.

La gestión de la migración se establece como un factor de éxito clave. Resulta importante determinar cómo van a ser gestionadas las mejoras que se van produciendo en sistemas *legacy* mientras se desarrolla el sistema final.

3. Identificación de las necesidades de desarrollo de prototipos para la migración.

De acuerdo con los autores, el plan de migración debe incluir las necesidades de desarrollar un prototipo. Al mismo tiempo, debe identificar cuáles de las cuestiones acerca de la migración tienen necesidad de la implementación de un prototipo con el objetivo de mitigar los riesgos, a la vez que sirve como demostración a los usuarios.

Esta actividad también debe recoger el ámbito del alcance del prototipo, de los conceptos de migración que están siendo estudiados, del conjunto de los resultados previstos, y de los mecanismos que se han de evaluar si se han alcanzado los resultados previstos. Se constata que un buen prototipo además, puede servir a los usuarios para probar el nuevo interfaz de usuario y la nueva funcionalidad, antes de que se tomen decisiones relacionadas con el sistema y su software.

4. Identificación de las necesidades de desarrollo de modelos piloto.

Debido al gran número de de factores a tener en cuenta en un entorno *legacy*, y al notable número de usuarios que pueden verse afectados por este entorno, se estima que la implementación de modelos pilotos a pequeña escala, puede servir de ayuda en el proceso de migración.

Esta actividad debe recoger cuáles de las cuestiones acerca de la migración sugieren la necesidad de la implementación de un modelo piloto para la validación de la integridad del sistema, el rendimiento del mismo, y el grado de aceptación por parte de los usuarios del sistema. Una vez finalizado el modelo piloto, se considera conveniente evaluar los resultados obtenidos a partir de éste para determinar si los objetivos propuestos inicialmente son alcanzables, o si por el contrario deben ser revisados y actualizados.

5. Descripción de la presentación para la migración.

Se especifica que la presentación de nuevas prestaciones debe ser planeada cuidadosamente. Esta actividad incluye dos tareas:

5.1. Descripción de las fases de las entregas del sistema.

Como parte del plan de migración, se considera oportuna la realización de una evaluación del número, el tipo, y las fases de las entregas del sistema que mejor se ajusten a las prioridades del cliente y las necesidades del mismo. Los autores señalan necesaria la identificación de hitos en el plan de migración que puedan ser aceptados por clientes, desarrolladores y usuarios por igual. Además se estima conveniente realizar una valoración de riesgos previa a la finalización de la entrega del sistema para poder certificar si las estimaciones iniciales son razonables, el enfoque está bien concebido y se encuentra dentro de las prioridades de la organización, y el impacto potencial en clientes y usuario puede considerarse aceptable.

5.2. Estipulación de las necesidades de apoyo.

De acuerdo con los autores, se debe proporcionar un nivel de apoyo para ayudar a cada usuario en la utilización de cada entrega del sistema y la transición al nuevo sistema. Además de esto, se identifica que los usuarios con frecuencia requieren de asistencia funcional que les ayude a comprender las capacidades y funcionalidad del sistema final.

6. Descripción del enfoque para la finalización de la migración.

Después de finalizar cada nueva presentación y fase de entrenamiento, los autores establecen que el desarrollador debe garantizar que los usuarios pueden migrar al nuevo sistema. Aquí se incluyen aspectos tan básicos como la elección del momento oportuno y la amplitud con la que los usuarios tendrán la posibilidad de trasladarse al nuevo sistema. Se determina que el plan de migración deberá incluir enfoques que aceleren el proceso y permitan al sistema *legacy* anterior que sea desechado lo antes posible.

4.7. Directrices para la migración de software.

A lo largo de los años, las grandes organizaciones han ido acumulando notables cantidades de software, que a medida que se ha ido desarrollando ha generado considerables gastos tanto de mantenimiento como de desarrollo. A partir de un determinado punto, se daba la situación de que cualquier mejora suponía unos costes de similares magnitudes a los que ya se habían abordado hasta ese punto, debido a la complejidad del cambio en el sistema.

En este sentido, surgen una serie de directrices que marcan la pauta a seguir en los proyectos de reingeniería basados en la migración que tienen como objetivo proteger, gestionar, evolucionar y modernizar dicho software. Estas directrices han sido diseñadas en concreto para sistemas *legacy* del departamento de defensa de los Estados Unidos, y son recogidas en el documento "*DoD Legacy System Migration Guidelines*" (*Bergey, Smith and Weiderman, DoD Legacy System Migration Guidelines 1999*). A la hora de valorar estas directrices se debe tener en cuenta el punto de vista de la organización adquiridora, la organización desarrolladora o migradora del sistema, o una combinación de ambas. Para la organización adquiridora es importante tener un conocimiento de la parte técnica lo suficientemente amplio como para realizar correctamente las especificaciones del sistema. Se considera relevante tener en cuenta el punto de vista de la organización desarrolladora o migradora del sistema para poder tomar las decisiones técnicas más adecuadas y seguir el correspondiente disciplinado proceso de reingeniería. A continuación, se procede a enumerar diez directrices básicas que según los autores del documento, se deben tener en consideración al llevar a cabo un proceso de migración.

1. Desarrollar una estrategia comprensiva que contenga hitos realizables y mensurables para cada proyecto de reingeniería.

La mayoría de las organizaciones cuentan con una estrategia de alto nivel a largo plazo cuando afrontan un proyecto de reingeniería. Se determina relevante que esta estrategia especifique resultados realistas derivados del esfuerzo de reingeniería y que, a su vez, estos resultados puedan ser relacionados con unos hitos específicos e incrementales que sean alcanzables y mensurables.

Tal cual sucede con las decisiones relacionadas con la arquitectura, las decisiones iniciales tomadas en un proyecto de reingeniería deben tener impacto duradero en

la estructura y funcionamiento de un sistema. Estas decisiones resultan difíciles de cambiar y conllevan altas repercusiones en el resultado global del proyecto. Por este motivo, se dice que estos primeros pasos deben ir enfocados hacia el problema real y deben ser relacionados con hitos realizables. Además, se ha establecido que una estrategia comprensiva debe encargarse de abarcar y probar escrupulosamente todos los supuestos, así como prestar atención a los detalles de mayor relevancia.

2. En caso de que sean necesarios servicios de ingeniería de sistemas externos, definir cuidadosamente sus funciones y monitorizarlas.

Un hecho que apuntan los autores es que los servicios externos para ingeniería de sistemas ofrecen ventajas como el amplio conocimiento del dominio, la especialización técnica, la objetividad, la rapidez con la pueden asignar personal extra al proyecto, o la nueva perspectiva que pueden aportar a una situación. Es necesario que sus funciones sean cuidadosamente definidas y monitorizadas para evitar que se produzcan deficiencias en el desarrollo del proyecto.

Para la organización, se considera necesario mantener un conocimiento lo suficientemente profundo como para distinguir si el desarrollo del proyecto se está desviando en exceso de las previsiones iniciales, mientras que los servicios externos necesitan disponer de los recursos y el tiempo adecuado para la realización del proyecto.

3. En caso de que nueva tecnología sea utilizada para un proyecto, se debe poner a disposición una formación adecuada que abarque tanto el contenido técnico como los motivos por los que se produce el cambio.

Los autores resaltan que con frecuencia, un proyecto de reingeniería va a beneficiarse de tecnologías más novedosas de las que existen en el sistema. Cada cierto tiempo, el hardware debe ser remplazado y actualizado, y nuevos paradigmas de programación deben ser adoptados. No se determina viable continuar con el método de trabajo habitual, al tiempo que se intenta instruir en las nuevas tecnologías al mismo grupo de trabajo. Para ello, se define necesario un esfuerzo consciente y persistente para el adiestramiento del grupo de trabajo en nuevas tecnologías, o este mismo grupo debe ser reemplazado o completado con nuevos trabajadores conocedores de dichas tecnologías. También existe la posibilidad de que se produzca una combinación de los tres casos anteriores. En

opinión de los autores, a la hora de planificar la formación, las organizaciones deben tener en consideración las habilidades técnicas necesarias para la utilización de nuevas tecnologías, además de las correspondientes cuestiones de motivación. Las habilidades técnicas pueden ser adquiridas a través de cursos ofrecidos por proveedores, organizaciones de consultoría externa, o por un departamento de formación interno. En lo referente a las cuestiones de motivación, se considera necesario que la organización presente una exposición razonada que respalde los cambios, además de los posibles beneficios que pueda aportar a los empleados el uso de las nuevas habilidades, ya que en una primera toma de contacto, los empleados suelen mostrarse reacios a los cambios.

4. Establecer y mantener el control de la configuración de la gestión del sistema.

Para que un sistema pueda ser gestionado de manera efectiva, se debe establecer una primera línea dentro de la configuración de la gestión que esté a disposición para poder mantener una evolución disciplinada del sistema. Se considera imprescindible que el sistema se encuentre bien documentado y que la prioridad entre las peticiones de cambio y su correspondiente impacto en el sistema resulte transparente. Además de todo esto, se determinan necesarios los datos de los costes de mantenimiento del sistema, una configuración de la gestión adecuada, y la planificación de las capacidades de gestión.

Muchos sistemas *legacy* no están bajo el control adecuado debido a que están pobremente documentados y tienen un historial de evaluaciones inadecuado unido a un insuficiente control de cambios.

El análisis acerca de cómo son atendidas las peticiones de cambio, supone un indicador para comprobar si el sistema está bajo control. Se establece que sólo se podrán tomar decisiones racionales para nuevos lanzamientos si las peticiones de cambio se priorizan de acuerdo con su importancia y dificultad. Otro indicador que se identifica para determinar el nivel de control de un sistema se puede obtener a través del estudio de sus medidas históricas. Estas medidas incluyen los cambios que se han producido, el coste que han producido esos cambios, y qué problemas han surgido. Cuando esta información no se encuentra disponible, se considera comprometido realizar estimaciones de costes significativas para los distintos tipos de cambios a los que es susceptible el sistema. También debido a esto, puede resultar complicada la planificación de cambios a largo plazo.

5. Debe existir un proceso cuidadosamente definido y documentado para la obtención y validación de requisitos.

Se establece necesario tener correctamente documentado el concepto de operaciones para el sistema final para poder evitar que se produzcan fallos debido a una escasa obtención y validación de los requisitos y que pueden derivar en importantes defectos en el mismo proceso de obtención y validación de requisitos. Existen diferentes tipos de requisitos: funcionales y no funcionales, de usuario o cliente, de hardware o software, de arquitectura, de mantenimiento, y logísticos. Esto refleja que los requisitos no son unidimensionales y que no sólo reflejan las necesidades del usuario interesado, sino que también pueden servir de utilidad para inversores que tengan interés en el sistema. A día de hoy, se han definido numerosas opciones para mejorar el proceso de definición de requisitos. Entre estas opciones se encuentra la creación de escenarios de usuario, el rápido prototipado de elementos del sistema, o el desarrollo de *storyboards*, con el objetivo de mejorar la parte en la que se define el interfaz de usuario, además de adquirir un conocimiento más profundo de las características y funcionalidad del sistema.

6. Tomar la arquitectura del software como una consideración de reingeniería de primer orden.

Una evaluación metódica de la arquitectura del software de un sistema se considera imprescindible en el desarrollo del proceso técnico de reingeniería. Esta evaluación resulta necesaria para determinar si es viable mantener como base para futuros desarrollos la arquitectura de software de la que se dispone.

Len Bass define la arquitectura de software de un programa o un sistema como "la estructura o estructuras del sistema, que comprende los componentes software, las propiedades externamente visibles de dichos componentes, y las relaciones entre ellos" (*Bass, Clements and Kazman 1998*). De acuerdo con los autores del documento, si la arquitectura existente representa un punto de partida viable, la aproximación técnica de la reingeniería debe cimentarse a partir de esta arquitectura. De lo contrario se debería abandonar la herencia previa y empezar desde el principio. Únicamente si la arquitectura del sistema fuese comprendida en todo su ámbito, sería posible, por ejemplo, aprovechar las interfaces existentes para modularizar los componentes y utilizarlos en la nueva arquitectura. En el caso

de que la arquitectura existente estuviese correctamente documentada, sería posible reutilizar este tipo de documentación para la nueva arquitectura. A esto se debe añadir que a menos que exista un profundo conocimiento del núcleo del sistema y su entorno operativo, se podrían generar requisitos de la evolución del sistema que resulten incomprensibles.

7. Debe existir una clara separación entre diferentes procesos de reingeniería.

Se establecen necesarios un conjunto de tareas para orientar la ejecución de cada paso además de la comprensión de cómo funcionan todas estas tareas en conjunto. A esto se le debe añadir una amplia visión con vistas al futuro del proceso de reingeniería que integre los procesos y los paquetes de trabajo del proyecto entero.

Se distinguen cuatro elementos críticos para un proceso de reingeniería: el personal, la tecnología, el proceso en sí, y los recursos disponibles. Estos cuatro elementos van intrínsecamente unidos porque para obtener un producto de calidad normalmente es necesario disponer de personal cualificado con amplios recursos y que dispongan de las tecnologías adecuadas usando un proceso válido, siendo este último imprescindible en el conjunto.

Un proceso de reingeniería debe centrarse principalmente en la resolución del problema, ya que en condiciones normales el punto de partida es el sistema *legacy* disponible en el momento. Esto puede implicar en muchos casos que sea necesaria la aplicación de reingeniería a la inversa sobre el software del sistema, con el objetivo de obtener una mayor comprensión del programa y como paso previo a la aplicación del proceso de reingeniería (ingeniería inversa).

Mientras que la reingeniería del software se destaca como crítica dentro del proceso, ésta se corresponde a una tarea de nivel bajo, que puede ser definida una vez se hayan resuelto las mayores consideraciones del proceso de reingeniería. El objetivo según los autores, es evaluar sistemáticamente los principales elementos que contribuyen al espacio de problemas y soluciones del proceso y además evaluar si la organización y el proyecto tienen todos los recursos necesarios para emprender las tareas de reingeniería.

8. Creación de un plan de reingeniería que esté orientado hacia el equipo de trabajo… y seguirlo.

A la hora de diseñar un planteamiento de migración para la reingeniería de un sistema *legacy*, los autores detectaron numerosos aspectos que con frecuencia necesitan ser resueltos por un equipo interdisciplinario de ingenieros y expertos en un determinado campo en consenso con la cúpula encargada de tomar las decisiones referentes a la reingeniería del software. El equipo de proyecto debe desarrollar un profundo conocimiento acerca del sistema, su misión, el entorno funcional en el que es implementado, y los usuarios del sistema. A esto se debería añadir un interés por entender los objetivos de la organización y los objetivos que propician la reingeniería del sistema. Se especifica como esencial que el plan de proyecto documentado cuente con el respaldo de los principales mandos de la organización. Los cambios más trascendentes planteados por el proceso de reingeniería pueden suponer varios pasos, y en muchos casos requieren de medidas dictadas por los principales mandos.

La decisión para ejecutar el plan de reingeniería viene dada cuando al equipo responsable de implementar el plan se le concede la responsabilidad de desarrollar el plan. Una parte específica de este plan debe tener como objetivo obtener el respaldo de los principales mandos de la organización.

9. El equipo de dirección debe estar comprometido con el proyecto hasta el final.

De acuerdo con los autores, el equipo de dirección debe respaldar el plan de proyecto hasta que éste haya finalizado. Esto implica un seguimiento muy detallado y la aplicación de medidas correctoras en cuanto el plan se desvíe ligeramente. Este planteamiento resulta especialmente crítico en un proyecto de reingeniería debido a que cualquier paso en falso en las etapas iniciales o intermedias puede resultar en errores de gravedad que son complicados de corregir si son percibidos con el proceso bien avanzado. Cuando un equipo directivo no está comprometido totalmente con el plan de reingeniería, tiende a perderse el foco del proyecto. En definitiva, el equipo directivo debe estar totalmente centrado y comprometido, y no debe verse distraído por otros proyectos de mayor prioridad.

10. Los decretos del equipo de dirección no deben ignorar las posibilidades técnicas.

Dentro de un mismo proyecto puede haber involucrados mandos o cargos pertenecientes a diferentes niveles, así como a diferentes organizaciones. Todo esto puede llegar a derivar en una toma de decisiones dirigida a intereses privados o agendas individuales de cada organización, o puede dar lugar a decisiones basadas en los deseos de los individuos encargados de tomar decisiones en lugar de en las posibilidades técnicas. Los autores establecen que las decisiones ejecutivas deben estar respaldadas por un conocimiento de las posibilidades técnicas. A su vez, los managers deben conocer los costes, la planificación y la capacidad de dar mayor relevancia a uno u a otro en un momento determinado con la ayuda de los asesores técnicos. A pesar de esto, el equipo de dirección puede ser propenso a exigir resultados y una realización anticipados.

Además, se especifica que la planificación detallada y los hitos de un proyecto, sólo se pueden conseguir con precisión mediante un estudio y análisis detallados de los parámetros técnicos del sistema, basados en la comprensión del propio sistema, con sus datos históricos y el conocimiento de las habilidades del equipo de trabajo. Continuando en esta línea, el equipo de dirección debe atender a los detalles técnicos y entender las consecuencias de la toma de decisiones, antes de establecer estos valores basándose en intuiciones.

Siguiendo las indicaciones de los autores, una vez que el equipo de trabajo haya seleccionado el plan de acción para acometer un análisis inicial para determinar cuáles son los requisitos, evaluar los primeros enfoques y arquitecturas y determinar la mejor estrategia para desarrollar el proceso de reingeniería del sistema, el equipo directivo estará capacitado para finalizar el plan de implementación y determinar los costes, la planificación, y los recursos necesarios con un alto nivel de certeza.

4.8. Conclusiones.

La reingeniería del software es una herramienta esencial en la renovación de sistemas computacionales, ya que su producto final es un sistema adaptado a las necesidades actuales de la empresa.

La reingeniería es una muy buena opción tanto en costo como en tiempo: en costos ya que costaría considerablemente menos hacer un sistema con reingeniería, basándonos en el sistema original, que crear un sistema totalmente nuevo; con lo que concierne a tiempo es análogo, hacer la reingeniería de un sistema *legacy* nos toma menos tiempo que hacer un sistema de la nada. Lo anterior nos lleva a que casi siempre es mejor hacer la reingeniería de software que hacer un software nuevo, de tal manera que se oriente el cambio hacia mayores niveles de facilidad en cuanto a mantenimiento, reutilización, comprensión o evolución.

En cuanto a la eficiencia del sistema, la reingeniería del software sigue siendo la mejor opción debido a que el sistema creado constará con las características y capacidades del sistema *legacy*, pero ampliadas, actualizadas y optimizadas, sin necesidad de perder la arquitectura ni interfaz que tenía el sistema inicial.

Insistir en que la reingeniería es fácil, es insistir en que no es ingeniería. Para que una empresa adopte el concepto de reingeniería, tiene que ser capaz de deshacerse de las reglas y políticas convencionales que aplicaba con anterioridad y estar abierta a los cambios por medio de los cuales sus negocios puedan llegar a ser más productivos.

Reingeniería también significa el abandono de viejos procedimientos y la búsqueda de trabajo que agregue valor hacia el consumidor. Las actividades de valor agregado tienen dos características: es algo que el cliente aprecia y es importante que se ejecuten correctamente desde la primera vez.

Lo interesante de la reingeniería es que no hay un modelo de reingeniería, y que se puede aplicar más de una vez para un mismo propósito. Aunque sí hay ciertos principios de valor general que pueden ser aplicados en prácticamente todas las organizaciones. Asimismo, hay una cierta metodología que se puede rescatar de la experiencia de casos exitosos.

Si se está convencido de las bondades de la reingeniería, de sus conceptos y de su metodología, debe ponerse en práctica cuanto antes. Las principales empresas del mundo ya hicieron o están haciendo reingeniería, para pasar de la era de la industrialización a la nueva era de la información-comunicación. En 20 ó 30 años

todas las empresas que sobrevivan habrán hecho reingeniería o se fundarán en base a sus principios.

5. Sistemas de información corporativos.

El presente proyecto trata la aplicación de la reingeniería del software en sistema de información corporativos. Por lo tanto se considera relevante designar un apartado a los sistemas de información.

Actualmente, los sistemas de información proporcionan una gran ventaja competitiva a las organizaciones. Independientemente de la estrategia de gobierno de la organización, los sistemas de información tienen un impacto estratégico al reducir los costes de producción y permitir identificar potenciales segmentos de mercado. La comunicación es uno de los elementos claves para poder dirigir una organización con éxito. Y es ahí donde comienza el papel de los sistemas de información, simplificando en gran manera los aspectos de la gestión efectiva de información (*Kadiyala and Kleiner 2005*).

5.1. El concepto de gestión de la información.

La información es la clave del éxito en cualquier aspecto relacionado con la gestión empresarial moderna. Comúnmente se dice que la información es poder y que quien la tiene, tiene el poder. El desarrollo y la implantación de sistemas de información para la gestión es un fenómeno relacionado con un uso adecuado de la información que permita mejorar la planificación empresarial, la toma de decisiones y los resultados (*Adeoti-Adekeye 1997*).

La gestión de la información se ha definido como la capacidad transversal de una organización para crear, mantener, capturar y hacer disponible la información para la toma de decisiones en el lugar correcto, en el momento adecuado, en manos de quien mejor la pueda utilizar y con el mínimo coste (*Langemo 1980*). En resumen, se puede decir que el aspecto clave para la gestión de la información es el tratamiento de la información de la organización por medio de tecnologías modernas (*Adeoti-Adekeye 1997*).

5.2. Definición de un sistema de información.

Una definición formal de un sistema sería: conjunto de elementos que relacionados entre sí ordenadamente contribuyen a un determinado objetivo (*Real Academia de la Lengua Española*). Un sistema de información también se corresponde con un conjunto de personas, procedimientos, bases de datos, hardware y software, que de forma coordinada reúne, procesa, almacena y distribuye datos para el procesamiento de transacciones en el nivel de operaciones e información para apoyar en la toma de decisiones (*Duff and Assad, 1980*) y el control de la organización (*Laudon and Laudon, Management Information Systems: New Approaches to Organization and Technology, 1998*). Además de ayudar en la coordinación de la organización, los sistemas de información también proporcionan soporte a la dirección y al resto de trabajadores ayudando a analizar problemas y a visualizar situaciones complejas (*Laudon and Laudon, Management Information Systems: New Approaches to Organization and Technology, 1998*).

El término información se refiere a los datos o conjuntos de datos, elaborados y situados en un contexto de forma que son útiles para los participantes en la organización (*De Pablos Heredero, 2001*). Sin embargo, los datos son los flujos de hechos en bruto, que representan los sucesos ocurridos en el entorno de la organización y en la organización misma, antes de ser organizados y almacenados como información (*Laudon y Laudon, Sistemas de información gerencial: Organización y tecnología de la empresa conectada a la red, 2002*).

Un sistema de información se encarga de transformar la información de entrada de modo que se obtenga una información de salida valiosa para la compañía. Un sistema de información consta de los siguientes elementos funcionales, que están relacionados con la organización y su entorno (*Adeoti-Adekeye 1997*):

- *Percepción*: Introducción de información en la organización, tanto capturada como generada.
- *Registro*: Captura física de datos.
- *Procesamiento*: Transformación de acuerdo a las necesidades específicas de la organización.
- *Transmisión*: Los flujos que tienen lugar en un sistema de información.
- *Almacenamiento*: Porque se supone que la información será utilizada en un futuro.
- *Recuperación*: Búsqueda de la información almacenada.
- *Presentación*: Creación de informes y difusión.

Un sistema de información tiene tres actividades básicas, que son: entrada, procesamiento y salida. La *entrada* es la captura o recolección de datos en bruto del interior de la organización o de su entrono externo, para que puedan ser procesados por el sistema de información. El *procesamiento* se encarga de la conversión, la manipulación y el análisis de entradas brutas para darles sentido. Y la *salida* consiste en la distribución de la información procesada a las personas para que la puedan usar. Además de estas actividades, otro elemento fundamental en los sistemas de información es la *retroalimentación*, por medio de la cual las salidas se devuelven a los miembros apropiados de la organización para ayudarles a controlar el funcionamiento de la etapa de entrada (*Laudon y Laudon, Sistemas de información gerencial: Organización y tecnología de la empresa conectada a la red, 2002*). La siguiente ilustración muestra gráficamente el curso de la información en los sistemas de información:

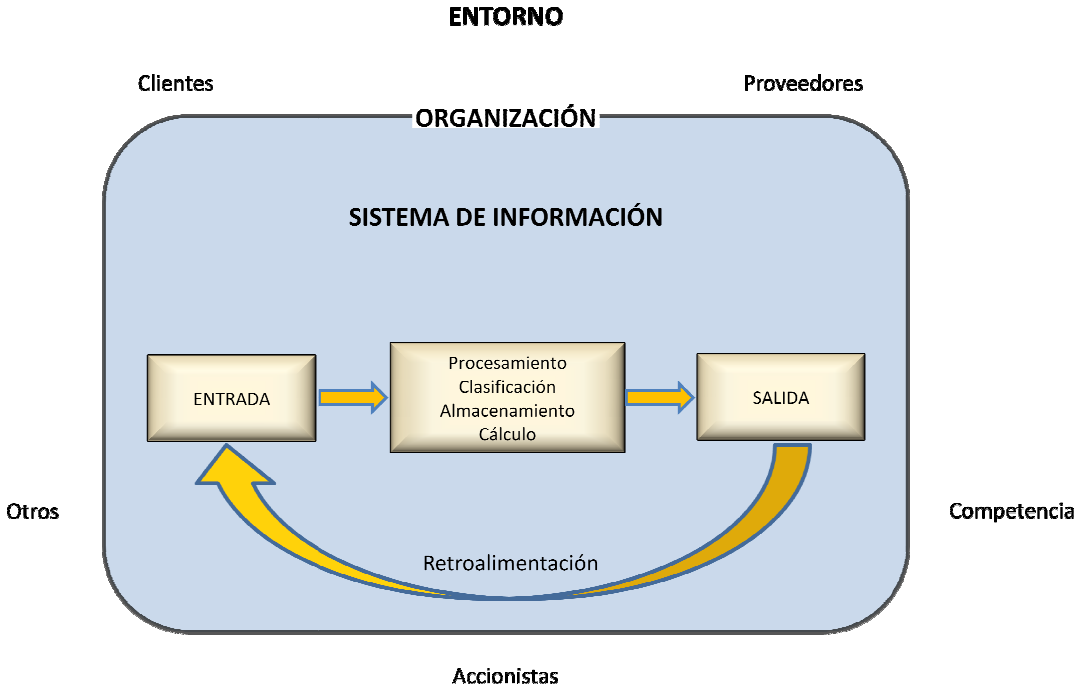


Ilustración 4 - Funciones de un sistema de información

Los sistemas de información manejan información acerca de personas, lugares y cosas relevantes de la propia organización o de su entorno, como se puede ver en la anterior figura.

5.3. Sistemas de información en la empresa.

Para la correcta coordinación del trabajo en una organización, se considera necesario disponer de una jerarquía estructurada y un conjunto de procedimientos formales. La jerarquía sitúa al personal en una estructura de pirámide en la que la autoridad y la responsabilidad aumentan con la altura. Los niveles más altos de la jerarquía contienen a los empleados de dirección, profesionales y técnicos, mientras que los niveles inferiores están formados por el personal operativo.

Según el nivel organizativo al que presten servicio, se determina que los sistemas de información se pueden agrupar en las siguientes cuatro categorías:

- **Sistemas del nivel operativo**: Ayudan a la dirección de operaciones a realizar el seguimiento de las actividades elementales de la compañía como las ventas, el aprovisionamiento de materiales, los pagos, etc.
- **Sistemas de nivel de conocimientos**: Su finalidad es ayudar a la empresa a integrar nuevos conocimientos para el negocio y para que la organización pueda controlar el flujo de la documentación dentro de sí misma.
- **Sistemas de nivel gerencial**: Están pensados para las actividades de seguimiento, control y toma de decisiones de la escala intermedia de dirección. Sirven para comparar los resultados empresariales actuales con los obtenidos con anterioridad.
- **Sistemas de nivel estratégico**: Sirven para que la alta dirección pueda afrontar y dirigir cuestiones estratégicas y medidas a largo plazo.

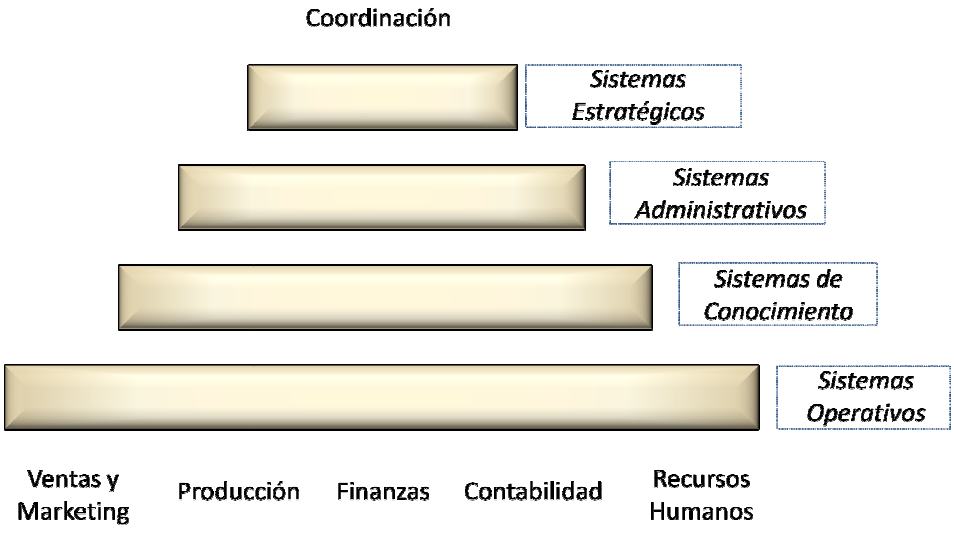


Ilustración 5 - Arquitectura de Información de la Organización

Es posible profundizar más sobre la clasificación anterior con el fin de llegar a los siguientes sistemas de información:

NIVEL ORGANIZATIVO	**TIPO DE SISTEMA**
Nivel Estratégico	Sistemas de Apoyo a Ejecutivos (ESS, Executive Support Systems)
Nivel Gerencial	Sistemas de Información Gerencial (MIS, Management Information Systems) Sistemas para la Toma de Decisiones (DSS. Decision-Support Systems)
Nivel de Conocimientos	Sistemas de Trabajo de Conocimientos (KWS, Knowledge Work Systems) Sistemas de Automatización de Oficinas (OAS, Office Automation Systems)
Nivel Operativo	Sistemas de Procesamiento de Transacciones (TPS, Transaction Processing Systems)

Tabla 1 - Tipos de sistemas de información

- **Sistemas de Procesamiento de Transacciones:** Son los sistemas de negocio básicos del nivel operativo de la organización. Estos sistemas efectúan y registran las transacciones diarias rutinarias que son necesarias para el funcionamiento del negocio.
- **Sistemas de Trabajo de Conocimientos** y **Sistemas de Automatización de Oficinas:** Los primeros satisfacen las necesidades de información del nivel de conocimientos de la organización y son utilizados por los trabajadores de conocimiento, para la creación y la integración de nueva información en la organización. Los segundos, son sistemas que ayudan a aumentar la productividad de los trabajadores de datos.
- **Sistemas de Información Gerencial:** Sirven al nivel de administración de la organización, proporcionando informes y cuadros de mando para controlar el rendimiento de la organización.
- **Sistemas para la Toma de Decisiones:** Además de la información que obtienen de forma externa, estos sistemas utilizan información de los Sistemas de Procesamiento de Transacciones y los Sistemas de Información Gerencial, para facilitar la toma de decisiones complejas y rápidas.

- **Sistemas de Apoyo a Ejecutivos:** La alta dirección utiliza estos sistemas para la toma de decisiones estratégicas, teniendo en cuenta datos externos como cambios en la legislación fiscal o comercial y aglutinándolos con la información extraída de los Sistemas de Información Gerencial y los Sistemas para la Toma de Decisiones internos.

5.4. Sistemas de información gerenciales.

Un sistema de información gerencial se puede definir como un sistema que utiliza procedimientos formales para proporcionar a la dirección, en todos los niveles y funciones, la información adecuada (tanto interna como externa) para permitir la toma de decisiones efectivas referentes a las actividades de planificación, control y gobierno (*Argyris 1971*). En la línea de lo comentado en el apartado anterior, se puede decir que estos sistemas se utilizan en las organizaciones para ayudar a alcanzar los objetivos marcados, planificar y controlar los procesos y operaciones, gestionar los riesgos y facilitar la adaptación a posibles cambios. Algunas de las características principales que se han descrito de estos sistemas se enumeran a continuación (*Adeoti-Adekeye 1997*):

- Están centrados en la información que la dirección de la organización necesita.
- Representan un flujo ordenado de información.
- Integran tareas de procesamiento de datos con funciones de negocio
- Proporcionan un sistema de generación de informes basados en la información almacenada.

A pesar de las aportaciones positivas que este tipo de sistemas tienen en las organizaciones, hay grandes evidencias, extraídas de numerosas encuestas, de que sistemas de información gerencial, incluso tecnológicamente muy avanzados, han tenido relativamente poco éxito al satisfacer las necesidades de las compañías que los habían implantado. En su mayor medida, esto es debido a un estudio inadecuado del sistema computacional, a la falta de compromiso y apoyo por parte de la dirección, y a un mal análisis acerca de las necesidades de información reales (*Adeoti-Adekeye 1997*).

5.5. Sistemas para la gestión de recursos humanos.

Los primeros departamentos que desarrollaron de manera formal las funciones de gestión de recursos humanos, datan del año 1920. Durante todos estos años la gestión de recursos humanos ha ido evolucionando desde una función de registro y mantenimiento de datos a otra más estratégica asociada a la contabilidad, el marketing y las finanzas. Los estudios sobre recursos humanos han ido ampliado su análisis a otras tareas como la selección, la formación, la compensación y la evaluación del personal (*Poutanen n.d.*). En los últimos años, se han planteado nuevos enfoques para la gestión de recursos humanos, entre los que cabe destacar (*Melián-González 2004*):

- *La utilización de buenas prácticas*. Diferentes estudios han permitido identificar algunas prácticas que tienen efectos positivos en los resultados empresariales como pueden ser: la contratación selectiva, la formación extensiva, la evaluación del rendimiento, la implantación de incentivos, la promoción y el desarrollo profesional de los empleados, etc.
- *La gestión por competencias*. Se considera que la principal aportación de un sistema de gestión por competencias es identificar el conocimiento clave que un empleado o una organización ha de poseer para alcanzar sus objetivos (*Dragandis and Mentzas 2006*). La gestión por competencias permite la integración de todos los sistemas de gestión de recursos humanos bajo un único modelo.
- *La gestión del capital intelectual*. El capital intelectual se refiere al material intelectual, conocimientos, habilidades, propiedad intelectual y experiencia que se pueden utilizar para crear valor añadido. Se estipula que esta gestión permite definir y controlar variables como la satisfacción, la motivación, el liderazgo, etc.

Una de las primeras definiciones de sistema de información para la gestión de recursos humanos *(HRIS, Human Resource Information System)* explica que un HRIS es un sistema utilizado para adquirir, almacenar, manipular, analizar, recuperar y distribuir información pertinente acerca de los recursos humanos de una organización (*Tannenbaum 1990*). Hay otra definición algo posterior, que está un poco más enfocada a las tecnologías que soportan el sistema; así, un HRIS se

define el conjunto de bases de datos, aplicaciones de ordenador, hardware y software que se utilizan para capturar, registrar, almacenar, gestionar, distribuir, presentar y manipular datos de recursos humanos (*Broderick and Boudreau 1991*).

Con la evolución de la gestión de recursos humanos, fueron apareciendo nuevas tareas y actividades para las que también se desarrollaron e implantaron sistemas de información. En un primer momento los sistemas de información para la gestión de recursos humanos se utilizaban para el soporte de transacciones y el control directivo, pero después aparecieron nuevas aplicaciones como la mejora en la toma de decisiones y el aumento de la competitividad (*Poutanen n.d.*). Por este motivo, se identifica que estos sistemas además de estar relacionados con los sistemas de trabajo de conocimientos (KWS), también pueden relacionarse con los sistemas de información gerencial (MIS), como muestra la siguiente figura:

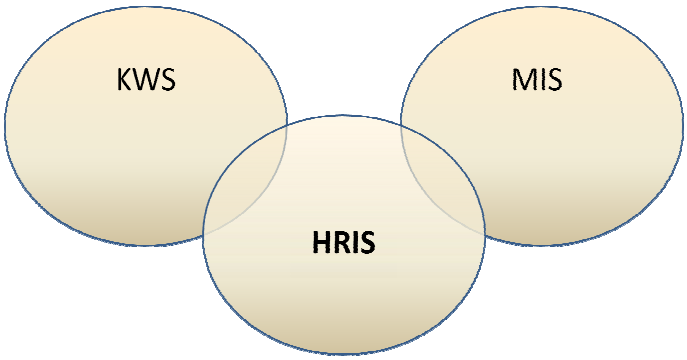


Ilustración 6 - Situación de los HRIS en los Sistemas de Información

Para que la adopción de un sistema de información para la gestión de recursos humanos tenga éxito, la compañía debe tener un convencimiento absoluto en los beneficios que éste puede aportar. Además de beneficios generales como pueden ser la eliminación de errores, el acceso a la información de forma rápida y oportuna y la disminución de costes de gestión, se han identificado otras razones administrativas y estratégicas por las cuales las empresas deberían implantar un sistema de información para la gestión de recursos humanos (*Ngai and Wat 2006*). Se distinguen entre ellas: el aumento de la competitividad asociado a la mejora de las operaciones de gestión de recursos humanos, la elaboración de un gran número de informes sobre gestión de recursos humanos que cubren una amplia variedad de aspectos de estudio, el cambio del centro de atención del procesamiento de transacciones de recursos humanos hacia una visión más estratégica de la gestión

de los mismos, la incorporación de los propios empleados al sistema de información para la gestión de recursos humanos, o la reestructuración de las funciones de gestión de recursos humanos de la compañía.

Sin embargo, el lado negativo de la implantación de este tipo de sistemas es que conllevan unos elevados costes asociados. Una encuesta realizada por el *Institute of Management and Administration* en 2002 indicó que los principales problemas y obstáculos encontrados por las empresas para la adopción de un HRIS son (*Ngai y Wat, 2006*):

- La falta de personal cualificado para su gestión y mantenimiento.
- La falta de presupuesto para dedicarlo al HRIS.
- Problemas con la gestión del tiempo.
- La necesidad de dirigir esfuerzos hacia otros departamentos distintos al de gestión de recursos humanos.
- La falta de soporte en tecnologías de la información.

5.6. Costes asociados a los sistemas de información.

El desarrollo, la implantación y el mantenimiento de un sistema de información requieren, además de un desembolso para la adquisición de hardware y software, una inversión en otros conceptos que pueden no ser tan evidentes a la hora de realizar un presupuesto del sistema. Aunque para la realización de estos presupuestos hay una gran cantidad de directrices que se pueden seguir, se ha comprobado que muchas veces estos cálculos no son realistas y obligan a las organizaciones a aceptar soluciones que distan mucho de ser óptimas (*Barreu 2001*).

Para la valoración del impacto causado por la introducción de un nuevo sistema en una organización se define una metodología de gran utilidad denominada análisis centrado en el trabajo. Esta metodología se encarga de evaluar un sistema de información en el marco de la organización teniendo en cuenta quiénes van a ser los usuarios, cómo va a ser utilizada por estos, cuál va a ser la estructura de la información que manejará y cuáles van a ser los procesos que va a llevar a cabo ese sistema. La aplicación de la metodología permite identificar las siguientes categorías, que se suelen omitir en los presupuestos:

- ***El salario y los gastos del personal y la dirección implicados en el análisis y la implantación del sistema***. Al planificar el proyecto para un sistema, los directores suelen presupuestar el tiempo y la remuneración de los individuos que componen el equipo de trabajo, pero raramente tienen en cuenta el esfuerzo propio que invierten en tareas de revisión, de análisis y de toma de decisiones relacionadas con el proyecto.
- ***Los costes relacionados con otras funciones y trabajos que se ven afectados como consecuencia del proceso de implantación***. Si el personal está muy involucrado en la implantación del sistema se considera que la calidad de otros servicios de la organización se puede ver mermada de forma temporal. Si por alguna razón esta disminución de calidad no es admisible, puede tomarse en consideración la contratación de trabajadores temporales durante el tiempo que dure la implantación; algo que normalmente se pasa por alto al hacer el presupuesto.
- ***Los costes de las modificaciones en las instalaciones que albergarán el nuevo sistema***. Entre estos costes se distinguen el mobiliario, el acondicionamiento de aire, la fontanería, el cableado, etc.
- ***Los costes y el período de formación***. Las empresas que venden sistemas de información suelen proporcionar paquetes de formación a sus clientes. Pero estos paquetes raramente incluyen las modificaciones particulares del sistema instalado ni las necesidades de formación de la organización a lo largo del tiempo. Por este motivo, puede determinar necesario contratar a más personal para la formación, y como el personal de la organización sufre altas y bajas, la formación se puede convertir en un proceso continuo.
- ***Los costes de migración y conversión de los sistemas y procesos existentes***. Se estima que la adquisición de un nuevo sistema puede implicar la conversión de datos y la reingeniería de procesos para adaptarse a la nueva situación. Los costes de conversión deben abarcar, en caso de que fuera necesario, el desarrollo de procedimientos para la transferencia de los datos al nuevo sistema, las pruebas y el borrado de los datos antiguos.

Para contrarrestar el impacto de los costes mencionados anteriormente se han identificado las siguientes estrategias (*Barreu 2001*):

- Realizar un análisis de la idoneidad del sistema de información y los procesos de la organización.

- Conseguir la implicación de la dirección en el proyecto.
- Consultar ideas a otros usuarios.
- Diseñar planes de contingencia con vistas a las peores situaciones posibles que puedan darse.

5.7. Conclusiones.

La información y su adecuado manejo son dos de los aspectos fundamentales que rigen el funcionamiento de cualquier organización empresarial. Los sistemas de información permiten implantar y automatizar el proceso de manejo de la información desde su recolección y registro, hasta su transformación en informes y otros tipos de información elaborada.

Toda organización tiene una estructura jerárquica que rige su funcionamiento. Esta estructura suele tener forma de una pirámide dividida en niveles, que van desde el personal operativo hasta la dirección estratégica. Cada uno de estos niveles tiene unas necesidades diferentes en cuanto a información se refiere. Por ello, existen diferentes tipos de sistemas de información dependiendo del nivel jerárquico al que prestan servicio: sistemas de nivel operativo, sistemas de conocimiento, sistemas administrativos y sistemas estratégicos.

En cuanto a los sistemas de información para la gestión de recursos humanos, estos se sitúan entre los sistemas de gestión del conocimiento y los sistemas de información gerenciales, ya que se utilizan tanto para el soporte de transacciones como para la mejora de la toma de decisiones y el aumento de la competitividad de la organización.

Para finalizar, se destaca que los sistemas de información proporcionan numerosos beneficios a las empresas. Una organización que cuente con una buena gestión de la información puede tomar mejores decisiones estratégicas y conocer con mayor certeza el resultado de dichas decisiones. Otras ventajas son la mejora del servicio prestado a los clientes y el incremento de la productividad por la optimización de los recursos. Junto con el crecimiento y el desarrollo de las tecnologías de la información y las comunicaciones, la implantación de sistemas de información está teniendo un impacto estratégico fundamental en las organizaciones.

CAPÍTULO III. DESCRIPCIÓN DEL PROBLEMA.

6. Situación previa.

Analizar la situación previa al proyecto actual que se desea llevar a cabo, resulta imprescindible para la correcta toma de decisiones acerca de cómo llegar a la solución deseada. En este sentido, este apartado pretende explicar cuál era la situación anterior y el camino que se seguía para llegar a la solución final.

La compañía española en la que ha sido desarrollado este proyecto, posee una plataforma cuyo empleo es la gestión organizativa. Dicha plataforma cuenta con una herramienta basada en un editor de presentaciones, del cual se requiere su migración a una versión que aproveche los avances que presenta la tecnología .NET Framework de Microsoft. El objetivo final de esta migración es que las presentaciones nuevas se puedan ejecutar en un entorno en el que .NET se pueda ejecutar, aligerando la instalación de la herramienta corporativa.

6.1. Consideraciones de la plataforma propiedad de la compañía.

Los responsables técnicos de esta plataforma pertenecen al departamento de tecnología de la compañía mencionada previamente. Se trata de una solución integral que ya cuenta con una década, y que a lo largo de su ciclo de vida ha ido consolidándose y adaptándose a los cambios tecnológicos y funcionales requeridos por el entorno. A su vez, la necesidad de aprovechar las continuas mejoras en las tecnologías de interfaces gráficas, una de cuyas máximas exponentes es .NET Framework de Microsoft, ha tomado un papel prioritario para la compañía. Este proyecto se va a centrar en la migración del editor de presentaciones de la mencionada plataforma, a la tecnología .NET Framework v3. O mejor dicho, se centrará en la migración de las presentaciones creadas en dicho editor de presentaciones, a la tecnología .NET Framework v3.

El proceso de creación del software que representa la plataforma, tuvo una duración de aproximadamente dos años, e involucró a más de dos millares de personas. Durante ese tiempo y ante la necesidad de la empresa de sacar con premura el producto al mercado, no se consideró necesario el seguimiento estricto de una metodología de desarrollo, por lo que mucha de la documentación necesaria en la implementación de una plataforma de estas dimensiones, no se encuentra

disponible o nunca existió. Además, un hecho que agrava la situación es que en los últimos años la empresa ha decidido prescindir de los servicios de un gran número de trabajadores que estuvieron involucrados en el desarrollo de la plataforma, lo que supone la consecuente dispersión y pérdida de conocimiento. Uno de estos empleados decidió, antes de abandonar la organización, eliminar todo el trabajo que había realizado para la plataforma, y lo consiguió efectivamente. Esto supuso una gran pérdida de trabajo ya desarrollado y que tuvo que volver a repetirse posteriormente, lo que indicaba que el nivel de seguridad existente en la compañía no era el adecuado. Esto fue un error paliado por la compañía, y en este momento el nivel de seguridad es aceptable. A esto se debe añadir la reticencia de algunos trabajadores a transmitir importante información acerca de los diferentes proyectos en los que estaba involucrado en la empresa, hecho que dificulta en gran medida la creación de una documentación adecuada para los procesos que se encontraban en desarrollo.

6.2. Consideraciones de las etapas previas a este proyecto.

El ciclo en el que nos encontramos (Ciclo 3, Proceso 1) consiste en la tercera etapa del proyecto de migración de software para la herramienta diseñador de presentaciones de la plataforma propiedad de la compañía en la que es desarrollado dicho proyecto. La primera etapa, o ciclo, consistió en una fase completamente teórica acerca de *qué* se quería hacer y *cómo* habría que hacerlo. La segunda etapa consistió en el desarrollo de un marco de reingeniería y en la ejecución del propio proceso de reingeniería, con una presentación sencilla para comprobar que la migración era correcta. La presente y tercera etapa representa el tercer ciclo del proceso de reingeniería, con un nuevo enfoque desde el punto de vista técnico, que será explicado posteriormente, y sin olvidarnos de la adopción del marco desarrollado en la segunda fase.

En la segunda etapa mencionada, fue necesario llevar a cabo una serie de reuniones entre los integrantes del grupo formado por miembros de la Universidad Carlos III, denominado a partir de ahora GC3, y el grupo tecnológico de la organización, con el propósito de unificar criterios y mejorar el proceso de desarrollo de software, sobre todo para intentar evitar la duplicación de código o la inclusión de código innecesario en el proceso de migración. En un primer momento,

el GC3 fue sometido a un proceso de autoformación con el fin de familiarizarse con la empresa y conocer el funcionamiento del software y los lenguajes de programación que serían necesarios para llevar a cabo el proceso de migración. Las personas encargadas de la supervisión y seguimiento del desarrollo del proyecto, se encargaron de realizar una planificación de las primeras etapas a abordar y calcularon las estimaciones iniciales para la duración de cada una de ellas. Una vez planificado el proyecto y completado el proceso de formación, se reunieron los integrantes del grupo tecnológico de la organización y los del GC3, para establecer las primeras pautas de trabajo a seguir en función a la planificación mencionada, consistente en el análisis e investigación de una primera presentación sencilla, abarcando los 21 objetos de los que estaba compuesta, con sus correspondientes propiedades, así como la funcionalidad inherente en la presentación.

Esta segunda etapa de la que estamos hablando tuvo problemas, en un principio, debido a la falta de documentación disponible en la organización, por lo que se necesitó elaborar una fórmula propia para recoger toda la información que se iba obteniendo a medida que se estudiaba cada componente. Esta tarea resultó ser más compleja de lo esperado y consumió una cantidad de recursos y tiempo notablemente mayor a lo estimado en el inicio. Pero uno de los mayores problemas que surgió durante esta etapa, fue conocer con exactitud la profundidad y la complejidad que pueden alcanzar ciertos objetos del editor de presentaciones, agravándose con el hecho del elevado desconocimiento por parte de los miembros de la empresa, y la falta de documentación, acerca de la funcionalidad y propósito de ciertos objetos que estaban analizándose en la sencilla presentación. Además, dichos componentes del departamento tecnológico de la empresa, tenían diversidad de opiniones acerca del conocimiento aportado.

Por lo tanto, los problemas básicos de esta segunda etapa fueron:

- Adoptar un framework eficiente
- Averiguar el conocimiento total y exacto acerca de los objetos contenidos en la presentación sencilla que se pretendían migrar
- Tratar de automatizar la migración de dicha presentación
- Acogerse a un estándar eficaz de documentación, ya que el formato de la documentación interna de la empresa no se ajustaba a ninguno en concreto.

6.3. Consideraciones de la presente etapa de este proyecto.

El propósito de la presente y tercera fase de este proyecto en la que nos encontramos, consiste en continuar con el plan de migración de software establecido en la anterior etapa. Pero en primer lugar, fue necesario confirmar que el método de trabajo seguido en la fase anterior era el idóneo.

El equipo de trabajo para esta etapa estaba formado por tres personas. Una de ellas es el jefe de proyecto, perteneciente a la empresa en la que se pretende ejecutar el plan de migración. Otra, es un integrante del GC3, que ya participó en las anteriores etapas de este plan de migración y pudo aportar su experiencia en esta fase. Y por último se encuentra el proyectando. Aparte, el equipo de trabajo se completaba con la aportación de otras dos personas fundamentales para la consecución de este proyecto. Una de ellas es el tutor del presente Proyecto Fin de Carrera, cuya aportación se cristalizaba en las tareas de documentación y seguimiento, y de forma adicional, en las tareas de gestión de la Ingeniería del Software. Y la otra aportación vino dada con la importante ayuda del responsable de este proyecto de migración, principalmente en funciones de arquitectura del sistema. Además, el personal de la empresa del área de tecnología estuvo en todo momento disponible para ayudar en lo que fuese necesario, tanto con respecto a cuestiones de diseño, como con respecto a cuestiones de codificación. Todos los anteriormente mencionados, participaron en las reuniones celebradas al comienzo de esta fase del proyecto con el fin de establecer el camino a seguir para la consecución del proyecto de manera correcta, es decir, resolver qué debía hacerse, cómo debía hacerse, y qué había que cambiar en cuanto a la etapa anterior.

Debido a que no hubo una etapa de formación hacia el proyectando por parte de la empresa como se hizo en la anterior etapa del proyecto, la primera tarea fue comprender en qué consistía el proyecto a realizar, estudiar qué método se seguía para la migración de las presentaciones y la documentación de la misma, la familiarización con la herramienta de la organización que se pretendían migrar, y el entendimiento del framework adoptado en la anterior etapa, el cual también fue seguido en la presente etapa debido a que era lo suficientemente flexible para adaptarse a los cambios que sufriera el proyecto. En este momento del apartado cabe destacar que gracias a la documentación elaborada en la anterior etapa del proyecto, fue mucho más cómodo y sencillo llegar a entender la finalidad del

proyecto de migración y el modo de trabajo seguido para conseguir el producto final, ya que dicha documentación era completa y estaba muy bien esquematizada.

En las primeras reuniones celebradas, el responsable del proyecto de migración expuso un cambio radical con respecto a la arquitectura presentada en el modelo de migración. En esta etapa, cualquier presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un ejecutable (.EXE) ni efectuar almacenamientos en el repositorio de la compañía, permitiendo, de esta manera, que las modificaciones en las presentaciones sean traducidas en el momento de ser necesitadas, eliminando los problemas de consistencia en el almacenamiento persistente entre las presentaciones. Mientras que en la etapa anterior, era necesario disponer de un proyecto completo en Microsoft Visual Studio (MVS a partir de ahora) que no se encargara simplemente de la traducción de código, sino de la creación de clases innecesarias en cuanto a la migración se refiere, así como de la compilación de esas clases para la creación del .EXE de la presentación a migrar, además de la necesidad de grabación de la presentación migrada en el repositorio de la organización.

Este hecho no cambió el objetivo final del proyecto, que al fin y al cabo consiste en la migración de las presentaciones creadas con el editor de presentaciones mencionado previamente; pero sí el camino seguido para obtener el producto final en formato WPF, convirtiéndolo en un proceso más sencillo y sobre todo con menor carga en cuanto a tiempo y espacio se refiere. Es decir, reducción de tiempo, porque no es necesario compilar las clases para generar el ejecutable, ni es necesario ejecutarlo posteriormente, ni tampoco se requiere del almacenamiento de la presentación migrada. Y reducción de espacio, porque se prescinde de clases innecesarias que sólo servían para la generación del .EXE, y además ya no es necesario el almacenamiento en el repositorio de la organización. Por lo tanto, se puede concluir que el nuevo enfoque es considerablemente mejor que el de la anterior etapa; el desarrollo de éste se encuentra explicado en el siguiente capítulo del presente Proyecto Fin de Carrera.

En la anterior etapa, para realizar la migración se disponía de un código propio de la organización con el que se definen las presentaciones, denominado OBL, y dicho código era migrado a WPF para aprovechar las ventajas de la tecnología .NET. Unido a este nuevo enfoque descrito anteriormente, el responsable del proyecto propuso utilizar para la migración un código XML que representaba exactamente el mismo código OBL de la presentación a migrar, en lugar de utilizar dicho código

propiedad de la compañía. Esta propuesta supuso la clara aceptación por parte de todo el equipo de trabajo, ya que el código XML a migrar, aunque bastante parecido al código OBL de la compañía, se asemejaba mucho más a la tecnología .NET, en cuanto al formato se refiere.

Otro hecho que se tuvo en cuenta fue que en la anterior etapa era completamente necesario indicar en el proyecto de MVS, cuál iba a ser la presentación a migrar. Esta solución no es correcta, por lo que en la presente etapa se decidió realizar esta labor (la labor de conocer qué presentación migrar) gracias a herramientas disponibles en la plataforma de la organización. Es decir, que el proyecto creado en MVS sólo se encargase del proceso de migración, y desde la propia plataforma de la organización se indicara qué presentación se pretendía migrar.

Debido a que en la etapa previa no hubo participación del proyectando en el proyecto de migración, se dispuso de la cierta ventaja de comprobar cómo se estaba llevando a cabo dicho proyecto desde una perspectiva externa, sin ningún tipo de obcecación ni dando nada por sentado. Tras entender completamente en qué consistía el proyecto y cómo se producía el proceso de migración, se pudo comprobar que ciertos objetos se estaban migrando creando controles personalizados para su representación en WPF, los cuales suponen un gasto con respecto al almacenamiento en el repositorio de la organización. Sin embargo, dichos controles podían crearse mediante una correspondencia directa en la migración entre el objeto obtenido de la presentación, y el objeto a representar en WPF. Por ello, tras ciertas reuniones con los empleados de la empresa, se llegó a la conclusión de que esta situación debía cambiar, reduciendo al máximo posible el número de controles personalizados en el proceso de migración.

Además de lo anterior, otro asunto al que se hizo frente fue el control de los eventos, esto es, las acciones a ejecutar en la presentación al pulsar un botón, elegir una opción de un menú, seleccionar una fecha de un calendario, etc. Debido al cambio de enfoque para esta etapa en la que nos encontramos mostrada en el presente Proyecto Fin de Carrera, era necesario disminuir al máximo posible el *code-behind*, código necesario en la anterior etapa para ejecutar determinadas acciones en una presentación. Tras pensarlo y discutirlo entre los miembros del equipo de trabajo y el responsable de este proyecto de migración, se decidió que sería absolutamente necesario el control de los eventos sin necesidad de disponer de controles personalizados que mediante *code-behind* ejecutaran las acciones oportunas.

A todo esto se suma el hecho de que el código generado en la etapa anterior para la consecución de la migración era objetivo de revisión, ya que no era lo más eficiente que podía llegar a ser. Además, la documentación también debía ser revisada y ampliada con los nuevos objetos, propiedades, eventos y atributos que aparecieran en las presentaciones.

Por todo lo mencionado anteriormente, en esta tercera etapa mostrada en el presente Proyecto Fin de Carrera, se consideró necesario continuar con el framework adoptado en la anterior etapa, pero teniendo en cuenta que el proceso de migración iba a verse sometido a varios cambios de consideración, aplicando el proceso de reingeniería de un modo efectivo y con vistas al futuro.

7. Problemas identificados.

En este apartado se procede a enumerar los diferentes problemas encontrados en la realización de este proyecto, y que dan lugar a la necesidad de encontrar una solución para intentar garantizar con el mayor porcentaje de seguridad posible el éxito del mismo.

1. **Insuficiente formación del equipo de desarrollo**, más concretamente el caso del proyectando, ya que los demás integrantes del equipo habían trabajado en las anteriores etapas del proyecto. A pesar de la documentación aportada gracias a su elaboración en la anterior etapa, las reuniones celebradas, así como a las respuestas ofrecidas a mis dudas planteadas, la autoformación fue escasa para poder desarrollar el proyecto adecuadamente desde el primer día de trabajo en él, principalmente debido al poco tiempo empleado para dicha formación.

2. **Cambio de enfoque en el modelo de migración**. En la anterior etapa era necesario la compilación y ejecución de un ejecutable para la migración de una presentación, la cual además debía almacenarse en el repositorio de la organización. En esta etapa nada de esto es necesario. Este cambio de perspectiva en el diseño del modelo de reingeniería supone la adaptación por parte del equipo de trabajo a un nuevo método de trabajo.

3. **Código a migrar distinto al de la anterior etapa**. El proceso de reingeniería en la anterior etapa se basaba en la migración de un código de entrada propiedad de la compañía referente a la presentación a migrar denominado OBL, a un código de salida en tecnología .NET referente a la presentación migrada. En esta etapa, la salida se espera que tenga la misma finalidad, pero el código de entrada en este caso se tratará de un código XML que representa fielmente el código OBL de la presentación a migrar, por lo que el método de traducción se ve alterado.

4. **Problemas con el nuevo código a traducir**. Tras la traducción previa de código OBL a código XML, se producían ciertos errores que provocaban que la migración no fuese correcta. Además, dicho código fue aportado por el responsable del proyecto mediante unas librerías que permitían la traducción mencionada, lo cual suponía que siempre hubiese que tener presente si dichas librerías estaban disponibles. Debido a los cambios casi semanales en

las versiones del entorno de la empresa para su correspondiente actualización, siempre que se produjese una de estas actualizaciones sería necesario registrar manualmente las librerías mencionadas anteriormente hasta que éstas fuesen incluidas en las actualizaciones.

5. **Indicar en la plataforma qué presentación se pretende migrar**. Anteriormente, se indicaba en el propio proyecto de MVS el identificador de la presentación a migrar. Sin embargo, en esta etapa se considera esto un error, y se decide que la propia plataforma donde se encuentra la herramienta diseñador de presentaciones, es el lugar donde indicar la presentación a migrar.

6. **Exceso de objetos creados mediante controles personalizados**. En la anterior etapa, muchos de los objetos que se migraron estaban realizados mediante un control personalizado. En esta etapa se considera que es más correcto disponer del mínimo número de controles personalizados, tratando de realizar la migración de todos los objetos posibles mediante correspondencia directa.

7. **Control de eventos**. El control de las acciones es un punto crítico para este proyecto, ya que se desea disminuir al máximo posible el *code-behind*. Dicho control de eventos debería realizarse en el código que representa la propia presentación ya migrada, utilizando el menor *code-behind* posible.

8. **Revisión del código desarrollado en la anterior etapa**. Debido a que era la primera etapa del proceso de reingeniería software en la que se desarrollaba código para la migración, éste está expuesto a razonables fallos que deben controlarse y paliarse.

9. **Escasez de tiempo para la revisión del código desarrollado en la anterior etapa**. Debido a la planificación prevista por el equipo de trabajo, no se estimó ningún tiempo para la revisión de código de la etapa previa. Por el hecho de no reservar un tiempo razonable para ello, durante toda la presente etapa fue necesario la resolución de errores encontrados en el código de la anterior etapa sobre el que se trabajaba, al mismo tiempo que se desarrollaba el código necesario para la migración. Esto se agrava con el hecho de que ciertos errores no fueron resueltos por cuestiones de falta de tiempo, y resultaron ser un lastre durante toda la etapa.

10. **Escasez de tiempo para la documentación**. Debido al deseo por obtener prioritariamente un producto final, el trabajo de documentación no fue todo lo amplio y completo que debería haber sido. Esto puede resultar muy delicado ya que el conocimiento lo tienen las propias personas, sin estar reflejado en ningún lugar, arriesgándose al hecho de que las personas abandonen la organización o se olviden de cómo hicieron su trabajo en el pasado.

11. **Escasez de documentación en los objetos de las presentaciones**. En la mayoría de las ocasiones, los objetos que se pretendían migrar no disponían de una documentación adecuada que indicara exactamente la finalidad de esos objetos, así como todas las propiedades y funcionalidad de que disponían.

12. **Dispersión del conocimiento**. Ante la falta de documentación dentro de la organización, es el propio personal el que reúne una parte importante de información. El problema reside cuando los empleados dejan la empresa sin transmitir todo su conocimiento, o cuando los propios empleados no están de acuerdo en determinados aspectos. En algunos casos, el resultado es una notable pérdida de información.

13. **Se prima la inmediatez respecto a la calidad.** Se buscan los resultados inmediatos, descuidando en algunos casos, la calidad del propio trabajo. Esto resulta peligroso debido a que determinados fallos pueden repetirse durante todo el proceso de reingeniería por no haber estudiado correctamente el caso desde el principio.

CAPÍTULO IV. DESCRIPCIÓN DE LA SOLUCIÓN.

8. Gestión de la solución.

Como resulta obvio tras las indicaciones mencionadas en los capítulos anteriores, se considera completamente necesario definir un marco de trabajo que permita ejecutar el proceso de reingeniería del software de manera eficaz, de modo que la solución obtenida tras la migración sea idónea. En el presente apartado se muestra el framework seguido para la resolución del proceso de reingeniería.

8.1. Framework adoptado.

El marco de trabajo continuado ha sido el mismo que se utilizó en la anterior etapa, en la cual fue definido (*Cabezas, PFC, Junio 2008*) (*Colomo-Palacios, García-Crespo & Ruano-Mayoral. "EuroSPI", 2008*) debido a la necesidad de crear un framework que sirviese como base metodológica al proyecto, a la vez que corrigiera y previniese los defectos que se pudieran producir a lo largo de su desarrollo. Básicamente, los pasos dados para crear el framework fueron: análisis de todos los elementos que interviniesen o estuviesen relacionados con el proyecto; recopilación de todo el soporte informativo necesario para llevar a cabo el proyecto; definición del plan para servir como guía durante el progreso del proyecto; descomposición del proyecto con el fin de conseguir una granularidad que permitiese definir problemas e hitos abordables con un enfoque evolutivo; creación de un formato correcto y eficaz de documentación. Todo esto con la premisa de crear un framework lo suficientemente flexible para su correcta adaptación en función de los cambios. Este marco para el plan de migración tiene como objetivo principal la normalización del proceso de migración para poder obtener un producto final de calidad mediante un proceso robusto y correctamente definido.

La base para la realización de este marco se cimenta sobre la adaptación de dos metodologías complementarias que están vinculadas al desarrollo de sistemas de información (*Cabezas, PFC, Junio 2008*) (*Colomo-Palacios, García-Crespo & Ruano-Mayoral. "EuroSPI", 2008*). La primera de ellas, es el estándar de desarrollo de software de PSS-05-0 propuesto por la ESA (European Space Agency, Agencia Espacial Europea) (*ESA Board for Software Standardisation and Control, 1991*), utilizado como guía en los procesos de generación de código. La segunda metodología adaptada es la descrita en el documento *DoD Software Migration Planning* desarrollado en el seno del Software Engineering Institute (*Bergey, O'Brien and Smith, DoD Software Migration Planning 2001*), complementado con

una serie de recomendaciones y directrices recogidas en el *DoD Legacy System Migration Guidelines*, también desarrollado en el Software Engineering Institute. Se trata de un documento que sirve como base para la definición de los planes de migración del Departamento de Defensa de los Estados Unidos, y en él se identifican un conjunto de fases y tareas comunes a cualquier tarea de migración / reingeniería.

Este marco de trabajo se presenta de manera visual en la siguiente ilustración:

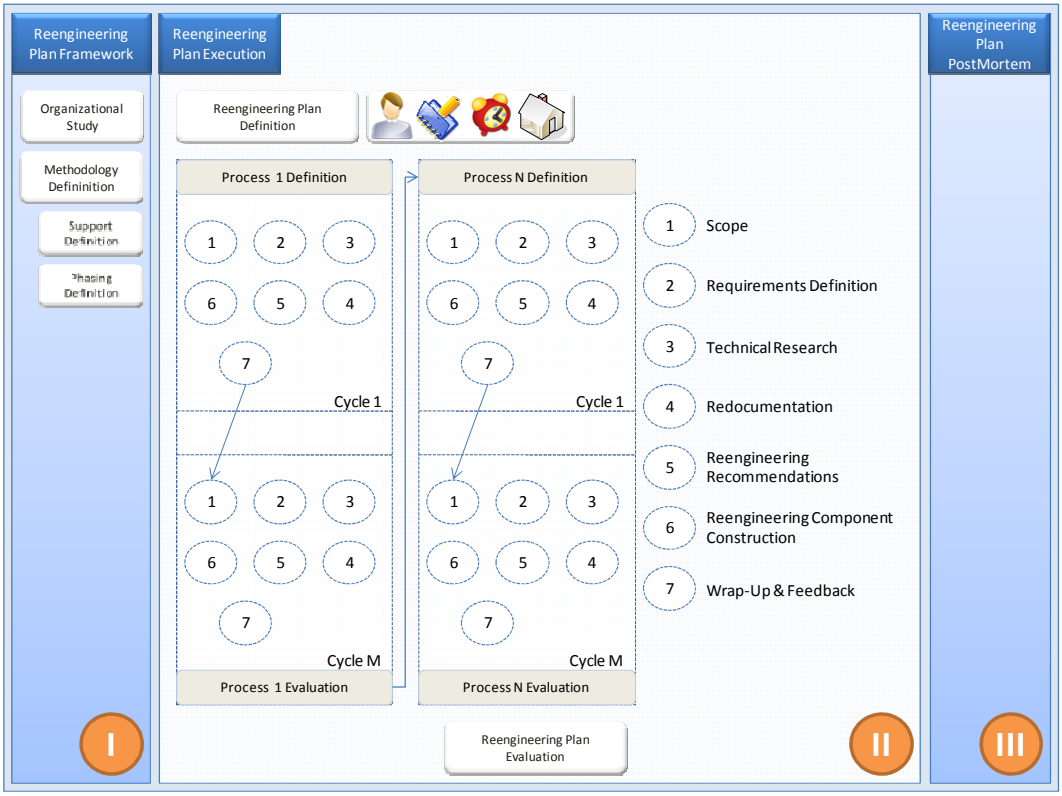


Ilustración 7 - Framework para la reingeniería

Como se puede observar en la ilustración previa, el marco de trabajo se divide en tres grandes bloques que quedan debidamente diferenciados temporalmente, siendo el primero previo a la ejecución del plan de migración, el segundo centrado en su ejecución, y el tercero y último posterior a la ejecución del mismo. Al mismo tiempo, estos bloques constan de fases y actividades que permiten descomponer el proyecto hasta conseguir establecer y distinguir en todo momento las tareas que deben realizar los miembros del equipo de trabajo. Además, permite ser consciente en cualquier instante en qué punto se encuentra el desarrollo del proyecto y qué porcentaje del mismo se lleva realizado hasta ese momento.

9. Aspectos relacionados.

La finalidad de este apartado consiste en mostrar todos aquellos aspectos requeridos para llegar a la solución del proceso de reingeniería. Para ello, en primer lugar se mostrarán temas tales como el calendario y el presupuesto del proyecto, y posteriormente se procederá a exponer la tecnología y herramientas empleadas para la consecución del mismo.

9.1. Calendario.

Todo proyecto con el que se pretenda llegar a un resultado idóneo, dispone de una planificación precisa y adecuada que debe ser llevada a cabo. Por lo tanto, realizar una buena planificación es una de las tareas iniciales más importantes que deben tenerse en cuenta. Lógicamente, la planificación puede cambiar durante el ciclo de vida del proyecto, por lo que no es una tarea estática, sino que con el paso del tiempo puede modificarse, siempre que dichas alteraciones sean apropiadas y permisibles.

En el caso del presente Proyecto Fin de Carrera, la planificación realizada inicialmente fue relativamente buena y no fue necesario modificarla en exceso, pero al tratarse de una migración de tecnología, muchas ocasiones exigían un mayor detenimiento en el análisis de ciertos objetos a migrar. Mientras que otras tareas para las que se estimaron un tiempo bastante amplio, no fue necesario hacer uso de todo ese tiempo y se consiguió finalizar el trabajo antes de tiempo. Gracias a poder haber seguido a tiempo dicha planificación, el resultado final consiguió ser el deseado.

Las ocupaciones con las que cuenta el proyecto son denominadas *tareas*:

- **La primera tarea** consiste en el estudio de la documentación del ciclo anterior, con el propósito de familiarizarme con el proyecto de reingeniería a realizar y entender a su vez el resultado que se pretendía obtener.
- **La segunda tarea** consiste en la integración del proyecto migrador con la aplicación propiedad de la empresa, que cuenta con la herramienta *Diseñador de presentaciones* que se requiere migrar.

- **La tercera tarea** se basa en el análisis de los componentes incluidos en la presentación objetivo que se pretende migrar, para saber el ámbito del proyecto y ser capaces de crear una planificación adecuada.
- **La cuarta etapa** es la más amplia, ya que necesita más del 85% del tiempo total del proyecto para su consecución. Lógicamente, esta etapa consiste en la migración de los objetos desde la tecnología propiedad de la empresa, a tecnología .NET, así como su correspondiente documentación.

A continuación se muestra la planificación de dichas tareas de las que consta el proyecto de migración, donde se observarán los nombres de las tareas y la duración de las mismas, así como su fecha de comienzo y fin. Si alguna de estas tareas necesita la anterior finalización de una de las tareas previas, dicha tarea prioritaria aparecerá como predecesora. Posteriormente se mostrará el correspondiente Diagrama de Gantt de dicha planificación.

Planificación.

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Proyecto KHAMIG (Ciclo 3)**	**110 días**	**lun 29/09/08**	**vie 27/02/09**	
2	Estudio documentación	5 días	lun 29/09/08	vie 03/10/08	
3	**Integración con la aplicación**	**7 días**	**lun 06/10/08**	**mar 14/10/08**	**2**
4	Análisis del ámbito del proyecto	3 días	lun 06/10/08	mié 08/10/08	
5	Modificar M4SysTask	1 día	jue 09/10/08	jue 09/10/08	
6	Crear M4NetBridge.dll con interfaz .COM	2 días	vie 10/10/08	lun 13/10/08	
7	Convertir M4Translator a .dll	1 día	mar 14/10/08	mar 14/10/08	
8	Pruebas	1 día	mar 14/10/08	mar 14/10/08	
9	**Analisis presentación UC3M_FRAMEWORK_I**	**3 días**	**mié 15/10/08**	**vie 17/10/08**	
10	Análisis de componentes incluidos en la presentación a migrar	3 días	mié 15/10/08	vie 17/10/08	
11	**MIGRACIÓN OBJETOS**	**95 días**	**lun 20/10/08**	**vie 27/02/09**	**3**
12	Documentos generales del proyecto (Scope, URD, SSD, etc.)	40 días	lun 05/01/09	vie 27/02/09	
13	**Auxiliares**	**83 días**	**lun 20/10/08**	**mié 11/02/09**	
14	TraductProp	2 días	lun 20/10/08	mar 21/10/08	
15	Control	1 día	lun 20/10/08	lun 20/10/08	
16	Constants	1 día	mar 21/10/08	mar 21/10/08	
17	ContextData	1 día	mar 21/10/08	mar 21/10/08	
18	Parser	2 días	mié 22/10/08	jue 23/10/08	
19	Utils	2 días	jue 23/10/08	vie 24/10/08	
20	Translator	1 día	vie 24/10/08	vie 24/10/08	
21	MigrateMethods	78 días	lun 27/10/08	mié 11/02/09	
22	**Presentation**	**1 día**	**lun 27/10/08**	**lun 27/10/08**	
23	Análisis y documentación	1 día	lun 27/10/08	lun 27/10/08	
24	Migración	1 día	lun 27/10/08	lun 27/10/08	
25	**Form**	**3 días**	**lun 27/10/08**	**mié 29/10/08**	
26	Análisis y documentación	1 día	lun 27/10/08	lun 27/10/08	
27	Migración	3 días	lun 27/10/08	mié 29/10/08	
28	**Button**	**4 días**	**mar 28/10/08**	**vie 31/10/08**	
29	Análisis y documentación	3 días	mar 28/10/08	jue 30/10/08	
30	Migración	3 días	mié 29/10/08	vie 31/10/08	

Ilustración 8 - Planificación (I)

31	✓	**Textbox**	**4 días**	**lun 03/11/08**	**jue 06/11/08**	
32	✓	Análisis y documentación	1 día	lun 03/11/08	lun 03/11/08	
33	✓	Migración	4 días	lun 03/11/08	jue 06/11/08	
34	✓	**Textarea**	**2 días**	**mar 04/11/08**	**mié 05/11/08**	
35	✓	Análisis y documentación	1 día	mar 04/11/08	mar 04/11/08	
36	✓	Migración	2 días	mar 04/11/08	mié 05/11/08	
37	✓	**Groupbox**	**2 días**	**jue 06/11/08**	**vie 07/11/08**	
38	✓	Análisis y documentación	1 día	jue 06/11/08	jue 06/11/08	
39	✓	Migración	2 días	jue 06/11/08	vie 07/11/08	
40	✓	**Includepanel y Panel**	**3 días**	**lun 10/11/08**	**mié 12/11/08**	
41	✓	Análisis y documentación	1 día	lun 10/11/08	lun 10/11/08	
42	✓	Migración	3 días	lun 10/11/08	mié 12/11/08	
43	✓	**Itemlabel y Label**	**3 días**	**mar 11/11/08**	**jue 13/11/08**	
44	✓	Análisis y documentación	1 día	mar 11/11/08	mar 11/11/08	
45	✓	Migración	3 días	mar 11/11/08	jue 13/11/08	
46	✓	**Image**	**1 día**	**vie 14/11/08**	**vie 14/11/08**	
47	✓	Análisis y documentación	1 día	vie 14/11/08	vie 14/11/08	
48	✓	Migración	1 día	vie 14/11/08	vie 14/11/08	
49	✓	**Toolbars (visualización)**	**10 días**	**lun 17/11/08**	**vie 28/11/08**	**28**
50	✓	Análisis	1 día	lun 17/11/08	lun 17/11/08	
51	✓	Toolbar	1 día	lun 17/11/08	lun 17/11/08	
52	✓	ToolbarButton	1 día	lun 17/11/08	lun 17/11/08	
53	✓	ToolbarDataGroup	4 días	lun 17/11/08	jue 20/11/08	
54	✓	ToolbarEditGroup	3 días	vie 21/11/08	mar 25/11/08	
55	✓	ToolbarRootNavigationGroup	2 días	mié 26/11/08	jue 27/11/08	
56	✓	Toolbarparamsgroup	1 día	vie 28/11/08	vie 28/11/08	
57	✓	Toolbarseparator	1 día	vie 28/11/08	vie 28/11/08	
58	✓	Documentación	4 días	mar 25/11/08	vie 28/11/08	

Ilustración 9 - Planificación (II)

59	✓	**Control de eventos**	**10 días**	**lun 01/12/08**	**vie 12/12/08**	
60	✓	Evclick	5 días	lun 01/12/08	vie 05/12/08	
61	✓	Action_dataprops	6 días	mié 03/12/08	mié 10/12/08	
62	✓	Action_value	2 días	mié 10/12/08	jue 11/12/08	
63	✓	Documentación	5 días	lun 08/12/08	vie 12/12/08	
64	✓	**Toolbars (funcionalidad)**	**10 días**	**lun 15/12/08**	**vie 26/12/08**	**49;59**
65	✓	Análisis	1 día	lun 15/12/08	lun 15/12/08	
66	✓	Nodestatus	3 días	lun 15/12/08	mié 17/12/08	
67	✓	Patternbox	1 día	jue 18/12/08	jue 18/12/08	
68	✓	ToolbarDataGroup	2 días	jue 18/12/08	vie 19/12/08	
69	✓	ToolbarEditGroup	3 días	vie 19/12/08	mar 23/12/08	
70	✓	ToolbarRootNavigationGroup	4 días	mar 23/12/08	vie 26/12/08	
71	✓	ToolbarButton	1 día	vie 26/12/08	vie 26/12/08	
72	✓	Toolbarparamsgroup	1 día	vie 26/12/08	vie 26/12/08	
73	✓	Documentación	5 días	lun 22/12/08	vie 26/12/08	
74	✓	**Menús**	**2 días**	**lun 29/12/08**	**mar 30/12/08**	
75	✓	Análisis y documentación	1 día	lun 29/12/08	lun 29/12/08	
76	✓	Menu	2 días	lun 29/12/08	mar 30/12/08	
77	✓	Menuitem	2 días	lun 29/12/08	mar 30/12/08	
78	✓	Includemenubar	2 días	lun 29/12/08	mar 30/12/08	
79	✓	Menuparamsgroup	2 días	lun 29/12/08	mar 30/12/08	
80	✓	**Tabs**	**3 días**	**mié 31/12/08**	**vie 02/01/09**	
81	✓	Análisis y documentación	1 día	mié 31/12/08	mié 31/12/08	
82	✓	Tab	3 días	mié 31/12/08	vie 02/01/09	
83	✓	Tabstrip	3 días	mié 31/12/08	vie 02/01/09	

Ilustración 10 - Planificación (III)

84	✓	**Mimic**	**1 día**	**lun 05/01/09**	**lun 05/01/09**	
85	✓	Análisis y documentación	1 día	lun 05/01/09	lun 05/01/09	
86	✓	Migración	1 día	lun 05/01/09	lun 05/01/09	
87	✓	**Splitts**	**8 días**	**lun 05/01/09**	**mié 14/01/09**	
88	✓	Análisis	1 día	lun 05/01/09	lun 05/01/09	
89	✓	Splittblock	2 días	mar 06/01/09	mié 07/01/09	
90	✓	Splitthorizontal	5 días	mié 07/01/09	mar 13/01/09	
91	✓	Splittvertical	3 días	lun 12/01/09	mié 14/01/09	
92	✓	Documentación	3 días	lun 12/01/09	mié 14/01/09	
93	✓	**Tree**	**4 días**	**jue 15/01/09**	**mar 20/01/09**	
94	✓	Análisis	1 día	jue 15/01/09	jue 15/01/09	
95	✓	Tree	3 días	jue 15/01/09	lun 19/01/09	
96	✓	Treenode	4 días	jue 15/01/09	mar 20/01/09	
97	✓	Treeview	4 días	jue 15/01/09	mar 20/01/09	
98	✓	Documentación	2 días	lun 19/01/09	mar 20/01/09	
99	✓	**ScrollPanel**	**3 días**	**mié 21/01/09**	**vie 23/01/09**	
100	✓	Análisis y documentación	1 día	mié 21/01/09	mié 21/01/09	
101	✓	Migración	3 días	mié 21/01/09	vie 23/01/09	
102	✓	**Table**	**3 días**	**lun 26/01/09**	**mié 28/01/09**	
103	✓	Análisis y documentación	3 días	lun 26/01/09	mié 28/01/09	
104	✓	Migración	2 días	mar 27/01/09	mié 28/01/09	
105	✓	**Listgroup**	**21 días**	**jue 29/01/09**	**jue 26/02/09**	
106	✓	Análisis	5 días	jue 29/01/09	mié 04/02/09	
107	✓	Listgroup	2 días	mar 03/02/09	mié 04/02/09	
108	✓	Function	2 días	jue 05/02/09	vie 06/02/09	
109	✓	Extends	1 día	lun 09/02/09	lun 09/02/09	
110	✓	Action_preload	1 día	lun 09/02/09	lun 09/02/09	
111	✓	Funcionalidad	12 días	mar 10/02/09	mié 25/02/09	107;108;109;110
112	✓	Documentación	5 días	vie 20/02/09	jue 26/02/09	

Ilustración 11 - Planificación (IV)

Diagrama de Gantt.

En primer lugar se muestra el diagrama de Gantt con las tareas resumidas, para poder visualizar claramente el tiempo que es necesario emplear para cumplimentar cada una de ellas. Posteriormente se mostrarán los diagramas de Gantt con las tareas disgregadas.

Ilustración 12 - Diagrama de Gantt

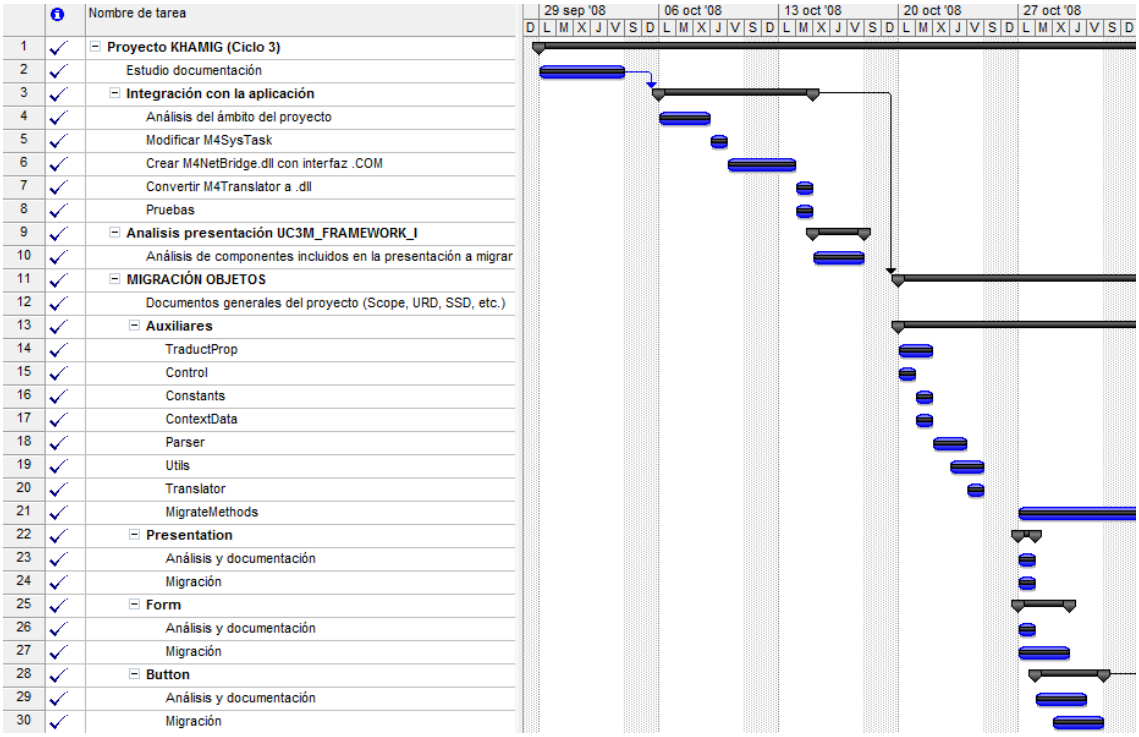


Ilustración 13 - Diagrama de Gantt (I)

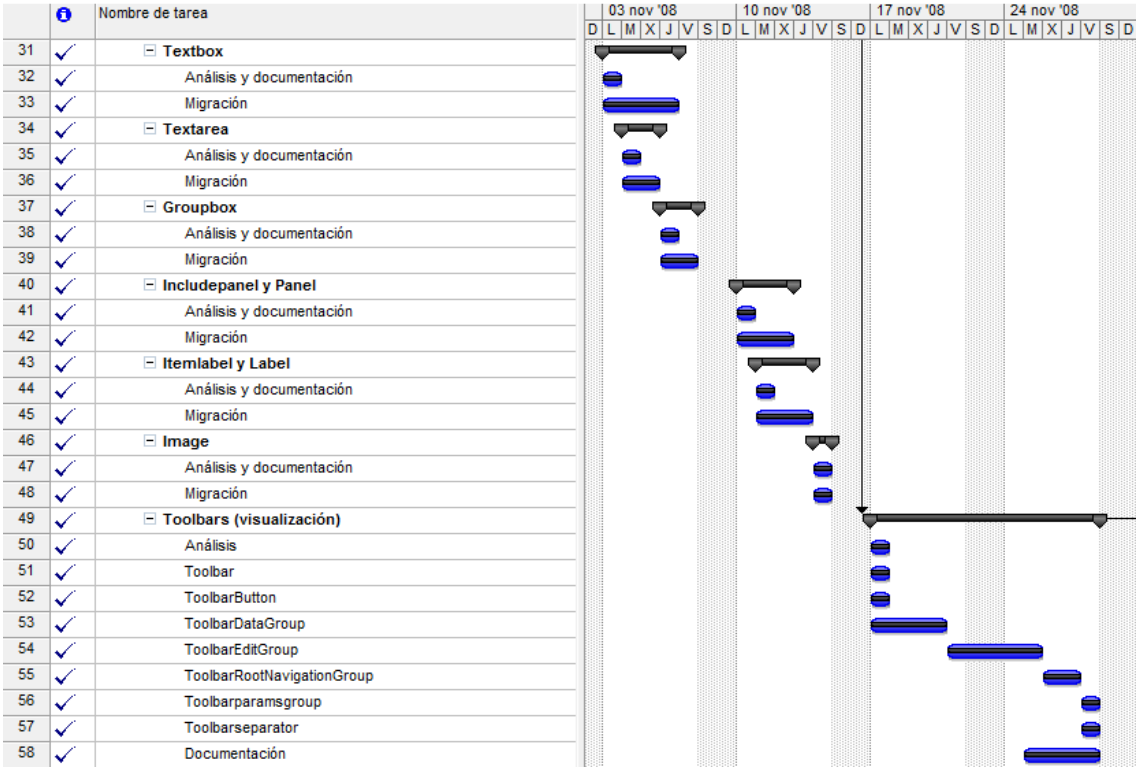


Ilustración 14 - Diagrama de Gantt (II)

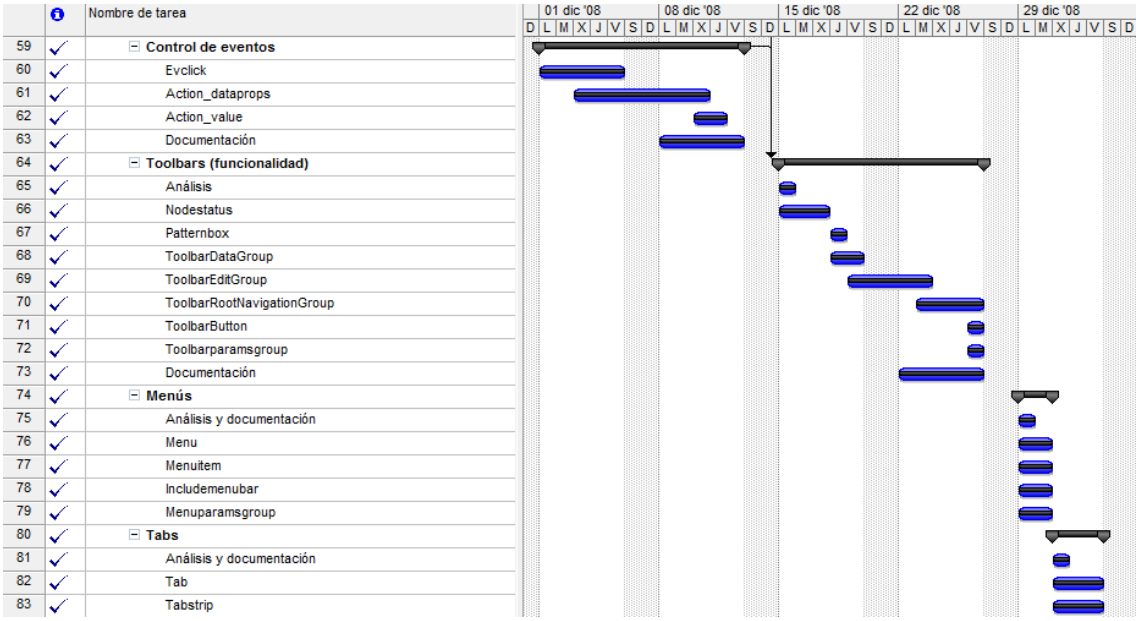


Ilustración 15 - Diagrama de Gantt (III)

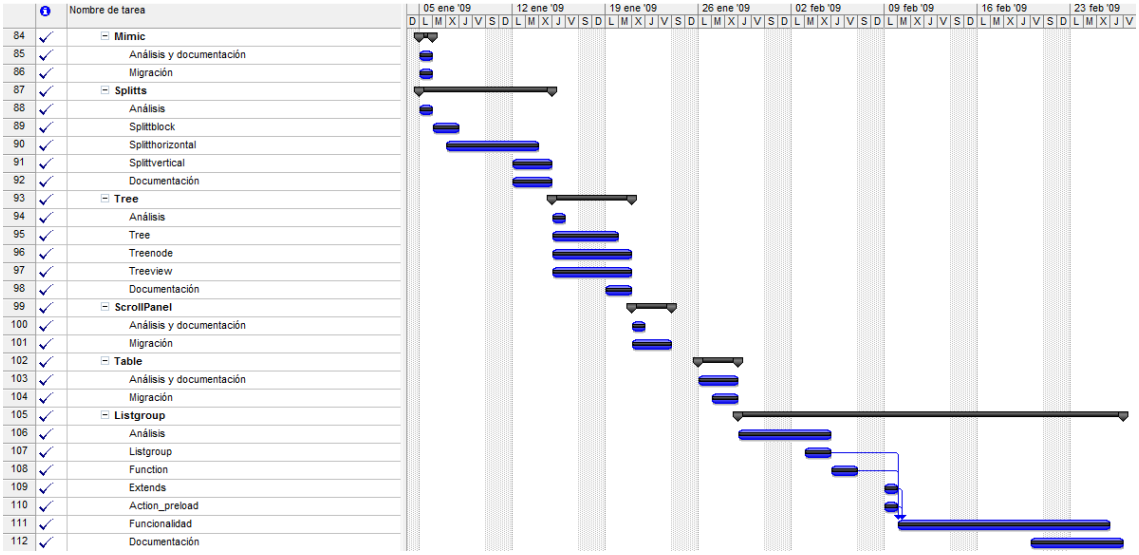


Ilustración 16 - Diagrama de Gantt (IV)

Consideraciones.

La tarea *MigrateMethods* contenida en el apartado *Auxiliares*, en *Migración de objetos*, tiene una duración de 78 días ya que esta tarea consiste en la creación de una clase en C# encargada de albergar los métodos de migración de las propiedades de todos los objetos migrados durante el proceso de migración. En este sentido, dicha clase deberá ser continuamente modificada hasta la finalización de la migración.

Como puede observarse en los anteriores diagramas, todas las tareas planificadas se encuentran finalizadas. Esto no significa que el objeto en cuestión haya sido migrado en su completitud, sino que la tarea de migración estipulada por el equipo de desarrollo para dicho control ha sido finalizada con éxito. Aunque es cierto que gran parte de los objetos mostrados están totalmente migrados, muchos otros requieren de una revisión y un análisis en mayor profundidad, acciones que serán tratadas en el siguiente ciclo del proyecto.

Por último, se considera conveniente comentar que para realizar la migración de ciertos objetos, resulta ventajoso haber migrado ciertos objetos previamente. Sin embargo, estos últimos no aparecen en la planificación como acciones predecesoras de los primeros, ya que para realizar la migración no es estrictamente necesaria la previa migración de dichos objetos.

9.2. Presupuesto.

Concluido el calendario, se procede a ofrecer un presupuesto, ya que éste depende lógicamente del tiempo de vida del proyecto. La fecha de inicio del ciclo desarrollado fue el 29 de Septiembre de 2008, y su fecha de finalización el 27 de Febrero de 2009, lo que suponen 110 días de trabajo en los 5 meses de duración de esta fase del proyecto. El número total de horas a trabajar en estos tres meses ha sido: 880 horas.

Plazo	29 de Septiembre de 2008	27 de Febrero de 2009
Número total de días	110 días	
Número total de horas	**880 HORAS**	

Tabla 2 - Coste temporal

Para calcular el presupuesto habría que sumar al coste monetario de los recursos humanos (en este caso, el propio proyectando), el coste del software utilizado. El coste software consistiría básicamente en la adquisición de equipos y licencias de tecnología y herramientas, explicadas estas dos últimas en los dos siguientes apartados. Sin embargo, los equipos utilizados, así como la tecnología y herramientas empleadas han supuesto un coste nulo, debido a que la Universidad Carlos III y la empresa en la que ha sido desarrollado el proyecto han proporcionado dichas utilidades. De esta manera, el coste monetario está basado en los recursos humanos.

IVA	Rol	Dedicación	Coste	TOTAL
Sin IVA	Ingeniero del software	880 horas	30 € / hora	**26.400 €**
Con IVA (16%)	Ingeniero del software	880 horas	34,8 € / hora	**30.624 €**

Tabla 3 - Coste monetario

9.3. Tecnología.

Para el desarrollo del proceso de reingeniería, se ha adoptado la filosofía de utilizar la tecnología más novedosa existente en el mercado, con el fin de que el producto final ofertado fuese novedoso a la par que adaptado totalmente a las nuevas tecnologías.

En este apartado acerca de la tecnología empleada, se procederá a explicar el sistema operativo utilizado para el desarrollo del proceso de reingeniería, así como el entorno .NET utilizado para la ejecución del proceso de migración.

Entorno Microsoft .NET Framework.

En Junio de 2000, la firma de Redmond presentó un entorno independiente del lenguaje, concebido para facilitar el desarrollo de aplicaciones para Windows capaces de trabajar y comunicarse de forma segura y sencilla. Casi una década después, el 70% de los desarrolladores españoles utiliza .NET.

En la actualidad la presencia de este entorno de programación de Microsoft en el mercado de las herramientas de desarrollo de aplicaciones para Windows es abrumadora. Y es que esta plataforma se ha consolidado como una alternativa eficaz a J2EE (Java 2 Platform Enterprise Edition) de Sun que, además, proporciona importantes ventajas a la hora de diseñar aplicaciones distribuidas.

Los componentes que conforman este entorno de desarrollo se denominan .NET Framework y consisten en el CLR (Common Language Runtime) o lenguaje común en tiempo de ejecución, una jerarquía de librerías de clases y soporte para diferentes tipos de aplicaciones, entre las que contempla herramientas modeladas con la arquitectura cliente-servidor tradicional, así como aplicaciones y servicios web. Además, .NET proporciona librerías que facilitan el acceso a bases de datos y código de gestión de XML, no en vano este metalenguaje se ha posicionado en esta plataforma como un estándar de facto para la interoperabilidad de aplicaciones.

.NET Framework es un componente de Windows para la generación, implantación, y ejecución de aplicaciones en un entorno con un extenso conjunto de lenguajes de programación. Asimismo, proporciona a los desarrolladores gran parte de la estructura requerida para la generación de software, permitiendo que estos puedan poner un mayor foco sobre el código lógico específico del producto en desarrollo.

.NET Framework permite la construcción rápida de aplicaciones conectadas que ofrecen experiencias de usuario increíbles, ya que ofrecen bloques de construcción (software pre-fabricado) que resuelven las tareas de programación más frecuentes. Las aplicaciones conectadas construidas sobre los modelos de negocio de .NET Framework procesan de manera efectiva y facilitan la integración de sistemas en entornos heterogéneos (*www.microsoft.com*).

.NET Framework consta de tres partes principales (*MSDN 2008*):

- *Common Language Runtime (CLR)*. El CLR se encuentra en el corazón del entorno, de hecho es una máquina virtual encargada de proporcionar una capa de abstracción entre el bytecode de .NET y la plataforma hardware sobre la que se ejecuta Desempeña una función tanto en tiempo de ejecución como en el proceso de desarrollo de componentes. En tiempo de ejecución, el CLR es responsable de la administración de memoria, iniciar y detener procesos y subprocesos, cubrir las dependencias entre estos, etc. Durante la etapa de desarrollo, el CLR cobra relevancia simplificando el trabajo del desarrollador al contribuir con características del tipo de la administración del ciclo de vida, la nomenclatura de tipos, y la gestión de excepciones entre distintos tipos de lenguajes. Esto permite reducir la cantidad de código que debe escribir un desarrollador para convertir sus productos en componentes reutilizables.
- *Clases de programación unificadas*. Presenta un conjunto unificado, orientado a objetos, jerárquico y extensible a bibliotecas de clases (API). El entorno de trabajo permite unificar modelos como las Microsoft Foundation Classes y las Windows Foundation Classes para permitir a programadores de diversos lenguajes poder tener acceso a las bibliotecas de clases. La creación de un conjunto de API comunes evita limitaciones en la utilización de los lenguajes de programación en el entorno de trabajo, dejando al desarrollador la elección del lenguaje que desee utilizar sin ningún tipo de restricción.
- *ASP.NET*. ASP.NET construye las clases de programación de .NET Framework, proporcionando un conjunto de controles e infraestructura que contribuye a la generación de aplicaciones Web.

A continuación se resumen algunas de las ventajas más importantes que proporciona el entorno (o plataforma) .NET Framework (*Manual electrónico sobre la plataforma .NET, 2004*):

- *Código administrado*: el CLR realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
- *Interoperabilidad multilenguaje*: El código puede ser escrito en cualquier lenguaje compatible con .NET ya que siempre se compila en código intermedio (MSIL).
- *Compilación just-in-time*: El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
- *Garbage collector*: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (garbage collector). El CLR detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene porqué liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente.
- *Seguridad de acceso al código*: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.
- *Despliegue*: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones.

Es cierto que no todo son ventajas, ya que procesos como la recolección de basura de .NET o la administración de código, introducen factores de sobrecarga que repercuten en la demanda de más requisitos del sistema. Como se ha visto, el código administrado proporciona una mayor velocidad de desarrollo y mayor seguridad de que el código sea bueno. En contrapartida, el consumo de recursos durante la ejecución es mucho mayor, aunque con los procesadores actuales esto cada vez es menos inconveniente. Además, el nivel de administración del código dependerá en gran medida del lenguaje que utilicemos para programar. Por ejemplo, mientras que Visual Basic .Net es un lenguaje totalmente administrado, C-Sharp permite la administración de código de forma manual, siendo por defecto también un lenguaje administrado, mientras que C++ es un lenguaje no

administrado en el que se tiene un control mucho mayor del uso de la memoria que hace la aplicación.

En nuestro caso, ha sido utilizado el entorno .NET Framework 3.5, ya que construye más que su predecesor .NET Framework 3.0. Las mejoras están basadas en áreas fundamentales como la biblioteca básica de clases, Windows Workflow Foundation, Windows Communication Foundation, Windows CardSpace y Windows Presentation Foundation. Con respecto a éste último, WPF es una de las novedosas tecnologías de Microsoft y uno de los pilares de Windows Vista, aunque su uso no es exclusivo de este sistema operativo ya que puede ser utilizado también sobre Windows XP. WPF es toda una plataforma que da soporte para poder colocar en sus aplicaciones una amplia riqueza visual e interactiva, e integrarlas fácilmente con la lógica del negocio.

Sistema operativo Windows.

El entorno en el que ha sido desarrollado el proceso de reingeniería para este proyecto de migración, ha sido el sistema operativo Windows. La empresa que cuenta con la herramienta que se pretende migrar trabaja con gran cantidad de librerías e instrumentos pertenecientes a Microsoft, y el entorno de ejecución de las presentaciones a migrar no es otro que Windows. En este sentido, se ha desechado la posibilidad de utilización de otros sistemas operativos para el desarrollo de la migración.

Aparte de ello, mundialmente es abrumadora la diferencia entre el uso del sistema operativo Windows y los demás sistemas operativos, por lo que el uso de esta plataforma resulta aún más evidente. A continuación se muestra una estimación del uso actual de sistemas operativos según una muestra de ordenadores con acceso a Internet (*www.w3counter.com, 2009*). Como se puede observar, el uso de Windows frente a los demás sistemas es muy superior:

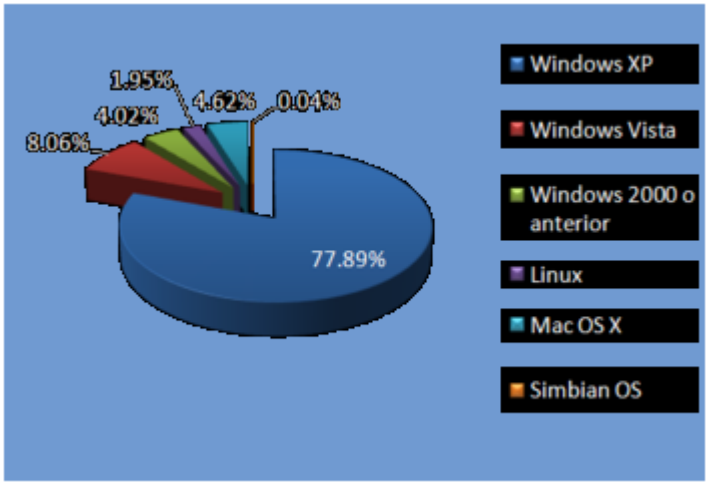


Ilustración 17 - Estimación del uso de los distintos sistemas operativos

Para el desarrollo de la migración, se ha utilizado tanto Windows Vista como Windows XP, obteniendo el mismo producto en ambas plataformas. En concreto, los desarrolladores externos utilizaban el sistema operativo Windows Vista, mientras que los empleados de la empresa utilizaban por el contrario Windows XP.

9.4. Herramientas.

Existen ciertas herramientas software necesarias para poder llevar a cabo el proceso de reingeniería mostrado en el presente Proyecto Fin de Carrera. Estas herramientas han permitido que el proceso de migración se pudiese realizar de un modo más rápido, cómodo y, sobretodo, eficaz. Todas ellas están basadas en el entorno Windows. Se tratan de Microsoft Visual Studio 2008, Microsoft Visual SourceSafe 8.0, el paquete Microsoft Office 2007, y distinto software ofrecido por Windows como las aplicaciones Microsoft Paint o Microsoft Notepad. Además, han sido completamente imprescindibles los navegadores Internet Explorer 7.0 y Mozilla Firefox 3.0, para el acceso a Internet con el propósito de recopilar información para ejecutar la migración, y también de completar la documentación online.

Aparte de todo ello, lógicamente ha sido necesario utilizar el Sistema de información desarrollado por el cliente, del cual se pretende realizar la migración de las interfaces de usuario creadas mediante una de sus herramientas, denominada "Diseñador de presentaciones".

En este apartado se procederá a explicar en qué consisten dichas herramientas y su funcionalidad para el proyecto.

Microsoft Visual Studio 2008

Microsoft Visual Studio 2008 cumple con la visión de Microsoft sobre aplicaciones inteligentes, al permitir que los desarrolladores creen rápidamente aplicaciones conectadas con la más alta calidad y con atractivas experiencias de usuario (*www.microsoft.com*).

Con Visual Studio 2008, las organizaciones encuentran que ahora es más fácil capturar y analizar información, y por lo tanto tomar decisiones de negocio más efectivas. Gracias a Visual Studio 2008, las organizaciones de todo tamaño podrán crear rápidamente aplicaciones más seguras, confiables y administrables, capaces de aprovechar mejor las características de Windows Vista y de Office 2007.

Visual Studio 2008 ofrece avances clave para desarrolladores en función de los siguientes cuatro pilares:

- ***Desarrollo rápido de aplicaciones***: para que los desarrolladores puedan crear rápidamente software moderno, Visual Studio 2008 ofrece funciones de programación y de datos mejoradas. Visual Studio 2008 también brinda la posibilidad de apuntar a distintas versiones de .NET Framework desde el mismo entorno de desarrollo. Por lo tanto, los desarrolladores podrán construir aplicaciones que apunten a .NET Framework 2.0, 3.0 ó 3.5, y así podrán admitir una amplia variedad de proyectos en un mismo entorno
- ***Colaboración eficiente entre equipos***: Visual Studio 2008 propone ofertas expandidas y mejoradas que ayudan a mejorar la colaboración entre equipos de desarrollo, incluidas herramientas que colaboran con la integración entre profesionales especializados en bases de datos y diseñadores gráficos.
- ***Innovación en experiencias de usuario***: Visual Studio 2008 ofrece nuevas herramientas que aceleran la creación de aplicaciones conectadas con las últimas plataformas, incluidas la Web, Windows Vista, Office 2007, SQL Server 2008 y Windows Server 2008. Para la Web, ASP.NET AJAX y otras nuevas tecnologías permitirán que los desarrolladores creen rápidamente una nueva generación de experiencias más eficientes, interactivas y personalizadas.

- ***Uso de Microsoft .NET Framework 3.5***: juntos, Visual Studio y .NET Framework reducen la necesidad de código en común, disminuyen los tiempos de desarrollo, y permiten que los desarrolladores se concentren en resolver problemas comerciales. El entorno .NET Framework 3.5 construye más que .NET Framework 3.0. Las mejoras fueron aplicadas a áreas fundamentales como la biblioteca básica de clases, Windows Workflow Foundation, Windows Communication Foundation, Windows CardSpace y **Windows Presentation Foundation**.

Microsoft Visual SourceSafe 8.0

Microsoft Visual SourceSafe es una herramienta de control de versiones que forma parte de Microsoft Visual Studio (MVS). Para las personas que desarrollan programas bajo el sistema operativo Windows, resulta una herramienta útil ya que se integra fuertemente con el entorno de desarrollo integrado (o IDE) de MVS, permitiendo un manejo relativamente simple de versiones sobre una computadora individual y en equipos de trabajo relativamente pequeños.

Esta herramienta ha sido completamente imprescindible para el desarrollo del código necesario para la migración, ya que automáticamente, al estar integrado con MVS, cuando uno de los integrantes del equipo de trabajo se encontraba modificando el código de un archivo, se bloqueaba a los demás miembros la posibilidad de modificarlo al mismo tiempo. Además, las nuevas versiones creadas tras los cambios oportunos, podían compararse con las anteriores versiones del archivo y recuperar así el código anterior, en caso de que fuese necesario.

Como se comentó anteriormente, Microsoft Visual SourceSafe es un sistema de control de versiones en el nivel de archivos, que permite a muchos tipos de organizaciones trabajar en distintas versiones de un proyecto al mismo tiempo. Esta funcionalidad es especialmente ventajosa en un entorno de desarrollo de software, donde se usa para mantener versiones de código paralelas. Sin embargo, el producto también se puede utilizar para mantener archivos en cualquier otro tipo de equipo (*MSDN, 2009*).

Visual SourceSafe admite el desarrollo multiplataforma al permitir la edición y el uso compartido de los datos. Se ha diseñado para controlar los problemas de seguimiento y portabilidad que implica mantener una base de control de código fuente, como una base de código de software, en varios sistemas operativos. Para los desarrolladores, Visual SourceSafe aloja código reutilizable u orientado a

objetos. Asimismo, facilita el seguimiento de las aplicaciones que utilizan módulos de código concretos.

Visual SourceSafe incluye, como mínimo, las siguientes funciones:

- Ayuda al equipo a evitar la pérdida accidental de archivos.
- Permite realizar un seguimiento de las versiones anteriores de un archivo.
- Admite la bifurcación, el uso compartido, la combinación y la administración de versiones de archivos.
- Realiza el seguimiento de las versiones de proyectos completos.
- Realiza el seguimiento del código modular (un archivo que se reutiliza, o se comparte, en varios proyectos).

Control de versiones y uso compartido de archivos: Visual SourceSafe permite compartir archivos entre proyectos de forma rápida y eficaz. La organización de los archivos en proyectos hace que la coordinación de los equipos sea un proceso intuitivo. Cuando se agrega un archivo a Visual SourceSafe, este archivo se almacena en la base de datos y queda a disposición de otros usuarios. Los cambios realizados en él se guardan para que cualquier usuario pueda recuperar una versión anterior en todo momento. Los miembros de un equipo de trabajo podrán ver la última versión de un archivo, realizar cambios en sus copias locales y guardar nuevas versiones en la base de datos. Cuando un conjunto de archivos está listo para entregarse, Visual SourceSafe permite compartir y obtener las distintas versiones del conjunto con facilidad y seguridad.

Extensibilidad: Mediante las interfaces de automatización de Visual SourceSafe, un usuario puede escribir las extensiones basadas en Visual SourceSafe que necesite su entorno. Estas extensiones se suelen proporcionar en forma de aplicaciones independientes escritas en las interfaces de automatización. También se puede ampliar la funcionalidad de Visual SourceSafe si se escribe un complemento que sea compatible con el entorno de desarrollo integrado (IDE) del programa de terceros que ejecutará el paquete de software.

Desarrollo paralelo: Visual SourceSafe admite el desarrollo paralelo y las técnicas de desarrollo multiplataforma. Una compatibilidad tal permite que todos los miembros del equipo terminen las distintas partes y versiones de un proyecto al mismo tiempo, en lugar de tener que esperar a que los otros usuarios terminen algunas tareas. Se admiten las operaciones de combinación de archivos en dos o tres direcciones, y Visual SourceSafe incluye varios mecanismos para resolver los

conflictos resultantes de la combinación. Las operaciones de combinación de los archivos permiten trabajar independientemente sin necesidad de sincronizar los cambios con los realizados por otros usuarios.

Con fines de compatibilidad con las operaciones paralelas, Visual SourceSafe también incluye una función de promoción de etiquetas que permite pasar los archivos necesarios a las diferentes versiones de un proyecto. Asimismo, admite el uso de operaciones de uso compartido, fijación y bifurcación para el desarrollo paralelo en un proyecto durante un período de tiempo prolongado.

Compatibilidad para los desarrolladores: Cada vez más, los desarrolladores obtienen acceso a las funciones de Visual SourceSafe desde sus entornos de desarrollo en programas de terceros. Visual SourceSafe se puede integrar con toda facilidad en Visual Studio y otras herramientas de desarrollo como Microsoft Access. Visual SourceSafe admite un entorno de desarrollo de diversas formas mediante:

- La definición de directivas de carpeta para habilitar escenarios de desarrollo en grupo.
- Las correcciones de errores.
- La transición sencilla a una nueva versión de un proyecto existente.
- Las generaciones por lotes o nocturnas.
- La automatización de los eventos de control de código fuente.
- El acceso a las interfaces de automatización.
- El control de código fuente en conexiones lentas.
- La configuración de nuevos proyectos para el desarrollo Web dividido.
- La incorporación de nuevos desarrolladores Web al proyecto Web de un equipo existente.
- El seguimiento de módulos de programación para permitir código reutilizable u orientado a objetos.

Mantenimiento de bases de datos

Visual SourceSafe proporciona una serie de herramientas de mantenimiento de bases de datos muy útiles que permiten que éstas funcionen de forma eficaz y segura. Admite el almacenamiento y la restauración mediante asistentes de uso sencillo, así como varias utilidades de mantenimiento basadas en la línea de comandos.

Paquete Microsoft Office 2007

El paquete Microsoft Office consiste en un conjunto de software compuesto básicamente por aplicaciones de procesamiento de textos, plantillas de cálculo y programa para presentaciones. Las aplicaciones utilizadas en este proyecto han sido:

- ***Microsoft Office Word 2007***. Utilizado para la realización de la mayoría de la documentación. Utiliza un nuevo formato basado en XML llamado .DOCX, pero también tiene la capacidad de guardar y abrir documentos en el formato DOC.
- ***Microsoft Office Excel 2007***. Utilizado para la elaboración de varias tablas que aparecen en los diferentes documentos del proyecto. Es un programa de hoja de cálculo que, al igual que Microsoft Word, posee actualmente un mercado dominante.
- ***Microsoft Office Outlook 2007***. Es un administrador de información personal y un complejo cliente de correo electrónico. Utilizado como herramienta de correo electrónico interno en la empresa, además de agenda para la convocación de reuniones.
- ***Microsoft Office PowerPoint 2007***. Utilizado para la creación de diversos gráficos e ilustraciones utilizados como complemento en este proyecto, además del desarrollo de las presentaciones realizadas.
- ***Microsoft Office Project 2007***. Utilizado para la creación y seguimiento de la planificación del proyecto, así como la asignación de recursos a las tareas.

Sistema de información desarrollado por el cliente

Este sistema de información se trata de una plataforma que nace una década atrás, en la cual estuvieron implicadas más de dos mil personas. Esta solución integral recoge dos puntos que resaltan sobre el resto, las personas y su conocimiento. Esta solución tiene un carácter diferenciador con respecto a otros de características similares existentes en el mercado debido a su mayor especialización, hecho que motiva su presencia en más de 1200 organizaciones a lo largo de más de 80 países, con una carga de más de 15 millones de empleados.

Este software consta de una herramienta consistente en un editor de presentaciones, que facilita la gestión de personal por parte de todas estas empresas. Actualmente, existen un número muy elevado de presentaciones, en su mayoría de mantenimiento, que se ajustan a las necesidades que requiere cada cliente. Dichas presentaciones están basadas en una tecnología anticuada, por lo que se desea su migración a WPF para aprovechas las ventajas del entorno .NET.

Este sistema, y más concretamente su editor de presentaciones, ha sido objeto de estudio por parte del equipo de trabajo durante el primer ciclo de este proyecto y también durante gran parte del segundo ciclo, lo que facilitó el trabajo de investigación en este tercer ciclo.

Su principal funcionalidad en este ciclo que se muestra en el presente Proyecto Fin de Carrera, ha sido la creación de presentaciones sencillas como soporte de pruebas, para poder comprobar que el comportamiento de las presentaciones migradas era exactamente idéntico al de las creadas en el editor de presentaciones de la herramienta. Gracias a estas pequeñas presentaciones, se puede construir poco a poco la herramienta migradora, de tal modo que finalmente se puedan llegar a migrar todas las presentaciones existentes y nuevas creadas con la herramienta "Diseñador de presentaciones".

También ha sido muy útil la ayuda contenida en el sistema de información para poder entender la funcionalidad de muchos objetos contenidos en las presentaciones, así como la funcionalidad de las propiedades de dichos objetos.

10. Desarrollo de la solución.

Una vez explicados el framework adoptado y los aspectos relacionados con el desarrollo del proyecto, en el presente apartado se procederá a explicar cómo se ha conseguido llegar a la solución de la migración. Esta solución consiste en una serie de pasos con los cuales será posible obtener los objetos contenidos en una presentación candidata a ser migrada, y transformarlos a tecnología .NET (WPF), con el fin de obtener una presentación con las mismas funcionalidades que la presentación inicial, pero con las ventajas que ofrece el entorno .NET.

En el apartado 9.3, asignado a la tecnología utilizada en el proyecto, se explica en qué consiste la tecnología .NET. A continuación, se procede a explicar con mayor detalle conceptos necesarios para el correcto entendimiento del proyecto. Esto es, en qué consiste WPF, XAML y la relación con .NET (*Windows Presentation Foundation; Katrib, Del Valle, Sierra, Hernández; 2007*).

10.1. ¿Qué es WPF?

A pesar de la riqueza interactiva de las aplicaciones clientes, introducidas desde los orígenes del propio Windows, y que han llegado a su máxima expresión con la tecnología .NET y con los recursos de Windows.Form, lo cierto es que bajo las pretensiones integradoras y abarcadoras de las aplicaciones de hoy en día, estas capacidades tienen limitaciones. No es simple integrar en nuestras aplicaciones el despliegue de documentos, dar efectos tridimensionales, o colocar imágenes, audio y vídeo. Posiblemente algunas de las atractivas presentaciones que veamos por ahí, han tenido que desarrollarse bajo el embrollo de mezclar variadas tecnologías y formatos (controles ActiveX, Flash, PDF, etc.) exigiendo de varios desarrolladores, y requiriendo de ellos un alto nivel de complejidad para coordinar sus respectivos trabajos y resultados.

En ocasiones, nos podemos encontrar ante la necesidad de mejorar la apariencia de una aplicación desarrollada por otros. O nuestros clientes requieren la mejora de la interactividad de una aplicación desarrollada por nosotros. O incluso se nos plantea el inconveniente de tener que ser al mismo tiempo un buen diseñador artístico y un buen programador. Con PowerPoint uno es capaz de preparar y retocar las presentaciones logrando atractivos efectos y apariencia. Esto mismo se desearía hacer en muchas ocasiones con las propias aplicaciones y que además incorporasen

la funcionalidad y la lógica de nuestro negocio, pudiendo retocar ambas partes por separado.

Windows Presentation Foundation (WPF) es la propuesta de Microsoft para lograr todo esto.

WPF es toda una plataforma que da soporte para poder colocar en nuestras aplicaciones una amplia riqueza visual e interactiva, e integrarlas fácilmente con la lógica del negocio. Los elementos visuales de nuestras aplicaciones podrán tener ahora transparencia, brillo, tonalidades, sombra, reflejo, efectos tridimensionales y animaciones. Se podrá desplegar documentos y colocar en las propias aplicaciones audio y vídeo. Las aplicaciones podrán tener capacidad de enlace y navegación basado en el modelo de páginas de la Web. La reutilización será propiciada gracias a la posibilidad de definir estilos propios y plantillas, aumentando la productividad de desarrollo y dando una apariencia personalizada a las aplicaciones. Además la arquitectura abierta de WPF nos permite definir e implementar nuestros propios controles y componentes personalizados.

La potencialidad gráfica de WPF se basa a su vez en la tecnología DirectX de Microsoft que sabe aprovechar al máximo las posibilidades gráficas de que disponga el hardware donde se ejecute la aplicación.

Las capacidades de WPF son totalmente asequibles desde cualquier lenguaje .NET y utilizables programáticamente en el modo tradicional de la programación orientada a objetos. No obstante, Microsoft nos propone, junto a **WPF**, un lenguaje declarativo **XAML** que acopla a la perfección con WPF.

10.2. ¿Qué es XAML?

XAML (e**X**tensible **A**pplication **M**arkup **L**anguage) es un lenguaje declarativo propuesto por Microsoft que nos ofrece una vía productiva para definir las interfaces de usuario de las aplicaciones. XAML se basa en una sintaxis bien formada en XML y en su fácil extensibilidad.

Esencialmente, con XAML expresaremos declarativamente la creación de estructuras arbóreas de objetos .NET. Con WPF como marco de trabajo, estos objetos podrán ser desplegados visualmente y podrán interactuar con los usuarios permitiendo conformar una rica y excitante interfaz de aplicación. A la vez

podremos conectarlos con el resto de la lógica para asociar a esta apariencia de presentación una funcionalidad todo lo compleja que se requiera.

Aunque XAML y WPF son el matrimonio perfecto, pueden considerarse independientes. Como lenguaje declarativo que es, XAML puede servir para ser interpretado por otras tecnologías (si es que hay o habrá alguna tan basta como WPF). Por otra parte, WPF puede servir de infraestructura de ejecución a otros lenguajes declarativos de elementos de interfaz de usuario. Pero lo cierto es que ambos están diseñados de un modo muy bien interrelacionado. Puede decirse que XAML es excelente para "escribir la partitura" y WPF es la mejor "orquesta para interpretarla".

Por su naturaleza declarativa, XAML es apropiado para especificar la interfaz de usuario de una aplicación de modo separado de la lógica de negocio propia de la aplicación. En este sentido, la combinación de los tres ingredientes: lenguaje de programación .NET (**C#** en este proyecto), **XAML** y **WPF**, nos ofrece un adecuado soporte para desarrollar aplicaciones con la arquitectura conocida como MVC (*Modelo Vista Controlador*). Esta arquitectura por su separación en capas nos permite lograr mayor productividad de desarrollo y mayor flexibilidad del producto. Por ejemplo, si estuviéramos desarrollando una aplicación para jugar al ajedrez, con XAML y WPF se podría expresar cómo visualizar e interactuar con el tablero de ajedrez de forma más animada, atractiva e interesante (las piezas proyectando sombras sobre el tablero, un diseño futurista de las mismas, movimiento de las piezas por encima de otras…), y a su vez conectarlo con la estrategia de juego más sofisticada que podrá desarrollarse en C# (o el lenguaje de .NET preferido) aprovechando todas las bondades de ejecutar en un contexto .NET.

La integración de C#, XAML y WPF, puede llevarse a cabo de manera cómoda y estructurada con la herramienta Microsoft Visual Studio, por lo que ésta ha sido la elegida para desarrollar el proceso de reingeniería para la migración del software mostrado en el presente Proyecto Fin da Carrera.

10.3. XAML y .NET.

Estrictamente hablando, XAML es un lenguaje de marcado que sigue las reglas sintácticas de XML, donde cada elemento tiene un nombre y puede tener varios atributos. Los elementos se definen uno dentro de otro formando una estructura arbórea, lo que le da expresividad y semántica a este lenguaje en su correspondencia con un marco de trabajo como lo es WPF y .NET.

Por lo general, todo elemento en XAML se corresponde con una clase de .NET, en particular de WPF, y todo atributo XAML se corresponde con el nombre de una propiedad, o de un evento, de dicha clase. A la inversa, aunque no todas las clases del CLR se puedan representar en XAML, aquellas que tengan un constructor por defecto y propiedades públicas pueden ser instanciadas declarativamente en código escrito en XAML si existiesen los convertidores apropiados para convertir una cadena (que es en definitiva el único tipo de valor que se escribe en un lenguaje como XAML) en el tipo de .NET correspondiente (numérico, booleano, enumerativo, etc.).

10.4. El code-behind.

Si hacemos *click* sobre el botón de una aplicación, no ocurrirá nada a menos que se defina un comportamiento asociado a esta acción.

XAML da soporte al concepto de ***code-behind***. Todo archivo XAML tiene un/unos correspondiente/s archivo/s con código ejecutable (C# en nuestro caso). La idea es que el XAML nos da la apariencia y estructura de la interfaz de usuario, mientras que el archivo con code-behind nos da el comportamiento.

En el caso de nuestro proyecto, se intentó reducir al máximo el uso de code-behind, de modo que el usuario que tuviese el propósito de disfrutar del proyecto migrador, no tuviese que disponer de una librería demasiado extensa encargada de realizar las acciones, sino que el propio XAML generado tratara temas como la herencia y el control de eventos. Por otro lado, las acciones a ejecutar asociadas a los eventos definidos en ciertos controles de las presentaciones, deben definirse haciendo uso del code-behind para aportar al control la funcionalidad requerida.

10.5. Ventajas de la combinación XAML y WPF.

Si desarrollamos alguna aplicación con Windows Forms usando Visual Studio, se puede llegara a pensar que no es necesario tener que desarrollar a mano el código XAML y el code-behind asociado al mismo. Todo esto es lo mismo que podemos hacer con el diseñador de Visual Studio cuando implementamos una aplicación Windows Forms (o directamente una aplicación WPF) y arrastramos un botón desde la barra de herramientas para luego retocarle las propiedades con el editor de propiedades. A continuación se muestra una WPF Application creada con MVS 2008, para entender en qué consiste el código XAML del que se ha hablado, así como la visualización de la interfaz gráfica y el code-behind asociado.

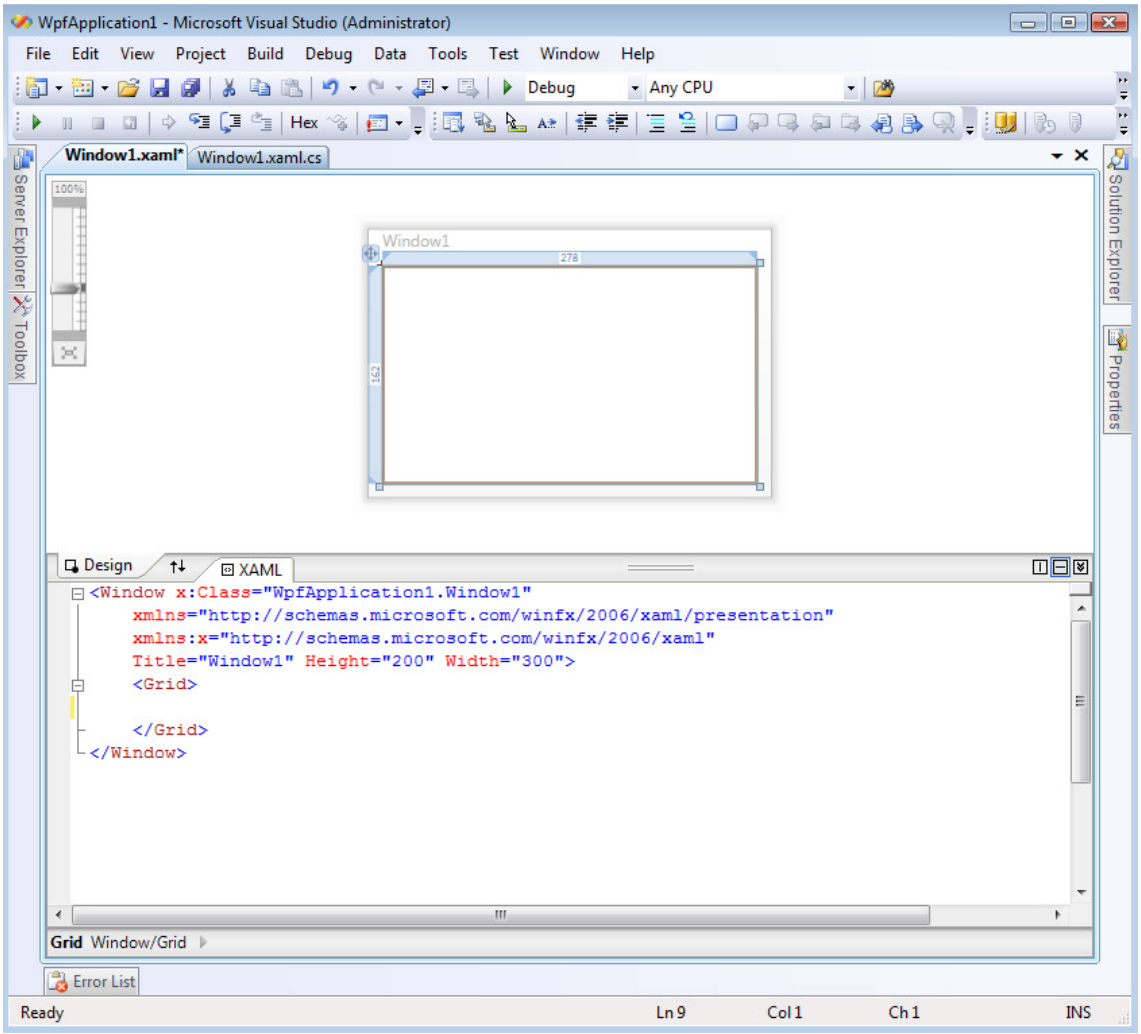


Ilustración 18 – XAML del WPF Application inicial

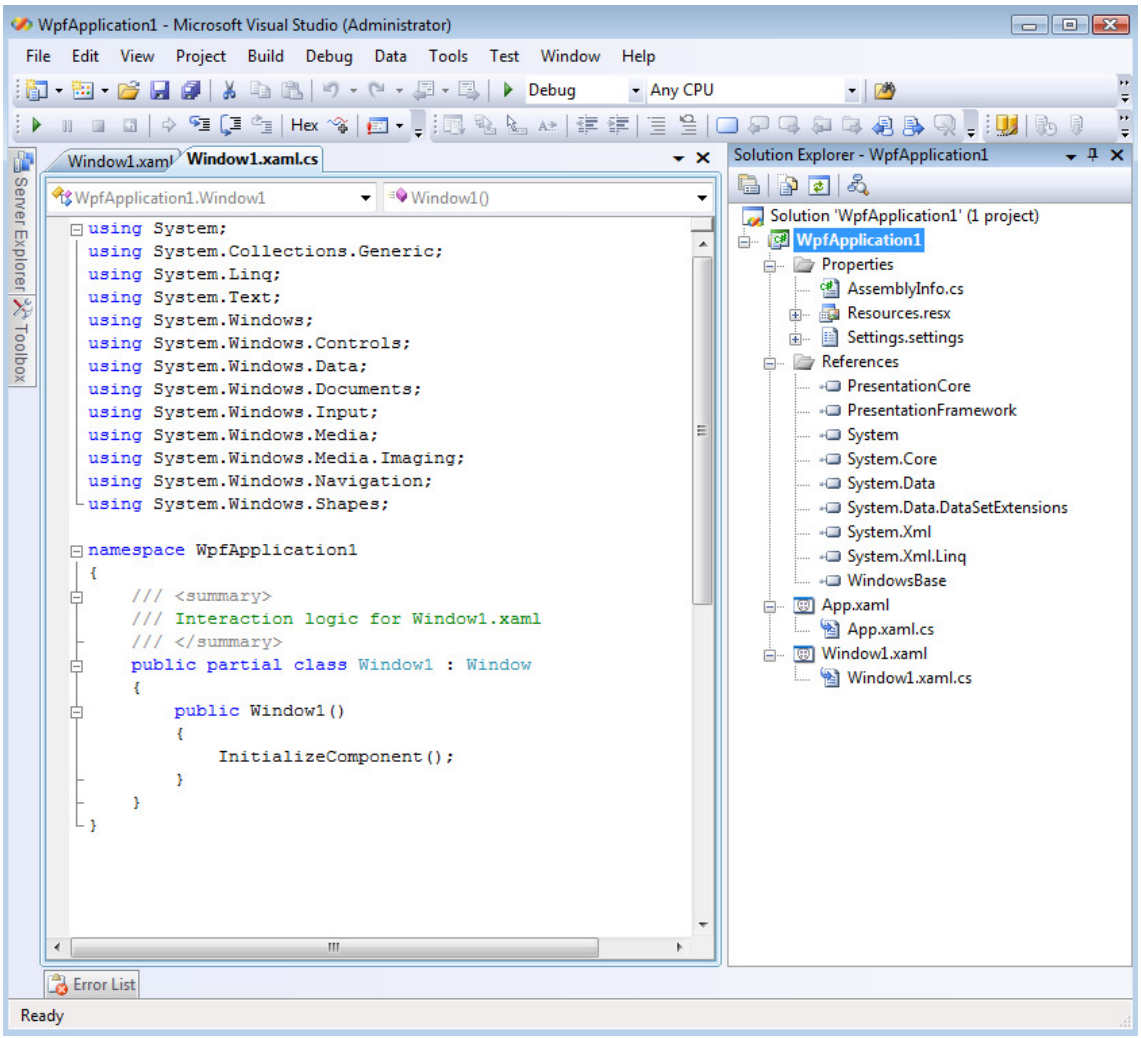


Ilustración 19 - C# del WPF Application inicial

En este momento se procede a arrastrar un botón desde la barra de herramientas, el cual podrá ser modificado retocándole las propiedades con el editor de propiedades. Algunas de estas propiedades son: diseño de los bordes, estilo y color de la letra, estilo y color del propio botón, tamaño, visibilidad, alineamiento.

Además, a dicho botón, en el propio código XAML se le ha añadido una propiedad Click con un evento asociado, como se indica en la siguiente imagen.

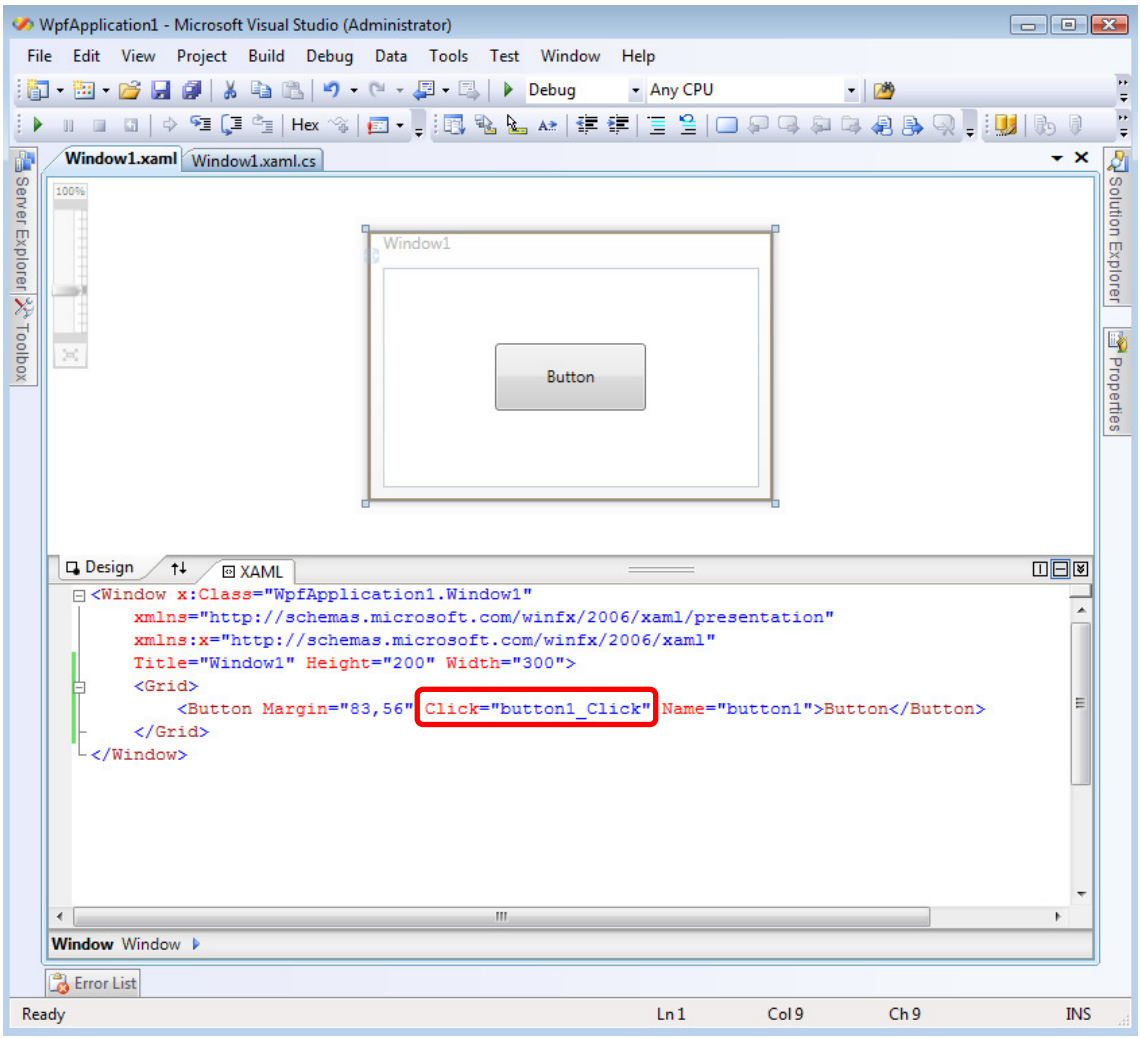


Ilustración 20 - XAML del WPF Application con un botón

Como puede observarse, el botón tiene una propiedad *Click*, la cual tiene un evento asociado de nombre *button1_Click*. Esto indica que cuando un usuario haga "click" sobre el botón *button1* (nombre del botón, indicado en la propiedad *Name*), se ejecutará la acción definida en el método *button1_Click*, definido en el code-behind como muestra la siguiente figura. Dicha acción a ejecutar consiste en la funcionalidad que se le quiera otorgar a dicho botón, por ejemplo, acceder a una página Web cuando el botón es pulsado.

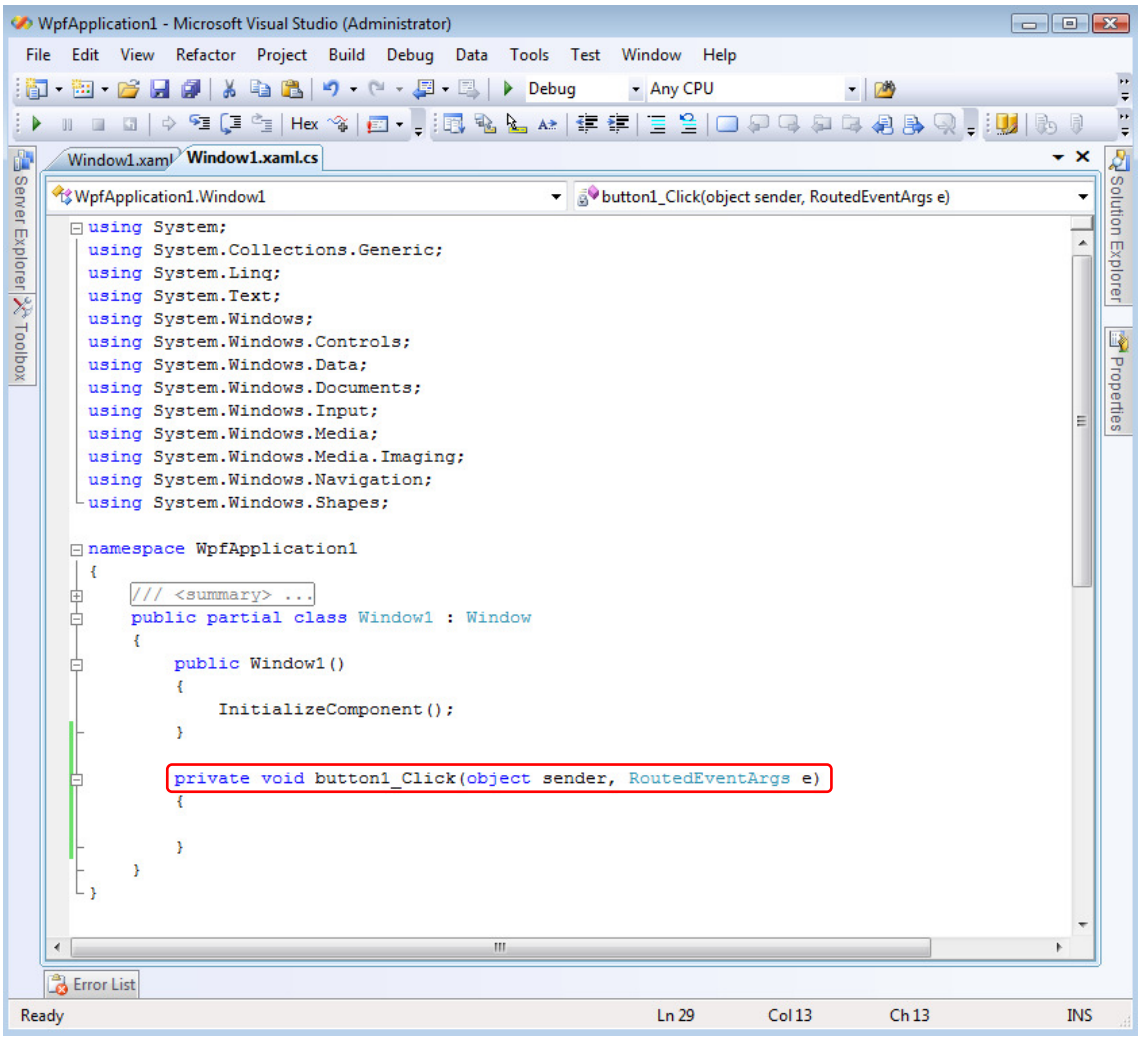


Ilustración 21 - C# del WPF Application con un botón

En este caso es el diseñador de Visual Studio quien está por detrás generando el código imperativo (C#) correspondiente. Como se puede observar, el nombre de la función creada es el mismo que se le especificó anteriormente en el código XAML.

Entonces se puede llegar a pensar que toda esta nueva parafernalia de WPF y XAML, ¿para qué? ¿No es más complicado escribir este código en XAML que trabajar visualmente con el diseñador de Visual Studio y el editor de propiedades?

Lo que hace más valioso a esta integración entre WPF y XAML es que al quedar la interfaz de usuario especificada con un lenguaje de marcas y declarativo, la hace legible y retocable por el usuario, y lo que es más importante, fácilmente susceptible de ser analizada y procesada por herramientas. Esto ofrece a terceros la

posibilidad de crear herramientas de diseño visual que soporten XAML. O lo que para este proyecto es más importante aún, la creación de controles personalizados que aporten una funcionalidad concreta a dichos controles, y la posibilidad de definir estilos propios para aportar a nuestra presentación la apariencia deseada.

La separación entre código XAML para expresar la interfaz y apariencia, y código de un lenguaje .NET como C# para expresar la lógica de la aplicación, facilita que los diseñadores con capacidades artísticas (posiblemente poco habilidosos para la programación) puedan integrar mejor y con más efectividad su trabajo con los programadores concentrados en la lógica de la aplicación (a veces poco talentosos en términos de diseñar una atractiva apariencia).

XAML es extensible, como su propio nombre lo indica, lo que significa que los desarrolladores podrán crear sus propios controles y elementos personalizados. Este hecho ha sido fundamental para la consecución de este proyecto, ya que gran parte de los objetos a migrar no están definidos en WPF, y muchos otros necesitan ser tratados de un modo especial para su acceso a bases de datos y control de eventos, por lo que se necesita poder definirlos como controles personalizados.

10.6. Detalles previos a la solución.

Como se ha comentado con anterioridad, **WPF** permite disponer de elegantes y atractivas presentaciones, a la par que éstas pueden incorporar la lógica de negocio y funcionalidad deseada, gracias al uso del code-behind. Las presentaciones que se pretenden migrar gracias a este proyecto de reingeniería del software, constan de una gran importancia en cuanto a la interfaz gráfica se refiere, pero más importante es aún si cabe la funcionalidad asociada a estas interfaces de usuario. Sin ir más lejos, la presentación objetivo a migrar en el presente Proyecto Fin de Carrera se basa en un mantenimiento de empleados, en la cual se muestran las características empresariales asociadas a dichos recursos humanos. Por lo tanto, la correcta visualización de dichas características toma un papel relevante en la presentación, pero más importante es aún poder aportar la funcionalidad para, por ejemplo, poder modificar los datos de los empleados en cuestión, acceder a aquellos recursos humanos deseados, crear nuevas entradas en la base de datos, permitir al usuario cambiar la visualización de la interfaz, etc.

Además, WPF ofrece la posibilidad de definir estilos propios y plantillas que otorguen a la presentación una apariencia personalizada, por lo que las interfaces

de usuario que migremos a tecnología .NET podrán tener un aspecto diferente al antiguo, más moderno, ofreciendo de este modo un producto aún más atractivo. Además la arquitectura abierta de WPF permite definir e implementar controles y componentes personalizados, aspecto básico para la consecución del proceso de reingeniería, ya que la empresa productora de software en la que se ha llevado a cabo el proyecto de migración, dispone de controles propios que no existen en WPF, por lo que es necesario su definición como control personalizado para su posible migración, como se comentará posteriormente.

Debido a que WPF acopla a la perfección con el lenguaje declarativo **XAML**, se decidió que éste sería el lenguaje idóneo para definir la interfaz gráfica de la presentación migrada. Esencialmente, con XAML expresaremos declarativamente la creación de estructuras arbóreas de objetos .NET. Gracias a la permisividad de WPF con respecto a la creación de controles personalizados, XAML permitirá la inclusión de dichos controles en el código, siempre y cuando, al crearlos en C#, estos controles definan un control WPF existente. Es decir, si queremos crear un botón personalizado, dicho botón deberá heredar de la clase *Button*, control existente en WPF; de ese modo, el código XAML será reconocido como una estructura bien formada y la presentación podrá ser visualizada correctamente.

A continuación, antes de comenzar con la explicación del proceso de migración del software, se mostrará un ejemplo en el que se recoge lo mencionado en el anterior párrafo. Se trata del código XAML obtenido tras la migración de una presentación en OBL (código perteneciente a la empresa donde se ha desarrollado el proyecto) que consta de un botón, más un contenedor que a su vez dispone de dos botones, más una caja de texto. Como se puede observar, los contenedores *Window*, *Grid* y *GroupBox* son controles WPF normales, que contienen las mismas características que tuvieran los controles correspondientes en el código OBL antes de ser migrado. Por otro lado, podemos comprobar que los botones y la caja de texto son controles personalizados, debido a que estos contaban con propiedades especiales con respecto a su funcionalidad y control. Como se ha comentado anteriormente, para que el código XAML fuese un código bien formado, estos controles personalizados deben definir un control existente en WPF. Esto es posible haciendo que el control propio que creemos herede del control WPF que vaya a representar. En el caso del presente Proyecto Fin de Carrera, se ha definido un paquete (denominado "proyecto" en el ámbito de la solución del proyecto de migración por su definición en MVS como *Project*) en el cual se definen todos los controles personalizados que se necesiten para la migración.

Estas son las dos clases diseñadas para la definición de los controles personalizados mencionados (botón y caja de texto), y el código XAML producto de la migración de la presentación comentada previamente:

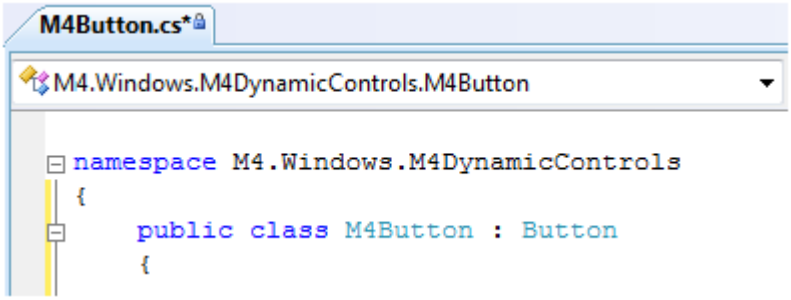


Ilustración 22 - Definición del control personalizado M4Button

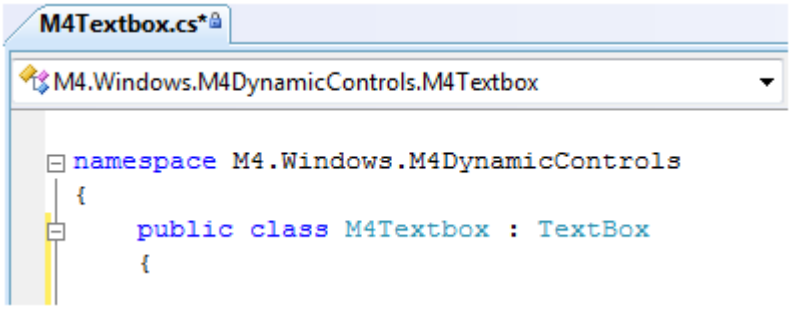


Ilustración 23 - Definición del control personalizado M4Textbox

En el primer caso tenemos la definición de un botón personalizado, y en el segundo caso la definición de una caja de texto personalizada. El primero de los controles define un control *Button*, mientras que el segundo define un control *Textbox*. Ambos pertenecen al paquete *M4DynamicControls*, por lo que la clase necesita saber que su namespace es *M4.Windows.M4DynamicControls*, en el cual se definen todos los controles personalizados necesarios para la migración.

En el siguiente código XAML, podemos comprobar en la cabecera del control *Window* como se define ***xmlns:local***, variable a la cual se le da el valor del namespace *M4.Windows.M4DynamicControls*, para que el código XAML reconozca que los controles personalizados (definidos como *local:M4Button* y *local:M4Textbox*) se encuentran definidos en dicho namespace.

```
<Window xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-
        namespace:M4.Windows.M4DynamicControls;assembly=M4Control" Cursor=""
        WindowStartupLocation="CenterScreen" Title="Form1" FontFamily="Ms
        Sans Serif" FontSize="8pt" Height="406" Width="445" Name="Form1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">

    <Grid Background="{x:Static SystemColors.ControlBrush}" Name="GridMain">

        <local:M4Button FontFamily="Ms Sans Serif" FontSize="8pt" Height="50"
                    VerticalAlignment="Top" HorizontalAlignment="Left"
                    Margin="30,31,0,0" Content="Migrate button" Width="136"/>

        <GroupBox BorderBrush="Gray" BorderThickness="2" Padding="3"
              Grid.Row="0" VerticalAlignment="Top" HorizontalAlignment="Left"
              Margin="121,186,0,0" Width="260" Height="160" Name="Groupbox1">

            <Grid>

                <local:M4Button Height="31" VerticalAlignment="Top"
                                HorizontalAlignment="Left" Margin="20,47,0,0"
                                Content="Button2" Width="78" />

                <local:M4Button Height="28" VerticalAlignment="Top"
                                HorizontalAlignment="Left" Margin="137,50,0,0"
                                Content="Button3" Width="51" />

            </Grid>

        </GroupBox>

        <local:M4Textbox Text="My Textbox" Height="27"
                         VerticalAlignment="Top" HorizontalAlignment="Left"
                         Margin="169,99,0,0" Width="215" Name="Textbox1" />

    </Grid>
</Window>
```

Como se puede demostrar tras lo comentado hasta este punto, XAML, C# y WPF se pueden combinar de manera idónea para llegar a la consecución del proyecto de migración deseado.

Lo que resta de este capítulo, está dedicado a: mostrar la situación de partida del proyecto de migración, los pasos dados para llegar al producto final con las ventajas del entorno .NET, y la solución final obtenida con una serie de presentaciones para comprobar el resultado final de la migración.

10.7. Aspectos generales de la solución.

Como se ha comentado a lo largo del presente Proyecto Fin de Carrera, nos encontramos en el tercer ciclo del proyecto de reingeniería para la migración de un software *legacy* (es decir, obsoleto: poco usado, anticuado, inadecuado a las circunstancias actuales) (*Real Academia de la Lengua Española*), a tecnología .NET. Tal como se mencionó en el apartado 6.2, la primera etapa, o ciclo, consistió en una fase completamente teórica acerca de *qué* se quería hacer y *cómo* habría que hacerlo. La segunda etapa consistió en el desarrollo de un marco de reingeniería y en la ejecución del propio proceso de reingeniería, con una presentación sencilla para comprobar que la migración era correcta. La presente y tercera etapa representa el tercer ciclo del proceso de reingeniería, con un nuevo enfoque desde el punto de vista técnico, y sin olvidarnos de la adopción del marco desarrollado en la segunda fase, el cual ha podido ser utilizado y adaptado en el presente ciclo.

El nuevo enfoque del que se hablado consistió en modificar la arquitectura del modelo de migración. En esta etapa, cualquier presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un ejecutable (.EXE) ni efectuar almacenamientos en el repositorio de la compañía, permitiendo, de esta manera, que las modificaciones en las presentaciones sean traducidas en el momento de ser necesitadas, eliminando los problemas de consistencia en el almacenamiento persistente entre las presentaciones. De este modo, la necesidad de acumular ficheros XAML fue eliminada, así como la creación de un fichero que necesitase de su ejecución para la visualización de la presentación migrada.

La siguiente figura muestra la arquitectura de la solución del ciclo anterior. En ella se puede comprobar cómo el proceso de migración es el siguiente, con cuatro pasos diferenciados:

- En primer lugar, se realiza una llamada a un objeto COM que contiene toda la funcionalidad. Sin entrar en detalle, se parte del código OBL generado tras la grabación de la presentación desarrollada en la herramienta a migrar, *Editor de Presentaciones*. Entonces comienza el proceso propio de migración. Dicho proceso genera un conjunto de clases, *Window1.xaml* y *Window1.xaml.cs*, tanto para la parte de código XAML como para la parte de *code-behind* correspondiente a dicho XAML.
- Posteriormente comienza un nuevo proceso donde se generan las clases restantes necesarias para elaborar un proyecto WPF, que representará en el

entorno .NET la presentación migrada. Por ello, se generan las clases *App.xaml, App.xaml.cs* y *ClsMain.cs*.

- Por último, y una vez se han generado todas las clases necesarias para construir un proyecto WPF que representará la presentación migrada, se comenzará con el proceso final. En este último paso, se deberá elaborar un fichero XML con una estructura especial, necesario para construir un proyecto WPF. Dicho fichero será el argumento de entrada al proceso *msbuild.exe*, encargado de generar el proyecto.
- Una vez realizado todo el proceso, se generarán una serie de carpetas *\obj\Debug* y *\bin\Debug* con todos los ficheros relativos al proyecto. Todo el contenido de la subcarpeta *\bin\Debug* se deberá comprimir para su posterior almacenamiento en el repositorio de la compañía, en la tabla correspondiente creada para el almacenamiento de presentaciones .NET.

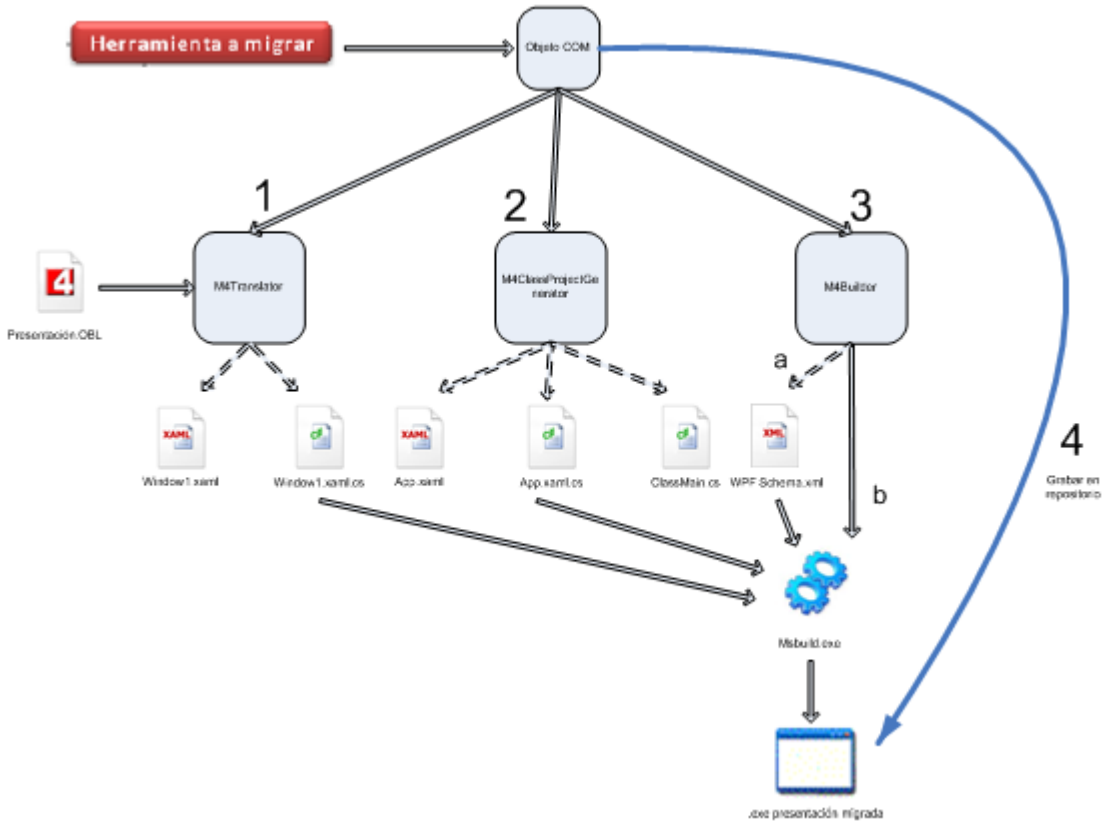


Ilustración 24 - Descripción de la operativa de la solución desechada

Atendiendo al nuevo enfoque seguido, esta arquitectura no tiene ningún sentido excepto en el paso 1, en el cual se realiza la migración del código OBL a la tecnología .NET (XAML y C#). En la siguiente figura se puede comprobar la nueva operativa de la solución. Se distinguen cuatro pasos específicos, los cuales serán explicados posteriormente con más detalle:

- Obtención del XML de Ejecución a partir del OBL.
- Obtención del XAML de Visualización tras la migración.
- Obtención del XAML Operativo tras la gestión de eventos.
- Invocación de la presentación migrada.

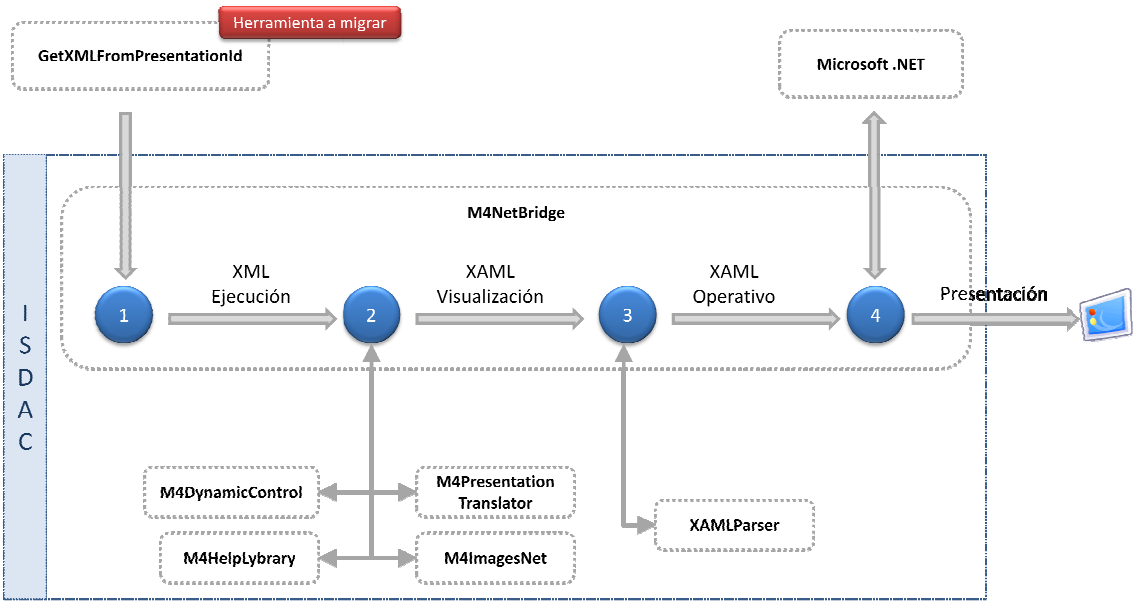


Ilustración 25 - Descripción de la operativa de la solución actual

Obtención del XML de Ejecución

Como primer paso interno a la solución se debe conseguir un XML de Ejecución a partir del OBL de la presentación. El primer paso que se debe dar será validarse en la plataforma propiedad de la compañía que dispone de la herramienta que permite crear y acceder a las presentaciones, con el propósito de poder acceder a la presentación a migrar. Para ello, será necesario en primer lugar realizar el registro de la máquina virtual que nos mantendrá logueados en la plataforma. En segundo lugar, se llama al método *GelXMLFromPresentationId*, contenido en una librería propia de la compañía, al que se le pasa como parámetro el identificador de la presentación que se quiere migrar, y la variable, de tipo *String*, donde se

almacenará el XML de Ejecución que se pretende obtener. Este método, externo a la solución, obtiene a partir de la presentación el código OBL de la misma, y lo traduce al XML de Ejecución. El XML de ejecución, de forma simplificada, se puede considerar como un XML formateado para los propósitos de la migración.

Obtención del XAML de Visualización

El segundo paso consiste en construir el XAML de Visualización a partir del XML de Ejecución. Para ello, a partir de la variable que contiene el XML comentada anteriormente, se parsea su contenido con el método *translateToNetPresentation* desarrollado dentro de la solución. Este método realiza los siguientes pasos para construir el XAML de Visualización, o con otras palabras, para realizar el proceso de migración de código:

- Comprueba la correspondencia de cada etiqueta del XML de Ejecución con las clases de migración desarrolladas a tal efecto.
- Ejecuta el método constructor de cada clase de migración que inicializa las variables propias de la correspondiente etiqueta analizada. Esta inicialización corresponderá con el valor por defecto que dicha propiedad tome en la plataforma mencionada anteriormente.
- Ejecuta el método *Migrate* propio de cada clase de migración, que traduce a XAML dicha etiqueta y las propiedades de la misma, en función de la etiqueta que se pretenda migrar.
- Realiza una llamada a la migración de sus objetos hijos y se repite el proceso de migración, ya que la estructura del XML de Ejecución, así como la del XAML, se trata de una estructura arbórea.
- Finaliza la ejecución cerrando las etiquetas para que el XAML obtenido sea un código bien formado.

Obtención del XAML Operativo.

Teniendo en cuenta que se pretende minimizar el uso de code-behind, en este paso se realiza la inclusión de capacidades XAML que permitan la ejecución de los eventos. Esta funcionalidad se consigue mediante el proyecto *XAMLParser* incluido en la presentación, y más concretamente gracias al método *Parse*. Dicho método realiza los pasos correspondientes a partir del XAML de Visualización para conseguir el XAML Operativo. Básicamente, en primer lugar parsea los objetos que tienen

asociados eventos. Y posteriormente, para cada objeto que pretende invocar un evento, dicho método incluye un específico código XAML en el código XAML de Visualización obtenido en el paso anterior, con el fin de capacitar la ejecución de los eventos, minimizando la invocación de code-behind. De este modo, se obtiene por fin el XAML Operativo, el cual ofrecerá, aparte de la visualización de la presentación migrada, la posibilidad de interactuar con ella gracias a la funcionalidad implementada.

Invocación de la presentación.

Una vez se ha finalizado todo el proceso de compilación y generación del proyecto XAML Operativo, que traslada la presentación obsoleta al entorno .NET, se invocan las librerías de .NET que permiten mostrar la presentación a partir del XAML Operativo, con el fin de visualizar la presentación migrada y poder interactuar con ella del mismo modo que se hacía con la presentación diseñada en el *Editor de presentaciones*.

10.8. La solución: migración de la tecnología.

Hacer posible la migración de la tecnología obsoleta al entorno .NET, ha supuesto una gran cantidad de horas dedicadas a la investigación y el desarrollo de la solución. Esta solución (denominada así ya que MVS lo denomina *Solution*) consiste en una serie de proyectos (denominados así ya que MVS los denomina *Project*) que en conjunto hacen posible que la migración sea correcta.

A continuación se muestra una imagen que recoge los componentes (proyectos) necesarios para la solución de la reingeniería del software mantenida en este proyecto. Posteriormente, se describirán dichos componentes del sistema de migración con detalle para entender cómo se lleva a cabo la migración. Se quiere poner de manifiesto que los componentes identificados no corresponden a componentes software en su visión más académica, sino a elementos funcionales dentro de la solución desarrollada. En el ciclo 03, presentado en el presente Proyecto Fin de Carrera, dichos componentes son completamente imprescindibles. Pero cuando el proyecto se encuentre inmerso en el siguiente ciclo, se estudiará la simplificación de estos, si fuese posible, o por el contrario, la posible inclusión de otros componentes para aportar nueva funcionalidad.

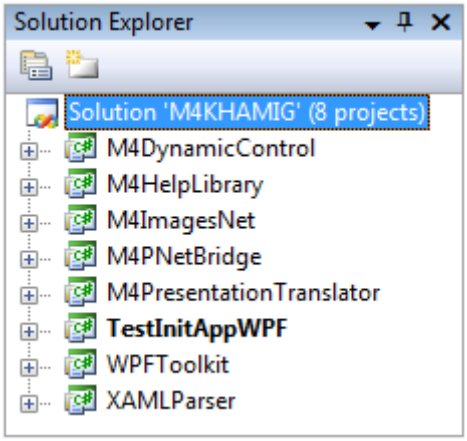


Ilustración 26 - Solución del proyecto de migración

M4PNetBridge

Este componente será el encargado de realizar la conexión con la plataforma que dispone de la presentación a migrar, para posteriormente realizar los pasos necesarios de la migración. Se recuerda que estos pasos son:

- Obtención del XML de Ejecución a partir del OBL.
- Obtención del XAML de Visualización tras la migración.
- Obtención del XAML Operativo tras la gestión de eventos.
- Invocación de la presentación migrada.

Además, el proyecto M4PNetBridge se encomienda de cargar en memoria aquellas referencias necesarias para la solución, así como los recursos gráficos y textuales que puedan ser necesarios para las presentaciones a migrar.

Cuando la solución sea ejecutada, se ejecutará a su vez la plataforma que permite diseñar presentaciones y, por supuesto, acceder a las presentaciones ya diseñadas. En dicha plataforma, existirá un apartado dedicado a la migración de las presentaciones. Indicando el identificador de la presentación a migrar, podremos obtener la solución de la interfaz de usuario en tecnología .NET.

TestInitAppWpf

Este componente no pertenece a solución final. Ha sido desarrollado al comienzo del proyecto de reingeniería ya que con él se obtiene una comodidad y eficiencia mucho mayor a la hora de realizar las pruebas.

El proyecto TestInitAppWpf consiste simplemente en una interfaz gráfica con dos botones, tal como se muestra a continuación:

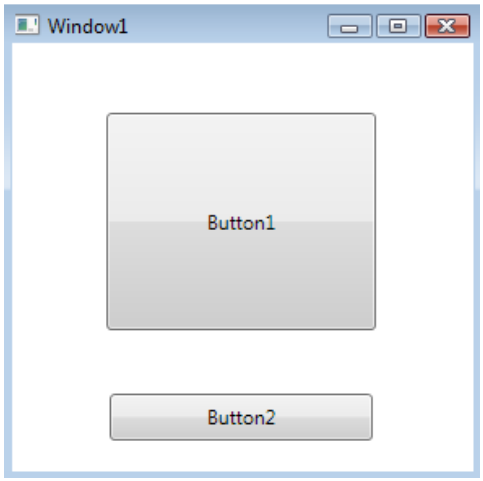


Ilustración 27 - Interfaz gráfica del componente TestInitAppWpf

La funcionalidad del botón "*Button1*" consiste en la conexión a la plataforma de la compañía mediante un usuario registrado en ella, gracias a la instancia de una máquina virtual creada con dicho fin. A partir de ésta, podremos acceder a una presentación previamente diseñada y almacenada en el repositorio de la compañía, y realizar la llamada al componente M4PNetBridge, explicado en el apartado anterior, con el fin de realizar la migración del software. De este modo, no será necesaria la ejecución de la plataforma para realizar la migración, lo cual supone un ahorro en tiempo considerable. Simplemente será necesario indicar en el presente componente el identificador de la presentación que se desea migrar.

Por otro lado, la funcionalidad del botón "*Button2*" consiste en acceder a un código escrito en XAML indicando un path en el que se encuentra almacenado un fichero con extensión .xml que contiene dicho código XAML. Este fichero consistirá en un XAML de Visualización, el cual se transformará en un XAML Operativo y se invocará su visualización. De este modo, facilitaremos la gestión de pruebas, ya que se ofrece la posibilidad de modificar un código XAML que puede contener incongruencias debido a que la migración no era correcta. El desarrollador puede

modificar dicho XAML y de este modo comprobará qué errores se habían cometido en el desarrollo del proceso de migración con mayor facilidad en cuanto a la otra solución, consistente en modificar el código generador del XAML equivalente a la presentación a migrar.

M4HelpLibrary

Este componente es el encargado de aportar la funcionalidad de ayuda necesaria, equivalente a la ayuda ofrecida por cualquier software, invocada habitualmente mediante el control de ayuda F1.

Se invocará desde el componente M4PNetBridge, previamente a la ejecución de los pasos que hacen posible la migración de las presentaciones.

M4ImagesNet

Este componente es el encargado de aportar a la solución las imágenes que cuentan con la posibilidad de ser necesitadas en la presentación que se esté migrando. Dicho componente estará formado por una serie de directorios, cada uno de ellos con diferentes imágenes utilizadas en presentaciones de diferente índole. En el caso que nos compete, se ha utilizado un directorio particular en el que se almacenan aquellos ficheros de imagen que tienen cabida en las presentaciones que se pretendían migrar en el presente proyecto de reingeniería.

Si se desea visualizar una nueva imagen que esté contenida en la presentación pero que no se encuentre almacenada en el directorio comentado anteriormente, será necesario añadir el fichero de imagen deseado al directorio. Para el correcto funcionamiento, el proceso de inclusión de una imagen es el siguiente:

- Se coloca el fichero que representa la nueva imagen en el directorio especificado.
- Desde el entorno MVS, sobre el directorio de imágenes anterior, elegir la opción "Add Existing Item". En este caso se podrá elegir la imagen añadida anteriormente a dicho directorio y se seleccionará la opción "Add".
- Se arrastra la nueva imagen al fichero *M4ActiveDashboard.resx*, contenido en el proyecto M4ImagesNet.
- Se compila y construye el proyecto M4ImagesNet, con el fin de que la imagen quede registrada.

De este modo, además de las imágenes que se dispone en la plataforma propiedad de la compañía, se podrán incluir nuevas imágenes con un aspecto más moderno, ofreciendo una interfaz de usuario más elegante.

WPFToolkit

Este componente es el encargado de contener las clases e iconos/imágenes necesarios para que se pueda mostrar la función de calendario en aquellas presentaciones que lo contengan.

En principio, esta funcionalidad ha sido agregada a la solución como un componente más, pero se estudiará la posibilidad en el siguiente ciclo de prescindir de él, acoplando las clases e iconos/imágenes necesarios en los proyectos que necesariamente deben estar presentes para el correcto funcionamiento de la migración del software.

M4PresentationTranslator

Este componente es el encargado de realizar la traducción del código OBL de una presentación ya diseñada, o mejor dicho, la traducción del código XML de ejecución correspondiente al código OBL de una presentación ya diseñada, a su equivalente en WPF, ofreciendo al usuario la ventajas del entorno .NET.

Uno de los pasos más atractivos de este proyecto reside en la migración del software. Básicamente, el proceso de migración consiste en leer paso a paso las etiquetas del código XML de ejecución, y traducirlas a código XAML. Cuando una etiqueta del código XML de ejecución de la presentación a migrar es leída, a dicha etiqueta se le añade el prefijo "M4", ya que todas las clases de este proyecto que se encargan de la migración de los controles, están bautizadas como: "M4" + <nombre_control>. De ese modo, se crea una instancia de la clase en cuestión que represente el control, y en cada una de estas clases se procede a controlar todas las propiedades posibles a disponer para dicho control, así como su método propio de migración.

A continuación se muestra un ejemplo sencillo de una de estas clases correspondientes a un control con correspondencia directa en XAML, por ejemplo, la clase del control *Treeview*:

```
...
using ...;
...

namespace M4.Windows.M4PresentationTranslator
{
	/// <remarks>
	/// Class that represents a M4Treeview in a OBL Presentation.
	/// </remarks>
	public class M4Treeview : M4Control
	{
		// Atributos
		// Constructor
		// Métodos get y set
		// Método Migrate
	}
}
```

Como se puede comprobar, el primer paso radica en incluir los using necesarios para la compilación de la clase. Además, es absolutamente necesario indicar en el namespace el proyecto en el que se incluye la clase en cuestión.

Cada control dispone de ciertas propiedades, las cuales han sido analizadas gracias a un archivo disponible en la compañía donde ha sido llevado a cabo el proceso de reingeniería del software. Dicho archivo consiste en un fichero de texto plano en el cual se indican todos los controles que pueden contener las presentaciones, así como las propiedades de dichos controles. De este modo, los atributos de la clase que representa cada control, se corresponderán con las propiedades mencionadas, con el fin de ser utilizados en el proceso de migración, para que la presentación migrada se corresponda con la presentación original.

El método *Migrate* consiste en una función que se encarga de la migración de los controles. Todos ellos heredan de una clase conocida como *M4Control*, en la cual está definido dicho método *Migrate*, y cada control sobrescribirá dicho método para aportar la funcionalidad requerida en función del control. Además, dicha clase *M4Control*, debido a ser una clase genérica que representa un control WPF, contiene las propiedades comunes a todos los controles, con el fin de que estos no tengan que incluirlas en cada clase, sino que las hereden directamente del padre.

Volviendo al método *Migrate*, éste sería como sigue. Se incluirán ciertos cuadros de texto numerados para posteriormente hacer mención a ellos y explicarlos:

```
public override void Migrate(M4Control oParentControl, XmlNode
                             oM4ParentNode, M4ContextData oM4Context)
{                                                                    1
    XmlNode oM4ChildNode;
    MethodInfo myMethod;
    XmlDocument oM4XAMLWriteDocument = oM4ParentNode.OwnerDocument;

    /* El control TreeView muestra una jerarquía de nodos
     * similar al modo en que se muestran los archivos y las
     * carpetas en el panel izquierdo de la característica
     * Explorador de Windows de sistemas operativos Windows.
     */
    oM4ChildNode = oM4XAMLWriteDocument.CreateElement("", "TreeView",
        "http://schemas.microsoft.com/winfx/2006/xaml/presentation");
                                                                     2
    // añadir el nodo creado al nodo padre
    oM4ParentNode.AppendChild(oM4ChildNode);

    // se añaden todos los atributos con sus respectivos valores
    PropertyInfo[] oM4PropertiesInfo=this.GetType().GetProperties();
    foreach (PropertyInfo oM4PropertyInfo in oM4PropertiesInfo)
    {
        if (oM4PropertyInfo.Name != "M4Controls")
        {
            /* Recuperar el método que migra dicha propiedad
             * (específico, o general en ausencia de este)
             */
3 ->        myMethod = (new M4MigrateMethods()).GetType().GetMethod(
                        "Migrate" + oM4PropertyInfo.Name,
                        new Type[] { typeof(M4Control),
                        typeof(M4Treeview), typeof(XmlNode),
                        typeof(PropertyInfo), typeof(M4ContextData)});

            // Invocar al método recuperado
            try
            {
4 ->            myMethod.Invoke(this, new object[] { oParentControl,
                    this, oM4ChildNode, oM4PropertyInfo, oM4Context });
            }
            catch (NullReferenceException)
            {
                string texto = "Error: la propiedad " +
                      oM4PropertyInfo.Name + " no pudo ser migrada ";
                M4Utils.M4AddToLog(texto);
            }
        }
    }

    // Invocar a la migración de cada uno de los controles contenidos
    foreach (M4Control oM4Control in this.M4Controls)
    {
        oM4Control.Migrate(this, oM4ChildNode, oM4Context);
    }
}                   5
```

Índice	Descripción
1	Todos los controles que se pretenden migrar, tendrán indicado el contexto en el que se encuentra dentro del código, ya que la migración puede variar en función de dicho contexto, por ejemplo, si el control estuviese contenido en un objeto *Table*.
2	Se dispone de un fichero con extensión .xml, que representará el XAML de la presentación migrada. Una vez leída una etiqueta del código XML de ejecución, se procederá a almacenar en el fichero mencionado anteriormente la correspondencia de este control con su homólogo en XAML.
3	Todos los atributos creados previamente deberán migrarse como propiedades del control en cuestión. Para conocer el método de migración al que se hará referencia en el siguiente paso, se añadirá el sufijo "Migrate" al nombre de la propiedad a migrar.
4	Las propiedades con las que cuenta el control que se esté migrando, deberán ser migradas para que la presentación en WPF se corresponda con la original. Para migrar dichas propiedades, se hará uso de una clase (*M4MigrateMethods*) que contiene los métodos de migración de las propiedades de los controles, escribiendo en el fichero XAML mencionado previamente la migración de dichas propiedades.
5	Para continuar el proceso, será necesario repetir estos pasos pero con los hijos contenidos en cada control tratado.

Tabla 4 - Explicación del método Migrate

Este componente, por lo tanto, será el encargado de la migración del software, en cuanto a la **visualización** se refiere. Para aportar funcionalidad a la presentación, será necesario utilizar el proyecto M4DynamicControl, el cual define los controles personalizados para dotar de las características correspondientes a los controles migrados a .NET.

M4DynamicControl

Este componente es el encargado de dos tareas principales:

- Creación de los controles personalizados de determinados objetos WPF.
- Aportación de la funcionalidad requerida a cada acción de cada evento.

El motivo de haber personalizado ciertos controles es debido a que se tratan de controles con propiedades especiales asociadas, como la gestión de eventos, el acceso a base de datos, cálculos matemáticos en tiempo de ejecución.

Existen ciertas clases especiales para la gestión de eventos. Éstas son:

- IM4Action.cs
- IM4EventDispatcher.cs
- M4EventHandler.cs
- M4EventParams.cs
- M4EventParamsList.cs

Gracias a ellas se podrá generar en XAML una disposición arbórea de tal manera que se permita ofrecer la posibilidad de que para un mismo control, se disparen varios eventos. Para ello se utiliza el control *M4EventHandler*, que representa cualquier evento que pueda tener un control, y una lista de estos eventos representada por el control *M4EventParamsList*. A su vez, para cada evento es posible disponer de una lista de acciones, representada por el control *M4EventParams*. Para ejecutar dichas acciones relativas a un evento, serán utilizados los controles propios de acciones como son *M4Action_dataprops* o *M4Action_value*, que ejecutarán una acción diferente en función al valor de la propiedad *Action*, *Dialog*, etc.

Se puede observar que existen dos interfaces. La primera de ellas es utilizada para representar las acciones a ejecutar asociadas a los eventos. La segunda, para representar cualquier control capaz de disparar eventos.

En cuanto a los controles personalizados, un claro ejemplo sería el control *Button*, ya que éste puede aparecer habilitado o deshabilitado en función del escenario concreto de la presentación. Además, los botones de una presentación existen para ejecutar alguna acción al ser pulsados, por lo que tendrán eventos asociados que ejecuten ciertas acciones. En este sentido, se dispone de clases que gestionan las acciones, como la del control *M4Action_dataprops*, que según la acción que el botón deba realizar al ser pulsado, el código desarrollado ejecutará una función

correspondiente, como puede ser el acceso a una base de datos para obtener la información de un registro, realizar acciones de edición sobre una presentación, imprimir la presentación, o cualquier funcionalidad que el desarrollador haya otorgado a la presentación que se quiere migrar.

XAMLParser

Este componente es el encargado de hacer posible la inclusión de capacidades XAML que permitan la ejecución de los eventos. Para conseguir ejecutar dichos eventos será necesario parsear los objetos contenidos en el XAML de Visualización que tienen asociados eventos, y para cada objeto que invoca la ejecución de un evento, incluir código XAML en el código que ya se disponía con el fin de capacitar la ejecución de los mismos, obteniendo de ese modo el XAML Operativo.

Dicho de otro modo, este componente dispone de clases y métodos suficientes que, entre otras funciones, tienen la capacidad de leer un código XAML, y para cada etiqueta de dicho código comprueba si contiene propiedades que representen eventos. Estas propiedades, por ejemplo, son: *MouseLeftButtonUp*, *Click*, *SelectionChanged*, *MouseLeftButtonDown*, etc. Entonces, cuando el presente componente se encuentre con una de estas propiedades en alguna etiqueta del código XAML, entenderá que tiene un evento asociado y procederá a la inclusión del código que gestiona dichos eventos.

Una vez se disponga del denominado XAML Operativo, será posible visualizar la presentación a la vez que ésta dispone de la funcionalidad requerida. Cada control que tenga un evento asociado, sabrá a qué evento debe responder, y qué acción deberá ejecutar. Es decir, sabrá navegar por el code-behind asociado a dicho evento, y ejecutará la acción pertinente requerida por el control.

A continuación se muestra un ejemplo en el que se explica paso a paso con más detalle la inclusión de capacidades XAML anteriormente mencionadas, que hacen posible disponer del código XAML Operativo que responde a la presentación migrada, incluida la funcionalidad.

1. Cuando se obtiene el código XML de ejecución correspondiente al OBL de la presentación, se le expone a un proceso de migración con el cual se obtiene el código XAML de Visualización. A continuación se mostrará un ejemplo en el que se dispone de un botón con un evento asociado. El código mostrado consiste en un pedazo de código XML de ejecución, el cual tiene una correspondencia directa con el código OBL recibido de la presentación. Se trata de un botón que tiene asociado un evento *Evclick*, el cual tiene asociada una acción *Action_dataprops*.

```
<Button ALIAS ="Button1">
      <Grants>31</Grants>
      <Left>151</Left>
      <Top>181</Top>
      <Width>199</Width>
      <Height>41</Height>
      <Text>Llamada al canal</Text>
      <Evclick ALIAS ="Evclick1">
            <Grants>31</Grants>
            <Action_dataprops ALIAS ="Action_dataprops1">
                  <Grants>23</Grants>
                  <Iditem>BOTONHOLA</Iditem>
            </Action_dataprops>
      </Evclick>
</Button>
```

2. A continuación se procede a mostrar el código XAML de Visualización resultante tras el proyecto de migración. En él se puede comprobar cómo la migración del control *Button* se corresponde con un control personalizado *M4Button*. Además, se puede comprobar como todas las propiedades de dicho botón también han sido migradas.

   El evento *Evclick* ha sido migrado según la estructura arbórea explicada en el apartado anterior reservado al componente M4DynamicControl; en este caso, como puede comprobarse en el código XML de ejecución, con un solo evento para el control *Button*, y una sola acción asociada a dicho evento.

Antes de pasar al siguiente punto, en el que se procede a incluir código XAML con el fin de capacitar la ejecución de los eventos, este código generado en XAML desconoce que la propiedad *Click* pertenezca al control *M4Button*. Pero posteriormente se explicará el porqué de esta propiedad.

```
<local:M4Button FontFamily="Ms Sans Serif" FontSize="8pt" Height="41"
                help:HelpProvider.HelpString="" VerticalAlignment="Top"
                HorizontalAlignment="Left" Margin="151,181,0,0" Cursor=""
                Content="Llamada al canal" Width="199"
                EnabledWhenRegisterIsDeleted="False"
                EnabledWhenRegisterIsNew="True"
                EnabledWhenRegisterIsNormal="True"
                EnabledWhenRegisterIsModified="True" Click="EventClick">
    <local:M4EventHandler.EventParamsList>
        <local:M4EventParamsList>
            <local:M4EventParams EventId="Click">
                <local:M4Action_dataprops Name="BOTONHOLA" Action="Execute" />
            </local:M4EventParams>
        </local:M4EventParamsList>
    </local:M4EventHandler.EventParamsList>
</local:M4Button>
```

En el atributo *Action* se indica la acción a ejecutar, en este caso *Execute*. Y en el control personalizado *M4Action_dataprops* será donde se controle la funcionalidad de dicha acción.

Como se acaba de comentar, el control personalizado *M4Button* no conoce que *Click* es un evento que deba ejecutar. Pero tras la inclusión del código que gestiona los eventos, se bindará la propiedad *Click* del control *M4Button* que se acaba de definir, con un método de la instancia del control *M4Button* definido, el cual procesará la acción correspondiente a realizar.

3. Por último, se muestra el pedazo de código correspondiente al XAML Operativo, conseguido tras la inclusión del código necesario para controlar los eventos en el XAML de Visualización anterior, aportando la posibilidad de ejecutar la funcionalidad requerida por el botón.

Los controles con eventos asociados tendrán un identificador de evento con un nombre concreto, por ejemplo, *Click*. Por otro lado, el nombre de la función que gestiona los eventos en cada control será *EventClick*. Al disponer anteriormente de la propiedad *Click* en el control *M4Button*, lo que se consigue es bindar la propiedad *Click* de la clase *M4Button*, con el método *EventClick* de la **instancia** de la clase.

```
<local:M4Button FontFamily="Ms Sans Serif" FontSize="8pt" Height="41"
                help:HelpProvider.HelpString="" VerticalAlignment="Top"
                HorizontalAlignment="Left" Margin="151,181,0,0" Cursor=""
                Content="Llamada al canal" Width="199"
                EnabledWhenRegisterIsDeleted="False"
                EnabledWhenRegisterIsNew="True"
                EnabledWhenRegisterIsNormal="True"
                EnabledWhenRegisterIsModified="True">

    <XamlParser:ElementBinder.ElementList>
        <XamlParser:ElementList>
            <XamlParser:Event EventName="Click" EventHandler="EventClick" />
        </XamlParser:ElementList>
    </XamlParser:ElementBinder.ElementList>
    <local:M4EventHandler.EventParamsList>
        <local:M4EventParamsList>
            <local:M4EventParams EventId="Click" >
                <local:M4Action_dataprops Name="BOTONHOLA" Action="Execute" />
            </local:M4EventParams>
        </local:M4EventParamsList>
    </local:M4EventHandler.EventParamsList>
</local:M4Button>
```

En este caso, el identificador del evento tiene como nombre *Click*.

¿Qué aporta el proyecto XAMLParser?

En los dos primeros ciclos del proyecto, la idea llevada a cabo consistía en generar y almacenar un archivo XAML, el cual sería ejecutado mediante un ejecutable generado por el propio proyecto. Todo esto quedaría empaquetado y grabado en el repositorio de la compañía.

Actualmente, la línea de desarrollo a seguir consiste en generar un XAML en cliente justo antes de ser ejecutada la presentación, es decir: generar el XAML, ejecutarlo y el cliente, sin necesidad de almacenar dicho fichero XAML, será capaz de visualizar e interactuar directamente con la presentación en WPF (.NET). En este sentido, el código generado en XAML deberá ser capaz de aportar toda la funcionalidad requerida por el cliente, por lo que la gestión de eventos debe ser controlada de algún modo en dicho XAML. Como se ha comentado hasta ahora, este modo de gestionar los eventos consiste en incorporar un código XAML para cada control que disponga de eventos asociados, que se encargue de realizar los correspondientes bindados a las acciones a ejecutar de cada evento.

10.9. La solución: ejemplos de migración.

El objetivo de migración software propuesto en la planificación inicial, consistía en la migración de una presentación relativamente complicada, la cual dispone de unas tres mil líneas de código en su XML de ejecución. Dicha presentación consta con un gran número de objetos a migrar, por lo que para poder resolver los problemas de migración de modo correcto, fue necesario aislar dichos problemas.

De ese modo, se cuenta con presentaciones dispares con fines diferenciados, las cuales han ido obteniendo una mayor dificultad con el tiempo, a medida que se iban resolviendo problemas mediante presentaciones más sencillas.

A continuación se muestran algunas de las presentaciones más utilizadas en el proceso de reingeniería del software que nos compete, las cuales han sido diseñadas con el fin de aislar la complejidad en el proceso de migración.

PRESENTATION_UC3M_BOTON

Esta presentación fue la primera utilizada para comprobar que la gestión de eventos era correcta. Consiste en un botón que al pulsarle, invoca un evento cuya acción reside en levantar una nueva ventana que muestra un mensaje.

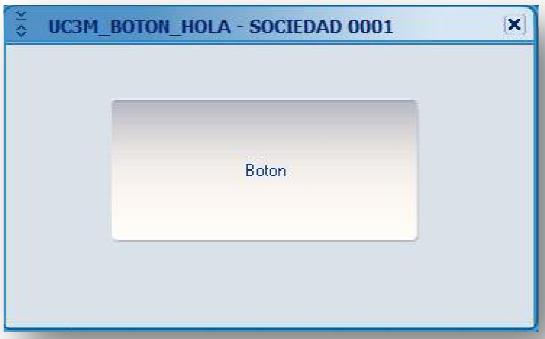


Ilustración 28 - PRESENTATION_UC3M_BOTON

Tras pulsar en el botón, se chequeará en XAML el código explicado en el subapartado del componente XAMLParser, en el apartado 10.8. De este modo se accederá al método *EventClick* del botón, y se comenzará el procesado del evento, el cual finalizará con la ejecución de la acción programada.

Este es el resultado tras pulsar el botón:

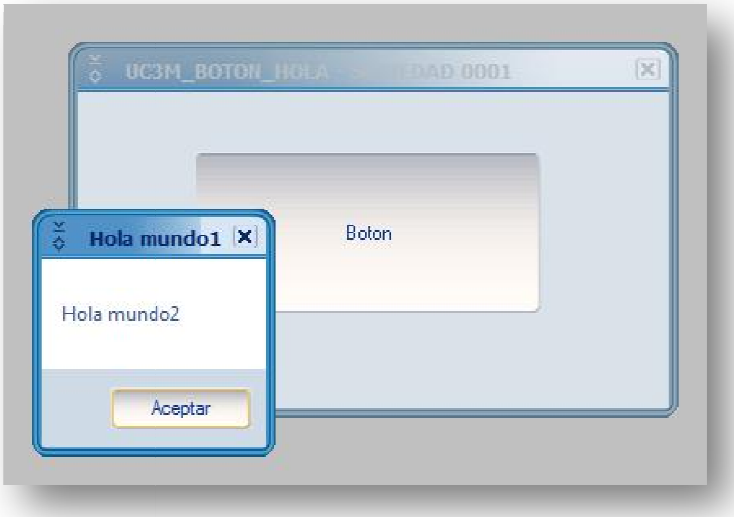


Ilustración 29 - PRESENTATION_UC3M_BOTON tras pulsar el botón

El procesado del evento en su code-behind es como sigue:

- El botón tiene asociado un evento *Click*. Cuando se dispara ese evento, es decir, cuando el usuario hace click sobre el botón, se activa la función *EventClick* del control personalizado M4Button.
- La función *EventClick* realiza una llamada a otra función, contenida en una clase genérica que representa todos los eventos, que se encarga de procesar el evento en cuestión.
- El evento a su vez realiza una llamada a otra función que se encarga de procesar la acción a realizar.
- Dicha acción tendrá asociada una clase definida como control personalizado en el componente M4DynamicControl. Dentro de esa clase, existe un método que en función de la acción indicada en el código XAML Operativo, ejecutará una labor concreta.

Este es el procesado para el caso del botón, pero cualquier control que pueda tener asociado un evento, se tratará del mismo modo, independientemente del evento a ejecutar y de la acción a realizar.

PRESENTATION_UC3M_TAB

Una vez concluido el desarrollo de la gestión de eventos, el propósito consistió en la migración de una presentación que, dentro de una pestaña, disponía de una barra de herramientas con opciones de control de datos (controles sobre los registros como: crear uno nuevo, eliminarlo, deshacer los cambios sobre un registro, ordenarlos, filtrarlos, etc.), opciones de edición (cortar, copiar, pegar) y opciones de navegación (dirigirse al registro elegido). Además, dicha barra de herramientas consta con un control que indica el registro en el que se encuentra el usuario del total de registros almacenados en la base de datos, y otro control que indica el estado del registro (nuevo, borrado o modificado). Aparte, la presentación constaba con dos cajas de texto que mostraban los valores de una tabla de sexos. Dicha tabla consiste en un identificador, como clave primaria, acompañado de un nombre, que están almacenados en la base de datos de la compañía.

La dificultad de esta presentación radica en, además de disponer de muchos más controles para ser migrados, conocer el modo de desarrollar en code-behind la funcionalidad requerida por cada botón de la barra de herramientas.

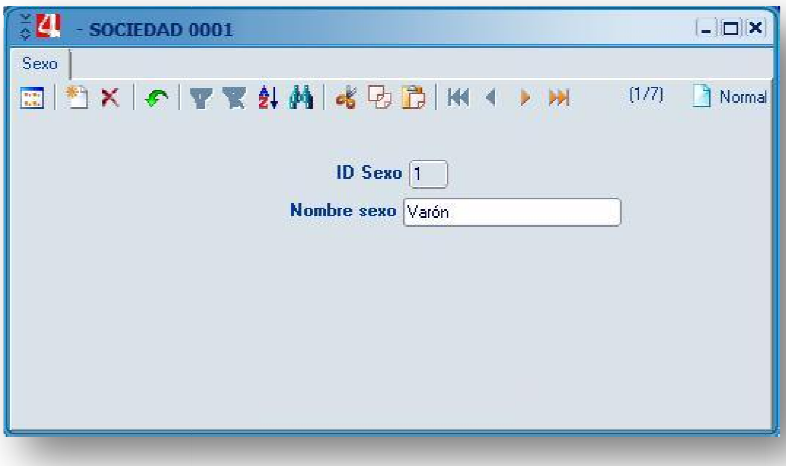


Ilustración 30 - PRESENTATION_UC3M_TAB

Como puede observarse en la siguiente figura, la migración de la presentación al entorno .NET se desarrolló con éxito, aportando además a la presentación migrada la funcionalidad con la que constaba la presentación antigua desarrollada en OBL. Algunos de los botones, visualmente fueron migrados con éxito, pero debido a que aportarles de su funcionalidad suponía un gran número de horas de análisis y desarrollo, se decidió abordar ese aspecto en el siguiente ciclo del proyecto.

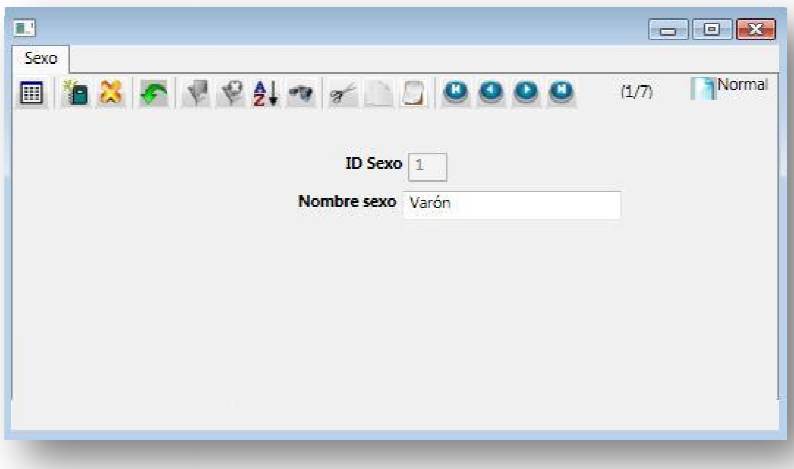


Ilustración 31 - PRESENTATION_UC3M_TAB migrada

PRESENTATION_UC3M_GENDER

Cuando el proceso de migración se consideró relativamente avanzado, se desarrolló una presentación en OBL muy similar a la presentación objetivo a migrar. Esta presentación ya consta con menús desplegables, tablas de datos, árboles de acciones, división en zonas de la presentación, etc.

Al igual que ocurrió con la anterior presentación, se decidió que el proceso de migración en este ciclo se iba a centrar más en la visualización de los controles que en su funcionalidad, por lo que a pesar de poder observar que la presentación está totalmente migrada, la funcionalidad de los menús no está desarrollada.

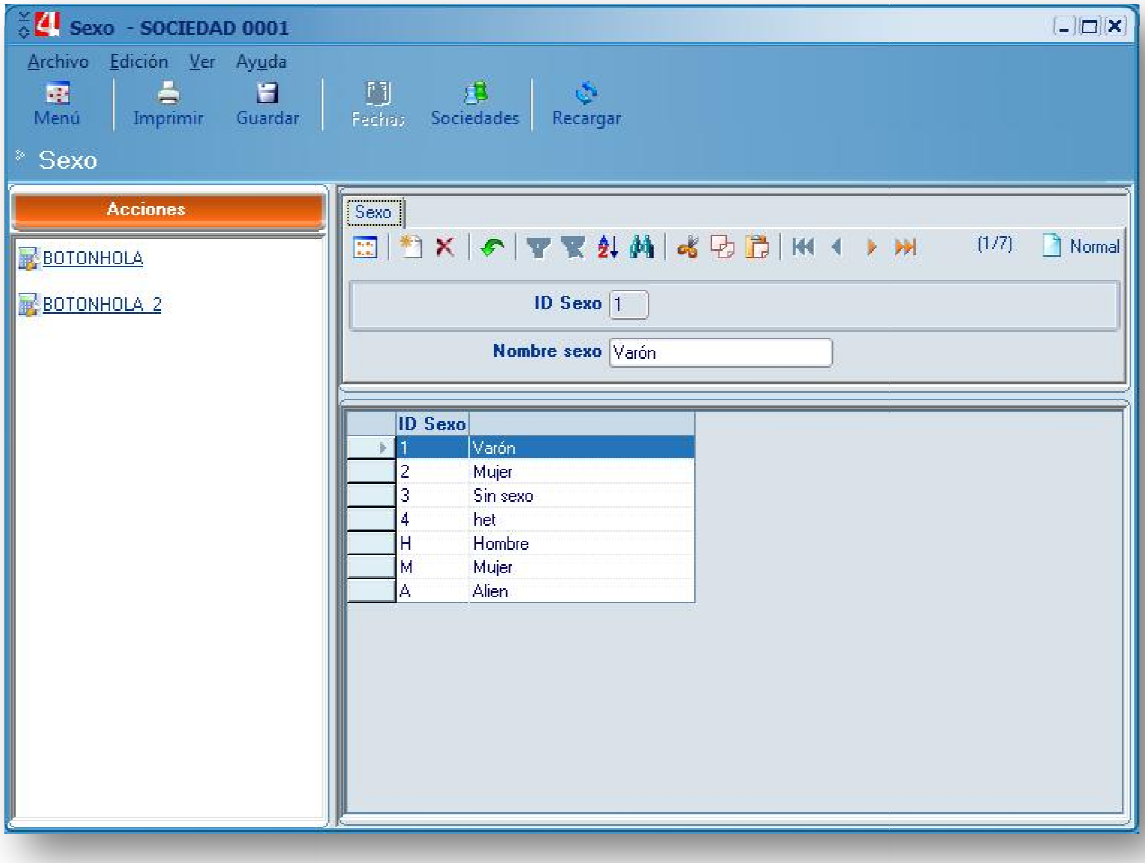


Ilustración 32 - PRESENTATION_UC3M_GENDER

En esta presentación se puede comprobar cómo la visualización es prácticamente idéntica a la presentación original. Ciertos elementos que pueden observarse en esta presentación migrada, los cuales no son visibles en la presentación original, aparecen por el hecho de que en OBL existen propiedades que hacen invisibles ciertos objetos. Dichas propiedades no fueron analizadas ni desarrolladas, por lo cual, en el proceso de migración se omiten, produciendo que los elementos que dispongan de dichas propiedades no realicen su función de invisibilidad, como es el caso del apartado "Enlaces" que se encuentra en la división izquierda de la presentación, o de los botones "Tomar decisión" y "Lista" que se encuentran en la barra de menú.

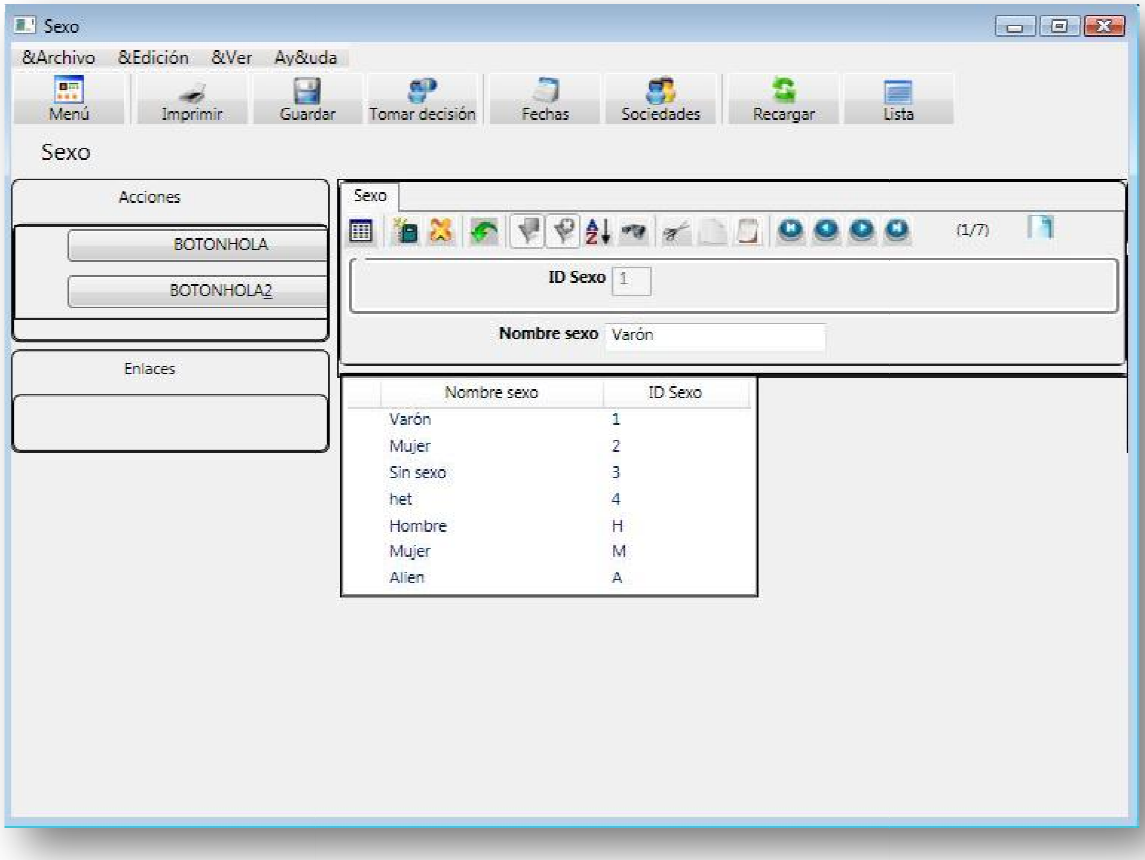


Ilustración 33 - PRESENTATION_UC3M_GENDER migrada

PRESENTATION_UC3M_FRAMEWORK

Esta presentación se trata de la presentación objetivo a ser migrada en el proyecto de reingeniería desarrollado.

Consiste en una interfaz de usuario encargada del mantenimiento de los empleados de una compañía, con el fin de mantener almacenados y fácilmente accesibles sus datos personales y habilidades. Esta presentación, en orden de aparición, dispone de un menú desplegable; una barra de menú; el título de la presentación; un apartado de acciones; un carril de navegación; una pestaña con diferentes controles como una barra de herramientas, etiquetas, cajas de texto, etc.; y finalmente una apartado que muestra los datos de los registros con un formato de tabla. Además de la funcionalidad de la barra de herramientas para poder crear registros nuevos, así como modificarlos y navegar por ellos, se ha aplicado la funcionalidad a la tabla de bindar el ítem en el que nos encontremos de la misma a los campos mostrados en las cajas de texto contenidas en la pestaña, de modo que el valor de dichos campos variará en función del registro en el que nos situemos en la tabla.

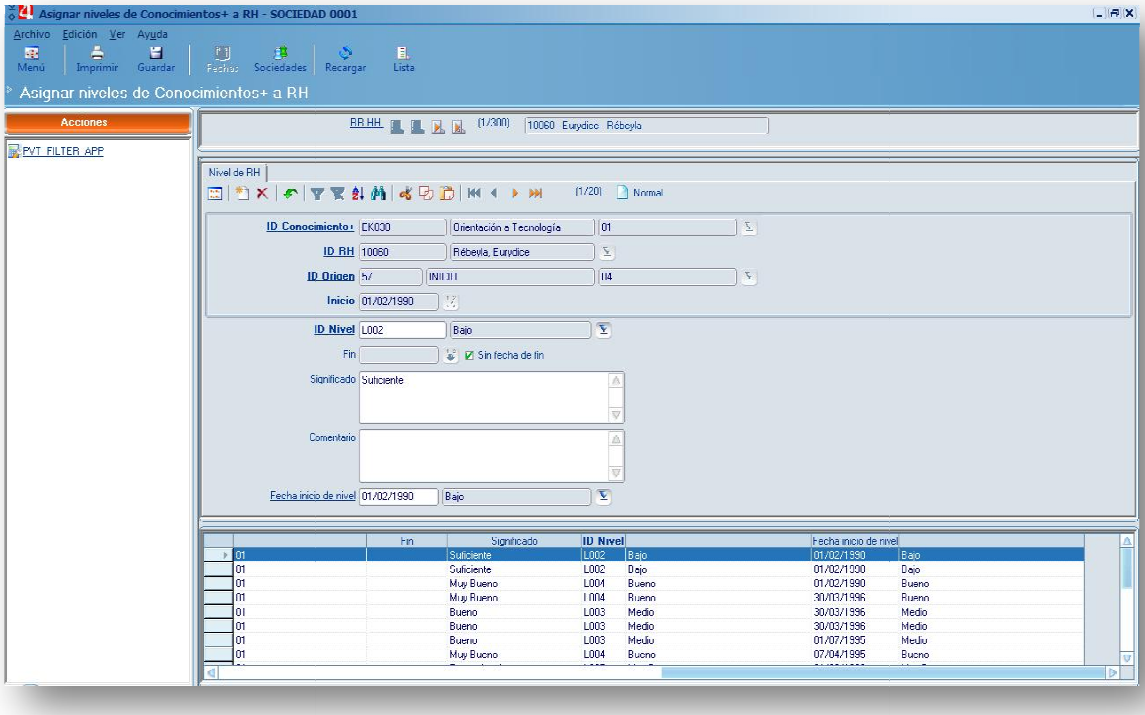


Ilustración 34 - PRESENTATION_UC3M_FRAMEWORK

Tal y como sucedía con la presentación *PRESENTATION_UC3M_GENDER*, se puede comprobar cómo la visualización es prácticamente idéntica a la presentación original. Y del mismo modo, ciertos elementos que pueden observarse en esta presentación migrada, los cuales no son visibles en la presentación original, aparecen por el hecho de que en OBL existen propiedades que hacen invisibles ciertos objetos, y dichas propiedades no fueron analizadas ni desarrolladas para su migración. Además, es evidente que ciertas cajas de texto aparecen vacías en la presentación migrada. Esto es debido a que el objeto encargado de la validación no ha sido completamente migrado en cuanto a su funcionalidad se requiere para este ciclo, posponiendo su funcionalidad para el ciclo siguiente. Éste es el motivo por el que en la tabla aparecen ciertas celdas con el valor *System.Data.DataRowView*, debido a que se espera la validación de un campo, pero no puede llevarse a cabo dicha validación porque el objeto empleado de cierta labor no fue migrado.

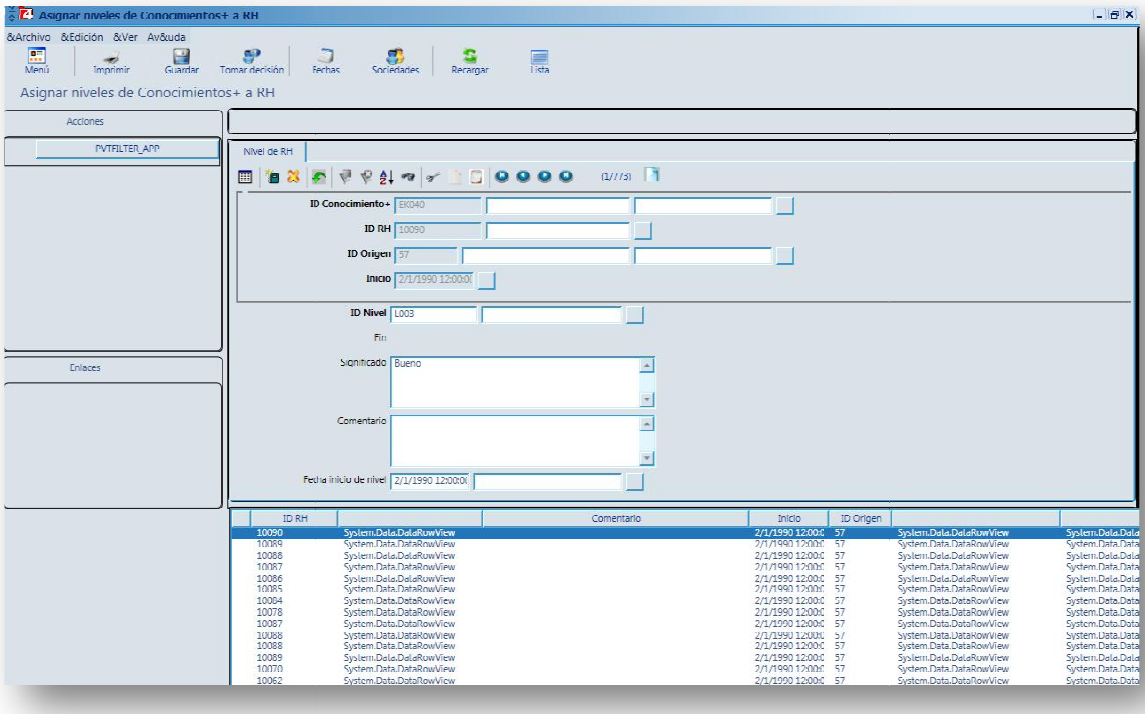


Ilustración 35 - PRESENTATION_UC3M_FRAMEWORK migrada

10.10. La solución: peculiaridades.

El presente apartado sirve para exponer ciertas singularidades que no han sido comentadas en el desarrollo de la solución.

- Pese a que pueda parecer extraño, debido a la gran cantidad de líneas de código desarrolladas, la mayor parte del tiempo empleado para el desarrollo de la solución no ha consistido en la generación del código encargado de llevar a cabo la migración, sino en el análisis de los controles para comprender perfectamente cuál era su cometido, y en la investigación acerca de la correspondencia de dichos controles en WPF.
- La solución dispone de un fichero LOG que almacena aquellas excepciones dadas durante la ejecución del programa. En el ciclo anterior del proyecto, cuando un control no migraba cierta propiedad, se escribía en el fichero de LOG que no se había producido la migración de la misma. En el ciclo 03 que nos compete se ha cambiado dicha funcionalidad, ya que si la propiedad no existe en OBL, no tiene porqué producirse una excepción al intentar migrar la propiedad, tal y como ocurría en el ciclo anterior. Esto ha supuesto que la migración suceda de un modo mucho más veloz que anteriormente.
- El comportamiento de los controles puede variar en función del contexto en el que se encuentren. Esto es, que dependiendo del contenedor de un control, éste puede tener un comportamiento diferente. Por lo tanto, se decidió que al realizar la migración de los controles, estos supieran en todo momento el contexto en el que se encontraban, para así migrarse de un modo u otro.
- Existe una clase M4Control que actúa como control genérico. De esta manera, se podrán definir los aspectos comunes de los controles para que cada control, al heredar de la clase mencionada, dispongan de dichas generalidades.
- En cuanto a las presentaciones objetivo de ser migradas, la funcionalidad de las mismas en el entorno .NET se ha tratado de mejorar, siempre manteniendo la premisa de no desviarse del objetivo de la función en sí misma.
- Otros aspectos de las presentaciones diseñadas en OBL también han sido mejorados, ya que en la migración, todos los controles se han tratado de modo homogéneo en cuanto a su migración. Sin embargo, en las presentaciones de la plataforma ciertas propiedades que deberían comportarse del mismo modo para cualquier objeto, no lo hacían, teniendo un comportamiento diferente en función del objeto tratado.
- El aspecto prioritario en la migración de los objetos se ha basado en su visualización, debido a que el jefe de proyecto consideraba conveniente que el

usuario no advirtiese cambios entre la presentación antigua y la presentación migrada. Y en caso de observar cambios, que estos fuesen para visualizar la presentación de un modo más elegante, no para cambiar la forma en que la presentación estaba diseñada.

- En el calendario, mostrado en el apartado 9.1, se muestran las tareas finalizadas. Dichas tareas ciertamente están completamente finalizadas en función a lo acordado entre el jefe de proyecto y el equipo desarrollador (en el que también estaba inmiscuido dicho jefe de proyecto). Por ello, los objetos que aparecen como completados, están finalizados según nuestros propios criterios de finalización para el presente ciclo del proceso de reingeniería.
- En ciertas presentaciones se puede comprobar cómo los colores de fondo, la forma de los botones, el tipo o tamaño de letra, u otros aspectos relacionados con la visualización, no siguen el mismo esquema que las presentaciones originales. Esto es debido a que se ha decidido definir estilos propios de la compañía que, aplicados a las presentaciones migradas, aportarán una apariencia más atractiva y elegante a las presentaciones.

CAPÍTULO V. CONCLUSIONES Y LÍNEAS FUTURAS.

11. Conclusiones.

El presente apartado pretende exponer las conclusiones extraídas tras la finalización del proyecto desarrollado. En primer lugar se muestran una serie de conclusiones generales acerca del proyecto, y posteriormente se mostrarán las conclusiones obtenidas acerca de los problemas identificados para el proyecto de reingeniería desarrollado.

11.1. Conclusiones generales.

El trabajo realizado radica en la aplicación de un proyecto de reingeniería del software, consistente en la migración de una tecnología anterior a una tecnología puntera en la actualidad, con el fin de hacer uso de los beneficios aportados por el nuevo entorno utilizado.

El proceso de reingeniería consta de varios ciclos. Para un proyecto de dimensiones considerables, como en el que nos encontramos, resulta conveniente una descomposición del mismo en procesos, ciclos, actividades y tareas que se centren en objetivos abordables que contribuyan a mejorar la eficiencia del método de trabajo, a la vez que permiten conocer en qué situación se encuentra el proyecto en todo momento. El mostrado en el presente Proyecto Fin de Carrera se trata del ciclo 03. A modo de resumen, el primero de ellos consistió en un trabajo de investigación acerca de qué se quería hacer y cuál era el mejor modo de abordarlo; mientras que el segundo ciclo se basó en la creación de un marco de trabajo que facilitase el proceso de reingeniería del software, y, además, se desarrolló el inicio de la herramienta migradora del software.

En este tercer ciclo del proyecto de reingeniería, el objetivo principal consiste en la migración de los controles encontrados en la presentación objetivo de ser migrada, tanto visual como funcionalmente. Gracias a disponer de un framework adaptado y adaptable al proceso de reingeniería que nos compete, el trabajo realizado se pudo desarrollar con mayor rapidez y eficiencia a la esperada en un primer momento.

Uno de los objetivos del proceso de reingeniería desarrollado reside en permitir la migración de cualquier presentación desarrollada en OBL. De este modo, las presentaciones candidatas a ser migradas son infinitas. Ante la necesidad de delegar al usuario la responsabilidad de realizar la migración de la presentación, se

decidió que uno de los propósitos principales en este proyecto fuese que el usuario se viese implicado lo menos posible en el proceso de migración. Por ello, su única labor deberá residir en elegir la presentación a migrar, y ejecutar la tarea de migración mediante un doble click sobre un "botón" destinado para tal fin.

Realizar la planificación y las estimaciones temporales y de recursos para un proyecto de migración del software, es una tarea inicial muy relevante para el correcto desarrollo posterior de la migración. Resulta necesario hacer un análisis en profundidad sobre el sistema a migrar para obtener los conocimientos técnicos suficientes que permitan tomar decisiones adecuadas. Además, se estima necesario un estudio del entorno dónde se va a trabajar, para conocer de antemano qué personas pueden servir de apoyo al proyecto en caso de que se requiera su ayuda. Este punto resulta especialmente importante en el caso de que el equipo de desarrollo sea externo a la empresa.

Una conclusión importante que podemos obtener en el trabajo de reingeniería realizado es que no se deben asentar ideas preconcebidas, ni tratar de continuar el mismo esquema seguido en el pasado sin asegurarse de que dicho esquema es el ideal. De hecho, el enfoque de la arquitectura del sistema de migración en el presente ciclo, varió en gran medida en comparación al ciclo anterior. Actualmente, cualquier presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un ejecutable (.EXE) ni efectuar almacenamientos en el repositorio de la compañía, permitiendo, de esta manera, que las modificaciones en las presentaciones sean traducidas en el momento de ser necesitadas, eliminando los problemas de consistencia en el almacenamiento persistente entre las presentaciones. De este modo, la necesidad de acumular ficheros XAML fue eliminada, así como la creación de un fichero que necesitase de su ejecución posterior para la posible visualización de la presentación migrada.

En definitiva, el cliente simplemente necesitará:

- El framework .NET.
- Las librerías que permiten la migración.
- La plataforma donde se encuentra el Diseñador de presentaciones, y donde se encuentra a su vez el apartado que ofrece la migración de la presentación deseada.

Por último, se desea finalizar este aparado de conclusiones generales indicando que resulta totalmente conveniente instruir a los integrantes del equipo de desarrollo externos a la empresa, acerca de aspectos básicos sobre la herramienta que se debe migrar. A pesar de haber terminado el ciclo y haber desarrollado con éxito el proceso de migración, a partir de este momento comenzarán a aparecer nuevos elementos con mayor complejidad con respecto al conocimiento de aspectos internos de la compañía se refiere, por lo que un proceso de formación previo se presenta necesario.

11.2. Conclusiones a los problemas identificados.

En el apartado 7 del presente Proyecto Fin de Carrera, se enumeraron una serie de problemas encontrados en la realización de este proyecto. A continuación se enumeran las conclusiones obtenidas para cada uno de estos problemas:

- **Insuficiente formación del equipo de desarrollo.**

Teniendo en cuenta que el equipo de trabajo era externo a la empresa, se requiere un proceso de formación centrado en los aspectos fundamentales que van a influir en el proyecto de reingeniería para la migración del software. Debido a que dicha etapa de formación no existió, se sucedieron inevitables retrasos en el desarrollo del proceso de reingeniería. Por lo tanto, las fases de adaptación se prueban como necesarias.

- **Cambio de enfoque en el modelo de migración.**

El cambio de arquitectura del sistema de migración fue un factor de desarrollo beneficioso debido a que el producto final ofertado es mucho más cómodo y atractivo para el usuario. Además, el desarrollador también se ve beneficiado por las ventajas del mismo.

- **Código a migrar distinto al de la anterior etapa.**

Realizar la migración de código XML a código XAML ha resultado ser mucho más productivo que la migración de código OBL a código XAML. El motivo es que el código XML que define idénticamente el código OBL, sigue una estructura muy

similar al código resultado de la migración. Por lo tanto, resulta conveniente estudiar previamente las distintas posibilidades de migración, de modo que la traducción a realizar sea lo más eficiente posible.

- **Problemas con el nuevo código a traducir.**

El código XML de ejecución a migrar, que representaba fielmente el código OBL, se vio afectado por una serie de problemas debido a que en la traducción de código OBL a código XML se producían ciertos errores. A pesar de ello, se ha concluido que resultó mucho más efectivo emplear un leve tiempo en resolver dichos errores, y así posteriormente seguir utilizando dicho código XML para la migración.

- **Indicar en la plataforma qué presentación se pretende migrar.**

Anteriormente a la resolución del presente Proyecto Fin de Carrera, la presentación objetivo a migrar debía especificarse en el propio código fuente que hace posible la migración. Enfocando el proyecto de migración hacia el usuario, resulta mucho más conveniente especificar en la propia plataforma propiedad de la compañía la presentación original que se desea migrar, eliminando la necesidad de modificar el código fuente que hace posible la migración.

- **Exceso de objetos creados mediante controles personalizados.**

Muchos de los objetos creados anteriormente mediante controles personalizados no deberían necesariamente haberse definido de ese modo. Por ello, sólo los controles que tengan eventos asociados, o necesiten acceder a bases de datos, o aspectos similares relacionados, serán los controles que se definan mediante un control personalizado. Por lo tanto, se deberán crear controles personalizados equivalentes sólo a aquellos objetos que absolutamente lo necesiten para su correcto funcionamiento. De este modo, las librerías ofertadas al cliente tendrán un menor tamaño y una menor complejidad.

- **Control de eventos.**

Para utilizar el menor code-behind posible, se ha definido una estructura en XAML que gestiona el control de los eventos y las acciones asociadas a los mismos. Esto

es algo novedoso que no existe en otros entornos, ya que se controla en el propio código XAML aspectos tales como la herencia y el polimorfismo.

- **Revisión del código desarrollado en la anterior etapa.**

Una conclusión importante que se puede obtener con este proyecto de reingeniería es que se considera absolutamente necesario realizar una revisión del código desarrollado en los ciclos anteriores del proceso. Dicho código debe estar expuesto a revisión para la detección de errores, empleando en tiempo que se considere oportuno para ello, ya que de otro modo, se puede continuar cometiendo los mismos errores en los sucesivos ciclos del proyecto, provocando que la solución final no sea la que se espera.

- **Escasez de tiempo para la revisión del código desarrollado en la anterior etapa.**

Debido al deseo por encontrar resultados inmediatos en la migración, se ha considerado prioritario el desarrollo del código para la migración de nuevos elementos antes de realizar una revisión profunda acerca del código ya desarrollado en la anterior etapa. Esto ha supuesto cargar con errores durante gran parte del presente ciclo, que no han sido resueltos hasta el final del mismo, cuando se decidió emplear un tiempo adecuado para la revisión total del código de migración. Si dichos errores se hubiesen resuelto con anterioridad, el proceso de migración de los nuevos objetos hubiese sido más rápido y eficiente. Por ello, para todo proceso de reingeniería se considera absolutamente necesario dedicar un tiempo oportuno a la revisión del código ya desarrollado, más aún si ese código va a ser la base del proyecto que compete.

- **Escasez de tiempo para la documentación.**

La documentación es considerada una de las tareas más importante en un proyecto de estas características. El hecho de que el sistema a migrar carezca de la documentación adecuada, además de haber sido desarrollado sin tener en cuenta ninguna metodología, dificulta las labores de investigación por parte del equipo de desarrollo, alargándolas inevitablemente con los consabidos retrasos que esto provoca. Por ello, debido a que la metodología seguida reservaba un apartado importante a la tarea de documentación, ésta se ha realizado de manera idónea con

el fin de servir como soporte al desarrollador, así como facilitar la inclusión de nuevos miembros al equipo de desarrollo.

- **Escasez de documentación en los objetos de las presentaciones.**

Debido a que el sistema a migrar carece de una documentación adecuada, la investigación acerca de la funcionalidad de muchos de los objetos que aparecían en las presentaciones ha necesitado de un tiempo mayor del esperado. Este hecho pone de manifiesto la importancia de una correcta documentación para cualquier sistema de información.

- **Dispersión del conocimiento.**

Un elevado número de las dudas surgidas durante el proceso de migración debían ser resueltas por miembros de la compañía, que en muchas ocasiones desconocían las respuestas a las preguntas formuladas por el equipo de desarrollo, debido a que quien conocía las respuestas, ya no se encontraba presente en la compañía. Este hecho pone de manifiesto la importancia de una correcta documentación para cualquier sistema de información.

- **Se prima la inmediatez respecto a la calidad.**

Por último, una conclusión que se debería aplicar a cualquier desarrollo, no sólo a los proyectos de reingeniería, sería que no siempre es importante desarrollar "gran cantidad", sino que aquello que es desarrollado tenga la calidad requerida. En el presente proyecto se ha primado el hecho de migrar gran cantidad de objetos, a pesar de que dicha migración no fuese completada hasta el final, o incluso no fuese totalmente correcta. Se decidió migrar visualmente el mayor número de controles, sin investigar apenas las propiedades de muchos de estos y, por lo tanto, omitiendo dichas propiedades en la migración. Esto es un error que no se debe cometer ya que, aunque el proceso de migración conste con varios ciclos, no se pueden delegar responsabilidades a los posteriores ciclos, o por el contrario, el número de ciclos empleados para completar la migración puede verse aumentado en mayor medida de lo esperado.

12. Líneas futuras.

En este punto se proponen una serie de líneas de trabajo que traten de ampliar y enriquecer los frutos conseguidos.

En primer lugar, el proceso de migración podría verse complementado con una definición de estilos visuales que dotaran a la presentación migrada de un aspecto mucho más elegante. Esto, aplicado a las ventajas que aporta el entorno .NET, haría de este proyecto un producto aún más interesante para el cliente.

Tras la aplicación de estilos visuales sobre la presentación migrada, los resultados son señaladamente favorables:

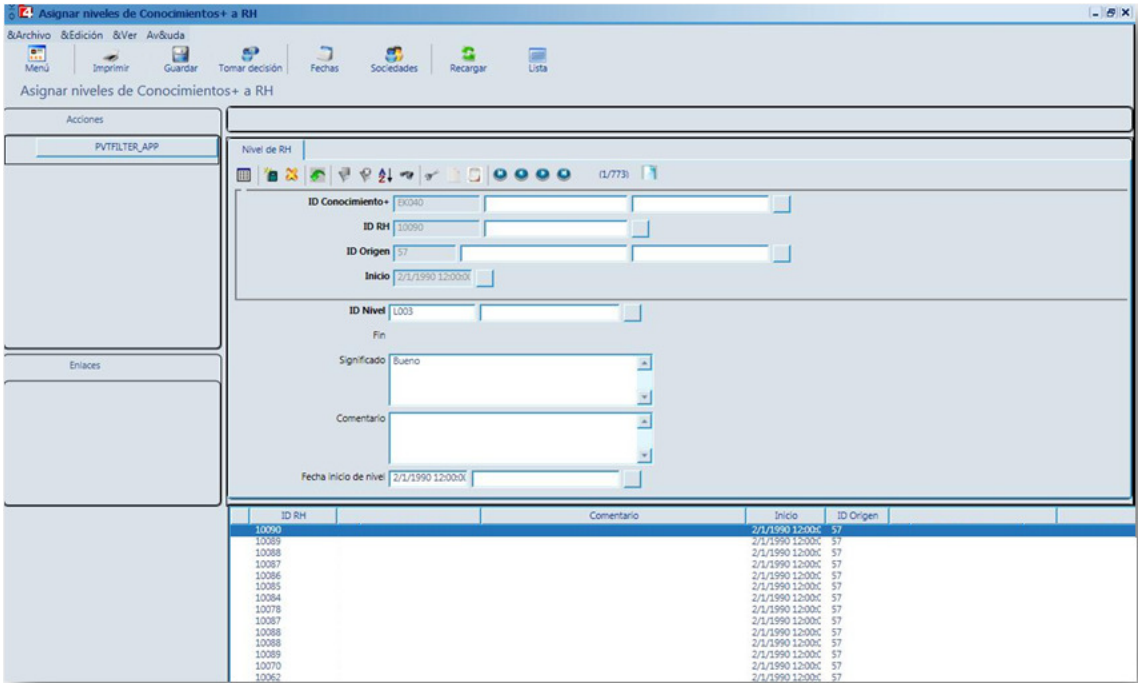


Ilustración 36 - PRESENTATION_UC3M_FRAMEWORK sin estilos visuales

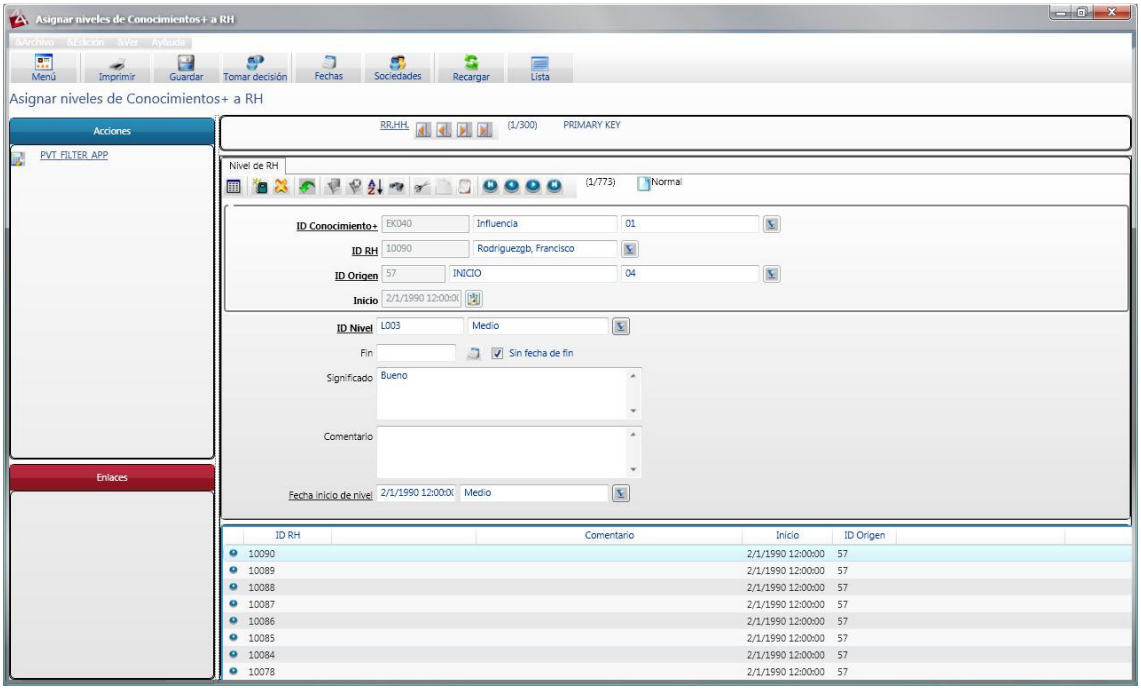


Ilustración 37 - PRESENTATION_UC3M_FRAMEWORK con estilos visuales

En segundo lugar, una tarea que debería considerarse prioritaria sería mejorar la funcionalidad migrada. A pesar de que se ha migrado gran funcionalidad asociada a los botones, hasta ahora se ha primado la migración de los objetos de modo visual, por lo que se han migrado gran cantidad de objetos pero muchos de ellos sólo visualmente. Esto es un hecho que al cliente no le servirá de gran ayuda, ya que su deseo es poder interactuar con la presentación y no simplemente poder observarla.

En tercer lugar, se considera que sería un gran avance realizar la migración de las presentaciones a dispositivos móviles, tales como teléfonos móviles o pda's. Además, también sería un gran avance desarrollar dicha migración modo Web, de manera que no se necesitasen librerías para realizar la migración de las presentaciones deseadas por el cliente. Los tiempos cambian, y cada vez son más las personas que no trabajan en una oficina, y no necesitan cargar con un ordenador portátil, por lo que con su pequeño dispositivo móvil podrían realizar el proceso de migración en cualquier lugar donde se encuentren.

CAPÍTULO VI. APÉNDICES.

I. Apéndice A. URD.

En el presente apéndice se recoge el Documento de Requisitos de Usuario (URD), dónde se muestra el conjunto de requisitos establecidos para el constructo de migración automática desarrollado.

Esta tarea de recopilación de requisitos de usuario se lleva a cabo con el de fin de comprender qué requisitos demanda el usuario final. Proporcionará una valiosa aportación que permitirá llevar a cabo una toma de decisiones acorde con las necesidades reales del usuario y que satisfagan, en la medida de lo posible, los requerimientos de éste.

ISDAC

Integración, Soporte y Difusión de Aplicaciones Complejas

USER REQUIREMENTS DOCUMENT

Referencia:	ISDAC_AMC.URD_P01C03_20081020
Versión:	0.01
Creación:	20-10-2008
Última Modificación:	29-03-2009
Preparado por:	Luis Miguel Esteban
Revisado por:	UC3M1

Resumen

El presente documento plasma el conjunto de requisitos establecidos para el constructo de migración automática desarrollado en el Ciclo C03 del Proceso P01 del proyecto ISDAC. Este constructo de migración tiene como propósito la migración de presentaciones desarrolladas por la plataforma de desarrollo propiedad de la compañía, a un formato WPF.

Hoja de Estado del Documento

1. *ISDAC.* Integración, Soporte y Difusión de Aplicaciones Complejas. User Requirements Document.		
2. Referencia del documento: ISDAC_AMC.URD_P01C03_20081020		
3. Versión	**4. Fecha**	**5. Motivación del Cambio**
0.01	20-10-2008	Primera versión interna del documento.

Tabla 1. Hoja de estado del documento.

Registro de Cambios

1. Versión	2. Fecha	3. Autor
0.01	20/10/2008	Luis Miguel Esteban
4. Descripción del Cambio		
• Primera versión interna del documento.		

Índice de Contenidos

Resumen ..... **i**
Hoja de Estado del Documento ..... **i**
Registro de Cambios ..... **ii**
Índice de Contenidos ..... **iii**
1 Introducción **1**
1.1 Propósito del documento 1
1.2 Ámbito del software 1
1.3 Definiciones, acrónimos y abreviaturas 2
1.3.1 Definiciones 2
1.3.2 Acrónimos 2
1.3.3 Abreviaturas 2
1.4 Referencias 2
1.5 Estructura del documento 2
2 Descripción General **4**
2.1 Perspectiva del producto 4
2.2 Capacidades generales 4
2.3 Restricciones generales 6
2.4 Características del usuario 6
2.5 Entorno operacional 7
3 Requisitos específicos **8**
3.1 Requisitos de capacidad 9
3.2 Requisitos de restricción 12

1 Introducción

Este documento contiene la especificación de requisitos de usuario del componente para la automatización de la migración de presentaciones de la plataforma de desarrollo propiedad de la compañía, a WPF.

Esta sección establece el propósito del documento, el ámbito del sistema desarrollado, así como las definiciones, las abreviaturas, los acrónimos y las referencias que aparecen en el mismo.

1.1 Propósito del documento

El documento *User Requirements Document* es uno de los productos de la tarea *Automatic Migration Component Construction* enmarcada en el Ciclo C03 del Proceso P01 de la migración del servicio de presentaciones de la plataforma de desarrollo propiedad de la compañía.

1.2 Ámbito del software

El software a desarrollar tomando como base los requisitos definidos en el presente documento tiene como objetivo la traducción a WPF de una presentación OBL generada por la plataforma de desarrollo propiedad de la compañía para su posterior visualización en el entorno de .Net. El producto obtenido será una solución de .Net que incluirá toda la funcionalidad descrita a partir de la generación de diferentes proyectos. Así, el principal beneficio del componente es aglutinar la funcionalidad de migración y exhibición de la presentación traducida. Dicha presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET.

1.3 Definiciones, acrónimos y abreviaturas

En esta sección se recogen las definiciones, acrónimos y abreviaturas que aparecen en este documento.

1.3.1 Definiciones

No aplicable.

1.3.2 Acrónimos

ESA
European Space Agency (Agencia Espacial Europea)

XAML
eXtensible Application Markup Language

XML
eXtensible Markup Language

1.3.3 Abreviaturas

No aplicable.

1.4 Referencias

[1] ISDAC_REQ_P01C03_20081020. *Documento de Requisitos de Migración.* Process 01 Cycle 03.

[2] ISDAC_REC_P01C03_20081020. *Documento de Recomendaciones de Migración.* Process 01 Cycle 03.

1.5 Estructura del documento

El presente documento se estructura de la siguiente manera. La sección 2 incluye la descripción general del sistema en la que se establecerán la perspectiva del producto, las capacidades generales del mismo, las restricciones que se establecen tanto en su operativa como en su desarrollo, la descripción de las tipologías de usuario en relación al elemento software desarrollado y la

descripción del entorno operacional en la que el producto se va a instalar. La sección 3 expone los requisitos de usuario establecidos para el elemento a desarrollar. Dichos requisitos serán divididos en dos tipologías: requisitos de capacidad y requisitos de restricción.

2 Descripción General

En esta sección se describen los factores concernientes al componente para la automatización de la migración de presentaciones de la plataforma de desarrollo propiedad de la compañía y a los requisitos de usuario del mismo.

2.1 Perspectiva del producto

El producto desarrollado se encuentra enmarcado en el proyecto ISDAC en su anualidad 2008 que surge como evolución del proyecto KHAPYTAL y más concretamente del subproyecto KHAMIG. De forma más específica, el producto se desarrollará dentro del marco de reingeniería establecido para el proyecto KHAPYTAL y extendido para su uso en el proyecto ISDAC. Así los esfuerzos de la anualidad 2008 del proyecto ISDAC se sitúan en el Proceso P01 Ciclo C03 del framework de reingeniería. Así, atendiendo al ámbito del ciclo, se pretende generar una solución software que, interactuando con la plataforma de desarrollo propiedad de la compañía y a partir de una presentación dada en formato OBL, traduzca la presentación para el ámbito de un proyecto WPF y la ejecute en entorno .NET conservando la información y respetando las funcionalidades de la misma. Así, las interacciones del producto se producen, por un lado con la plataforma de desarrollo propiedad de la compañía y, por otro, con la herramienta .NET. Para llevar a cabo el trabajo descrito, se ha de tener en cuenta desde el punto de vista funcional y técnico todos los documentos generados en los Ciclos C01, C02 y C03 del Proceso P01 dentro del marco de migración/reingeniería que se ha propuesto para el subproyecto KHAMIG y se ha heredado en el proyecto ISDAC en la anualidad 2008.

2.2 Capacidades generales

El software desarrollado debe ser capaz de forma automática de, a partir de una presentación dada, ya sea antigua o de nueva creación, generar el código WPF equivalente y mostrarlo. Para ello se han establecido cuatro procesos diferenciados:

- En primer lugar se necesita refinar el código OBL de la presentación almacenado en el repositorio de la compañía para conseguir un XML de ejecución. Para ello se deberá utilizar *GetXMLFromPresentationId*.
- En segundo lugar, tomando como datos de entrada el código XML de ejecución de la presentación obtenido en el paso anterior, mediante el uso del componente *M4PresentationTranslator*, se debe generar un código XAML. Para ello usará el método *translateToNetPresentation*. El XAML generado se considera un XAML de visualización, es decir, carece de la funcionalidad necesaria, pero respeta y reproduce la visualización.
- En tercer lugar, es necesario que el código XAML de visualización sea enriquecido para obtener la funcionalidad requerida. Dicho proceso se llevará a cabo a partir de la utilización del componente *XAMLParser*. El resultado de la ejecución de la función *Parse* será un XAML que denominaremos operativo y que, además de las características de visualización comparables, reproduce la funcionalidad original de la presentación.
- Por último, se procede a mostrar el XAML operativo aprovechando las capacidades de .Net. Para ello, en primer lugar se carga el Framework de visualización y sus recursos, y en segundo lugar, se procede a llamar al método estándar de visualización de formatos XAML que muestra la presentación resultante.

2.3 Restricciones generales

Las restricciones generales que condicionan la construcción de componente para la automatización de la migración de presentaciones de la plataforma de desarrollo propiedad de la compañía se pueden separar en dos categorías. Por un lado se encuentran aquellas condiciones que emanan de los requisitos y las recomendaciones de migración generados en las tareas correspondientes de este Ciclo P01C03 y que se encuentran especificados en los documentos ISDAC_REQ_P01C03_20081020 [1] y ISDAC_REC_P01C03_20081020 [2] respectivamente. Y por otro están las restricciones propias del desarrollo entre las que cabe destacar:

- Se establecerá un proceso de desarrollo alineado con las especificaciones del estándar PSS-05-0 de la ESA.
- La codificación del sistema se realizará siguiendo el estándar de codificación Coding Guidelines de la compañía.
- El código C# y XAML de salida se generará de acuerdo a las directrices generales de desarrollo de esta tecnología, que especifican que se deberá trasladar al código XAML tanta funcionalidad como sea posible, tendiendo a minimizar la cantidad de código C#.
- Con el propósito de que la solución adoptada sea lo más flexible y estandarizada posible, se utilizarán en lo posible herramientas estandarizadas de Microsoft y las desarrolladas de forma específica por la compañía.

2.4 Características del usuario

El automatizador para la migración de presentaciones de la plataforma de desarrollo propiedad de la compañía va dirigido a todos los tipos de usuario que utilizan la plataforma de desarrollo propiedad de la compañía.

Teniendo en cuenta que el proceso de migración de una presentación es totalmente transparente al desarrollador de la misma, no es necesario que el usuario cuente con ningún tipo de experiencia o formación adicional acerca del proceso de migración.

2.5 Entorno operacional

El sistema se ejecutará en los puestos de desarrollador de la plataforma de desarrollo propiedad de la compañía. Los requisitos y las características de dicho entorno están especificados en la propia documentación técnica de la plataforma.

La comunicación entre el sistema y la plataforma de desarrollo propiedad de la compañía se realizará por medio de los interfaces que ésta última proporciona.

3 Requisitos específicos

En esta sección se especifican los requisitos de usuario identificados por el equipo de UC3M para el proyecto de automatización de la migración de presentaciones de la compañía.

Los requisitos de usuario se han catalogado en dos categorías bien diferenciadas de acuerdo a las directrices del estándar PSS-05-02. Estas dos categorías son:

- **CAP:** Requisitos de capacidad.
- **RES:** Requisitos de restricción.

Cada uno de los requisitos de usuario listados en esta sección está compuesto, además de por el enunciado del requisito propiamente dicho, por una serie de atributos que lo caracterizan. Los distintos atributos, su significado y los valores que admiten se enuncian a continuación:

- **Identificador:** Sirve para referenciar de manera unívoca cada requisito. Los identificadores de los requisitos de usuario obedecen a la siguiente regla:

  UR-<Categoría de Requisito>-<Contador dentro de la categoría>

- **Referencias:** El atributo referencias contiene los identificadores de los requisitos de usuario que estén relacionados con el requisito actual.
- **Prioridad:** Mide el grado de urgencia en la implementación de un determinado requisito, pensando en desarrollos evolutivos. Tiene cuatro posibles valores: *Por defecto, Alta, Media y Baja.*
- **Estabilidad:** La estabilidad de un requisito mide el grado en el que el requisito es susceptible de cambiar a lo largo de la vida del proyecto.

Los posibles valores que puede tomar este atributo son: *Alta, Media y Baja.*

- **Fuente:** Identifica la procedencia del requisito y sus posibles valores son:
  - *UC3M:* El requisito ha sido propuesto por el equipo de UC3M.
  - *M4:* El requisito ha sido propuesto por el equipo de la compañía.
- **Claridad:** Mide la falta de ambigüedad del requisito. Tiene tres posibles valores: *Alta, Media y Baja.*
- **Verificabilidad:** Este atributo mide cómo de fácil es comprobar que el requisito está incorporado al sistema. Sus posibles valores son: *Alta, Media y Baja.*
- **Prueba:** Indica, de manera general, el criterio que se debe seguir para comprobar que el sistema incorpora el requisito.
- **Descripción:** Es un enunciado con detalles que puedan ayudar a la interpretación del requisito de usuario asociado. Este atributo es opcional.

El resto de apartados de esta sección está dedicado a cada una de las categorías de requisitos de usuario identificadas.

3.1 Requisitos de capacidad

UR-CAP-01

El sistema permitirá la conversión de una presentación determinada de la plataforma de desarrollo propiedad de la compañía a otra en WPF desde la plataforma de desarrollo propiedad de la compañía.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-CAP-02

El sistema deberá interpretar una presentación de la plataforma de desarrollo propiedad de la compañía representada en formato OBL.

Referencias: UR-CAP-01.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-CAP-03

El sistema deberá generar una representación en XAML a partir del OBL de la presentación original respetando los datos y las funcionalidades establecidas.

Referencias: UR-CAP-01, UR-CAP-02.
Prioridad: Alta.
Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-CAP-04

El sistema creará el código fuente XAML y su code behind asociado de la presentación a partir de la representación XML.

Referencias: UR-CAP-01, UR-CAP-03.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-CAP-05

El sistema establecerá un entorno .NET para la compilación y presentación del código XAML.

Referencias: UR-CAP-01, UR-CAP-04.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

3.2 Requisitos de restricción

UR-RES-01

Los requisitos de migración de cada uno de los componentes de una presentación están especificados en el documento 'ISDAC_REQ_P01C03_20081020'.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-RES-02

Las recomendaciones para la migración de los componentes de las presentaciones se encuentran especificadas en el documento 'ISDAC_REC_P01C03_20081020'.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-RES-03

Los trabajos de migración realizados deberán poder ser utilizados como parte de futuros procesos y ciclos de migración.

Referencias: UR-RES-01, UR-RES-02.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-RES-04

El proceso de construcción del sistema deberá estar alineado con el estándar de desarrollo de software PSS-05-0 de la ESA.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-RES-05

Todo el desarrollo del sistema seguirá el estándar de codificación Coding Guidelines de la compañía.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-RES-06

El desarrollo de los componentes migrados tenderá a incorporar tanta funcionalidad en código XAML como sea posible, en detrimento del código C#.

Referencias: UR-CAP-04.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-RES-07

El código XAML generado para cada presentación seguirá las recomendaciones del World Wide Web Consortium.

Referencias: UR-CAP-04.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-RES-08

El sistema estará destinado a cualquier usuario de la plataforma de desarrollo propiedad de la compañía.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

UR-RES-09

El proceso de conversión de una presentación deberá ser transparente al usuario.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-RES-10

La conversión de una presentación no requerirá ningún tipo de conocimiento ni de formación específica por parte del usuario.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-RES-11

El entorno de ejecución del sistema será un puesto de usuario o desarrollador de la plataforma de desarrollo propiedad de la compañía equipado con .NET 3.0 o superior.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

UR-RES-12

La comunicación entre el sistema y la plataforma de desarrollo propiedad de la compañía se realizará por medio de los interfaces estándar que ésta última proporciona.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.

Descripción: -.

II. Apéndice B. SSD.

En el presente apéndice se recoge el Documento de Especificación Software (SSD), dónde se incorpora la especificación de la solución software a desarrollar para la habilitación de la migración de presentaciones de plataforma propiedad de la compañía, para ser ejecutadas en un entorno .NET.

Esta tarea de especificación del software consiste en dos tareas básicas: la recopilación de requisitos del software con el de fin de comprender qué requisitos demanda el producto final; y la especificación de la funcionalidad del software a desarrollar.

ISDAC

Integración, Soporte y Difusión de Aplicaciones Complejas

SOFTWARE SPECIFICATION DOCUMENT

Referencia:	ISDAC_AMC.SSD_P01C03_20081020
Versión:	0.02
Creación:	20 10 2008
Última Modificación:	29-03-2009
Preparado por:	Luis Miguel Esteban
Revisado por:	UC3M1

Resumen

El presente documento incorpora la especificación de la solución software a desarrollar para la habilitación de la migración de presentaciones de la plataforma propiedad de la compañía para ser ejecutada en un entorno .NET. Dicha migración comprende la traducción de la presentación de OBL a WPF, y posteriormente la creación.

Hoja de Estado del Documento

1. *ISDAC*. Integración, Soporte y Difusión de Aplicaciones Complejas. Software Specification Document.		
2. Referencia del documento: ISDAC_AMC.SSD_P01C03_20081020		
3. Versión	**4. Fecha**	**5. Motivación del Cambio**
0.01	20-10-2008	Primera versión interna del documento.
0.02	20-01-2009	Incorporación de las características del ciclo

Tabla 1. Hoja de estado del documento.

Registro de Cambios

1. Versión	2. Fecha	3. Autor
0.01	20/10/2008	Luis Miguel Esteban
4. Descripción del Cambio		
• Primera versión interna del documento.		

1. Versión	2. Fecha	3. Autor
0.02	20/01/2009	Luis Miguel Esteban
4. Descripción del Cambio		
• Incorporación de las características del P01C03.		

Índice de Contenidos

Resumen ..... **i**
Hoja de Estado del Documento ..... **i**
Registro de Cambios ..... **ii**
Índice de Contenidos ..... **iii**
1 Introducción **1**
1.1 Propósito del documento 1
1.2 Ámbito del software 1
1.3 Definiciones, acrónimos y abreviaturas 2
1.3.1 Definiciones 2
1.3.2 Acrónimos 2
1.3.3 Abreviaturas 2
1.4 Referencias 2
1.5 Estructura del documento 3
2 Descripción General **4**
2.1 Relaciones con otros proyectos en desarrollo 4
2.2 Función y propósito 4
2.3 Consideraciones de entorno 5
2.4 Relación con otros sistemas 5
2.5 Restricciones generales 5
2.6 Descripción del modelo 6
2.6.1 Obtención del XML de Ejecución 7
2.6.2 Obtención del XAML de Visualización 8
2.6.3 Obtención del XAML Operativo. 8
2.6.4 Invocación de la presentación. 9
3 Requisitos específicos. **10**
3.1 Requisitos funcionales 12
3.2 Requisitos no funcionales 14
3.2.1 Requisitos de rendimiento 14
3.2.2 Requisitos de interfaz 14
3.2.3 Requisitos operacionales 15

3.2.4 Requisitos de recursos 16
3.2.5 Requisitos de verificación 17
3.2.6 Requisitos de prueba de aceptación 17
3.2.7 Requisitos de documentación 17
3.2.8 Requisitos de seguridad 19
3.2.9 Requisitos de portabilidad 19
3.2.10 Requisitos de calidad 20
3.2.11 Requisitos de confiabilidad 21
3.2.12 Requisitos de mantenimiento 21
3.2.13 Requisitos de seguridad operacional 21
4 Descripción de componentes de la solución 22
4.1 M4PresentationTranslator 22
4.2 M4Control 28
4.3 M4HelpLibrary 29
4.4 M4ImagesNet 29
4.5 WPFToolkit 30
4.6 XamlParser 30
5 Matriz de trazabilidad UR-SR. 35

1 Introducción

Este documento es el documento de especificación software del componente para la automatización de la migración de presentaciones de la plataforma propiedad de la compañía, a tecnología .NET.

1.1 Propósito del documento

El propósito del presente documento es ilustrar los requisitos específicos del componente para la migración automática de presentaciones de la plataforma propiedad de la compañía, relativos al Proceso P01 Ciclo C03 del proyecto ISDAC en su anualidad 2008 que nace como heredero del subproyecto KHAMIG dentro del proyecto KHAPYTAL. Adicionalmente, se incluye, como la trazabilidad entre requisitos de usuario, especificados en el documento correspondiente [1] y requisitos de software, en primer lugar, y requisitos de software y componentes, en segundo término. Se pretende que el documento sirva de base a la fase de construcción de la herramienta que se desea realizar como producto principal dentro del ámbito del ciclo mencionado.

1.2 Ámbito del software

El software a desarrollar, tomando como base los requisitos definidos en el presente documento, tiene como objetivo la traducción a WPF de una presentación OBL generada por la plataforma propiedad de la compañía para su lanzamiento en el entorno .NET. El producto obtenido es una solución concebida como un conjunto de proyectos en el entorno .NET que integrará el conjunto de la funcionalidad descrita. Así, el principal beneficio del citado producto es aglutinar la funcionalidad de traducción y muestra de la presentación que se ha tomado como base para el P01C03. Dicha presentación

será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un .EXE ni efectuar almacenamientos en el repositorio de la compañía.

1.3 Definiciones, acrónimos y abreviaturas

En esta sección se recogen las definiciones, acrónimos y abreviaturas que aparecen en este documento.

1.3.1 Definiciones

Requisito Funcional
Requisito que especifica una acción que un sistema ha de ser capaz de realizar, sin tener en cuenta restricciones físicas; un requisitos que especifica el comportamiento de entrada/salida de un sistema

1.3.2 Acrónimos

ESA
European Space Agency

MVS
Microsoft Visual Studio

1.3.3 Abreviaturas

No aplicable.

1.4 Referencias

[1] ISDAC_AMC.URD_P01C03_20081020. *User Requirements Document*. Process 01 Cycle 03.

1.5 Estructura del documento

El presente documento se estructura de la siguiente manera. En primer lugar, la sección 2 incluye la descripción general del sistema en la que se establecerán las relaciones con otros proyectos en desarrollo en el marco de la compañía, el propósito del mismo, las consideraciones del entorno en el que se ejecutará, la relación con otros sistemas existentes, las restricciones con las que se enfrenta y la propia descripción del modelo. En la sección 3 se identifican los requisitos de software con indicación expresa de su tipología: funcionales, de rendimiento, de interfaz, operacionales, de recursos, de verificación, de prueba, de documentación de seguridad, de portabilidad, de calidad, de confiabilidad, de mantenimiento y de seguridad operacional. Posteriormente, en la sección 4 se describe el diseño de la solución que aborda y resuelve la problemática planteada. Por último, la sección 5 incorpora las matrices de trazabilidad entre requisitos de usuario y de software.

2 Descripción General

En esta sección se describen los factores concernientes al componente para la automatización de la migración de presentaciones de la plataforma propiedad de la compañía.

2.1 Relaciones con otros proyectos en desarrollo

El componente de migración automática construido en el P01C03 del proyecto ISDAC en su anualidad 2008 está ampliamente relacionado con el proyecto KHAPITAL y más concretamente al subproyecto KHAMIG. Ambos proyectos se sitúan en el ámbito de la colaboración entre la compañía y la Universidad Carlos III de Madrid. Las soluciones que se han adoptado en el subproyecto KHAMIG han supuesto un marco de actuación para las actividades desarrolladas en ISDAC. Sin embargo, se considera que ISDAC ha supuesto un cambio fundamental en la filosofía de la migración, ya que la antigua filosofía de almacenar el código WPF en el repositorio y construir y compilar un ejecutable se ha visto sustituida por un approach que aboga por la realización de las tareas de forma automática sin interacción con el repositorio de la compañía.

2.2 Función y propósito

Tal y como se ha indicado en la sección 1.2, el propósito del software es automatizar la migración de presentaciones de la plataforma propiedad de la compañía a entorno .NET a través de la utilización de elementos propios de la

compañía y de utilidades y customizaciones de .NET. Este proceso no implica la generación de archivo ejecutable alguno ni el almacenamiento de ningún código en repositorio alguno, permitiendo, de esta manera, que las modificaciones en las presentaciones sean traducidas en el momento de ser necesitadas, eliminando, de esta manera los problemas de consistencia en el almacenamiento persistente entre las presentaciones y la traducción de las mismas a WPF.

2.3 Consideraciones de entorno

El entorno de ejecución es el mismo que se presenta para la ejecución de la plataforma propiedad de la compañía, a la que hay que añadir la presencia de la herramienta de Microsoft .NET en el cliente que ejecuta la migración de la presentación dada. De esta forma, el entorno de ejecución es el cliente con la plataforma propiedad de la compañía al que se añade la presencia de .NET.

2.4 Relación con otros sistemas

El componente se encuentra relacionado con dos sistemas diferenciados. En primer lugar, la plataforma propiedad de la compañía, de la que se recibe el *id_presentacion* e invocando los métodos pertinentes permite conseguir el XML de ejecución que construye de forma interna a partir del OBL. En segundo lugar, se encuentra el entorno .NET, encargado de mostrar la presentación en su entorno.

2.5 Restricciones generales

Las restricciones generales se heredan de las identificadas en el USER REQUIREMENTS DOCUMENT del ciclo, ISDAC_AMC.URD_P01C03_20081020.

2.6 Descripción del modelo

El entorno de ejecución de la solución aportada se ha previsto que se encuentre facilitado por la creación de accesos directos desde la plataforma propiedad de la compañía. Así, dentro del entorno de la plataforma, en la sección de "Herramientas de desarrollo", se crearán una serie de "menús" con tareas asociadas que correspondan a la ejecución del proceso de migración de una presentación determinada. Esta operativa ha sido adoptada de forma interina en espera de la generación de una plataforma estable para el lanzamiento de los procesos de conversión para todas las presentaciones. En la siguiente figura se ilustra la localización interina adoptada en el P01C03 con el propósito de ejecutar la migración de las presentaciones:

Figura 1. Lanzamiento de la ejecución de la presentación migrada

A partir de la ejecución de la tarea comienza todo el proceso de migración en sí. Dicho proceso se encuentra esbozado en la figura siguiente:

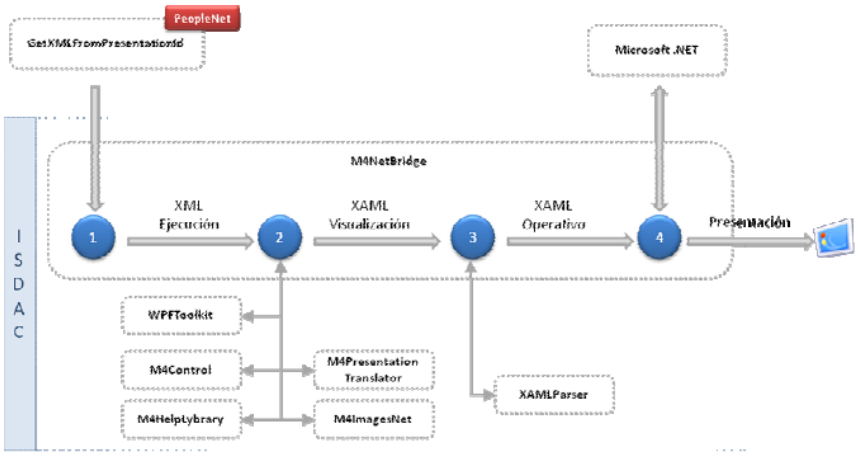


Figura 2. Descripción de la operativa de la solución

A continuación se realiza una descripción detallada de todos los subprocesos llevados a cabo para conseguir que la presentación de la plataforma propiedad de la compañía se ejecute en un entorno .NET.

2.6.1 Obtención del XML de Ejecución

Como primer paso interno a la solución se debe conseguir un XML de Ejecución a partir del OBL de la presentación. El primer paso que se debe dar será validarse en la plataforma propiedad de la compañía con el propósito de poder acceder a la presentación. Para ello se utiliza la clase *M4Service* que permite el registro de la máquina virtual. En segundo lugar, se llama al método

GetXMLFromPresentationId al que se le pasa como parámetro el ID de la presentación que se quiere migrar y la variable de tipo *String* donde se almacenará el XML de Ejecución que se pretende obtener. Este método externo a la solución obtiene a partir de la presentación el código OBL de la misma y lo traduce al XML de ejecución. El XML de ejecución, de forma simplificada, se puede considerar como un XML formateado para los propósitos de la migración.

2.6.2 Obtención del XAML de Visualización

El segundo paso consiste en construir el XAML de visualización a partir del XML de ejecución. Para ello, a partir de la variable que contiene el XML se parsea su contenido con el método *translateToNetPresentation* desarrollado dentro de la solución. Este método realiza los siguientes pasos para construir el XAML de visualización:

- Comprueba cada etiqueta con las clases de migración desarrolladas a tal efecto.
- Ejecuta el método que inicializa las variables propias de la etiqueta analizada.
- Ejecuta el método *Migrate* propio que traduce a XAML dicha etiqueta y las propiedades de la misma.
- Realiza una llamada a la migración de sus hijos y se repite el proceso de migración.
- Finaliza la ejecución cerrando las etiquetas.

2.6.3 Obtención del XAML Operativo.

Teniendo en cuenta que se pretende minimizar el uso de Code Behind, en este paso se realiza la inclusión de capacidades XAML que permitan la ejecución de los eventos ejecutando el Code Behind generado en el paso anterior. Esta funcionalidad se consigue mediante el proyecto XAMLParser y más concretamente el método Parse. Dicho método realiza los siguientes pasos a partir del XAML de visualización para conseguir el XAML operativo:

- Parsea los objetos que tienen asociados eventos.
- Para cada objeto que presenta un evento, incluye código XAML en el mismo con el fin de capacitar la ejecución de los eventos sin necesitar la invocación de code behind (aunque en algunas ocasiones la estructura del evento obligue a que el evento en sí se encuentre codificado mediante code behind).

2.6.4 Invocación de la presentación.

Una vez se ha finalizado todo el proceso de compilación y generación del proyecto XAML operativo que traslada la presentación de la plataforma propiedad de la compañía al mundo .NET, se invocan las librerías de .NET que permiten mostrar la presentación a partir del XAML operativo.

3 Requisitos específicos.

En esta sección se especifican los requisitos software identificados por el equipo de UC3M para el proyecto de automatización de la migración de presentaciones la compañía en su P01 C03.

Los requisitos software se han catalogado en dos categorías bien diferenciadas de acuerdo a las directrices del estándar PSS-05-02. Estas dos categorías son:

FUN: Requisitos funcionales.

NFN: Requisitos no funcionales. Dentro de estos requisitos encontramos:

- Requisitos de rendimiento
- Requisitos de interfaz
- Requisitos operacionales
- Requisitos de recursos
- Requisitos de verificación
- Requisitos de prueba de aceptación
- Requisitos de documentación
- Requisitos de seguridad
- Requisitos de portabilidad
- Requisitos de calidad
- Requisitos de confiabilidad
- Requisitos de mantenimiento
- Requisitos de seguridad operacional

Cada uno de los requisitos software listados en esta sección está compuesto, además de por el enunciado del requisito propiamente dicho, por una serie de atributos que lo caracterizan. Los distintos atributos, su significado y los valores que admiten se enuncian a continuación:

Identificador: Sirve para referenciar de manera unívoca cada requisito. Los identificadores de los requisitos de usuario obedecen a la siguiente regla:

SR-<Categoría de Requisito>-<Contador dentro de la categoría>

Referencias: El atributo referencias contiene los identificadores de los requisitos software que estén relacionados con el requisito actual.

Prioridad: Mide el grado de urgencia en la implementación de un determinado requisito, pensando en desarrollos evolutivos. Tiene cuatro posibles valores: *Por defecto, Alta, Media y Baja.*

Estabilidad: La estabilidad de un requisito mide el grado en el que el requisito es susceptible de cambiar a lo largo de la vida del proyecto. Los posibles valores que puede tomar este atributo son: *Alta, Media y Baja.*

Fuente: Identifica la procedencia del requisito y sus posibles valores son:

- *UC3M:* El requisito ha sido propuesto por el equipo de UC3M.
- *M4:* El requisito ha sido propuesto por el equipo de la compañía.

Claridad: Mide la falta de ambigüedad del requisito. Tiene tres posibles valores: *Alta, Media y Baja.*

Verificabilidad: Este atributo mide cómo de fácil es comprobar que el requisito está incorporado al sistema. Sus posibles valores son: *Alta, Media y Baja.*

Prueba: Indica, de manera general, el criterio que se debe seguir para comprobar que el sistema incorpora el requisito.

Descripción: Es un enunciado con detalles que puedan ayudar a la interpretación del requisito software asociado. Este atributo es opcional.

El resto de apartados de esta sección está dedicado a cada una de las categorías de requisitos de usuario identificadas.

3.1 Requisitos funcionales

SR-FUN-01

El sistema será invocado a través de una tarea de la plataforma propiedad de la compañía.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

SR-FUN-02

El sistema recibirá en su invocación el identificador id_presentación.

Referencias: -.
Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

SR-FUN-03

El sistema convertirá la presentación en OBL a XAML utilizando para el proceso la mayor parte de funcionalidades disponibles en la plataforma propiedad de la compañía para la generación de XML y otros pasos necesarios.

Referencias: .

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

SR-FUN-04

El sistema generará un código XAML correspondiente a la presentación migrada.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

SR-FUN-05

El sistema utilizará las API de .NET para mostrar el XAML de la generado de la presentación.

Referencias: SR-FUN-04.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

3.2 Requisitos no funcionales

3.2.1 Requisitos de rendimiento

No applicable.

3.2.2 Requisitos de interfaz

SR-NFN-01

La comunicación entre el sistema y la plataforma propiedad de la compañía se realizará por medio de los infercates COM que ésta última proporciona.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

3.2.3 Requisitos operacionales

SR-NFN-02

La solución deberá poder servir como base para la migración de componentes de presentaciones diferentes a los identificados en los documentos de requisitos y recomendaciones de migración.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

SR-NFN-03

El sistema estará destinado a todos los usuarios de la plataforma propiedad de la compañía.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

SR-NFN-04

El proceso de conversión de una presentación deberá ser transparente al usuario.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

3.2.4 Requisitos de recursos

SR-NFN-05

El entorno de ejecución del sistema deberá contener la plataforma propiedad de la compañía con .NET 3.0 o superior.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

3.2.5 Requisitos de verificación

No aplicable.

3.2.6 Requisitos de prueba de aceptación

No aplicable.

3.2.7 Requisitos de documentación

SR-NFN-06

Los requisitos de migración de cada uno de los componentes de una presentación están especificados en el documento 'ISDAC_REQ_P01C03_20081020'.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

SR-NFN-07

Las recomendaciones para la migración de los componentes de las presentaciones se encuentran especificadas en el documento 'ISDAC_REC_P01C03_20081020'.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

SR-NFN-08

El proceso de construcción del sistema deberá estar alineado con el estándar de desarrollo de software PSS-05-0 de la ESA.

Referencias: -.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

SR-NFN-09

Todo el desarrollo del sistema seguirá el estándar de codificación Coding Guidelines de la compañía.

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

SR-NFN-10

La producción de la documentación será gobernada por las directivas presentes en los documentos Migration Plan Framework (KHAPYTAL_MPF) y Migration Plan Definition (KHAPYTAL_MPD).

Referencias: -.

Prioridad: Alta.

Estabilidad: Alta.

Fuente: UC3M.

Claridad: Alta.

Verificabilidad: Alta.

Prueba: -.

Descripción: -.

3.2.8 Requisitos de seguridad

No aplicable.

3.2.9 Requisitos de portabilidad

No aplicable.

3.2.10 Requisitos de calidad

SR-NFN-11

El desarrollo de los componentes migrados tenderá a incorporar tanta funcionalidad en código XAML como sea posible, en detrimento del código C#.

Referencias: SR-FUN-04, SR-FUN-07.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

SR-NFN-12

El código XAML generado para cada presentación seguirá las recomendaciones del World Wide Web Consortium.

Referencias: SR-FUN-04.
Prioridad: Alta.
Estabilidad: Alta.
Fuente: UC3M.
Claridad: Alta.
Verificabilidad: Alta.
Prueba: -.
Descripción: -.

3.2.11 Requisitos de confiabilidad

No aplicable.

3.2.12 Requisitos de mantenimiento

No aplicable.

3.2.13 Requisitos de seguridad operacional

No aplicable.

4 Descripción de componentes de la solución

En la presente sección se describen los componentes del sistema de migración. Se quiere poner de manifiesto que los componentes identificados no corresponden a componentes software en su visión más académica, sino a elementos funcionales dentro de la solución desarrollada.

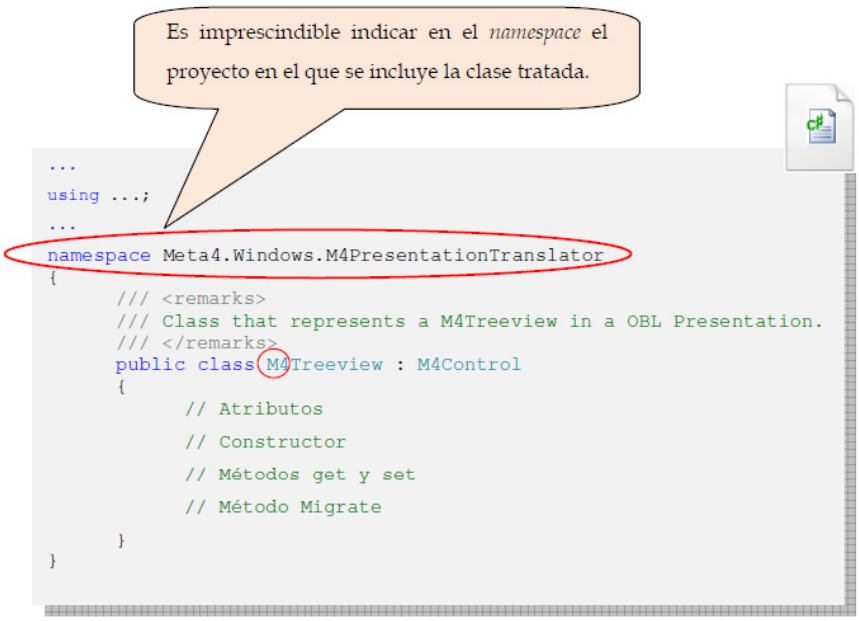
En el ciclo 03, en el cual nos encontramos, dichos componentes son completamente imprescindibles. Pero cuando el proyecto se encuentre inmerso en el siguiente ciclo, se estudiará la simplificación de estos, si fuese posible.

4.1 M4PresentationTranslator

Este componente es el encargado de realizar la traducción del código OBL de una presentación de la plataforma propiedad de la compañía, o mejor dicho, la traducción del código XML de ejecución correspondiente al código OBL de una presentación de la plataforma propiedad de la compañía, a su equivalente en XAML.

El proceso de migración consiste en leer un control del código XML de ejecución de la presentación a migrar; a dicho control se le añadirá el prefijo "M4", ya que todas las clases de este proyecto que se encargan de la migración de los controles, están bautizadas como: "M4" + <nombre_control>. Y en cada una de estas clases se procede a controlar todas las propiedades posibles a disponer para dicho control, así como su método propio de migración.

A continuación se muestra un ejemplo sencillo de una de estas clases correspondientes a un control con correspondencia directa en XAML, por ejemplo, la clase del control *Treeview*:



El método Migrate sería como sigue. Se incluirán ciertos cuadros de texto numerados para posteriormente hacer mención a ellos y explicarlos:

```
public override void Migrate(M4Control oParentControl, XmlNode
                             oM4ParentNode, M4ContextData oM4Context)
{
    XmlNode oM4ChildNode;
    MethodInfo myMethod;
    XmlDocument oM4XAMLWriteDocument = oM4ParentNode.OwnerDocument;

    /* El control TreeView muestra una jerarquía de nodos
     * similar al modo en que se muestran los archivos y las
     * carpetas en el panel izquierdo de la característica
     * Explorador de Windows de sistemas operativos Windows.
     */
    oM4ChildNode = oM4XAMLWriteDocument.CreateElement("", "TreeView",
        "http://schemas.microsoft.com/winfx/2006/xaml/presentation");

    // añadir el nodo creado al nodo padre
    oM4ParentNode.AppendChild(oM4ChildNode);

    // se añaden todos los atributos con sus respectivos valores
    PropertyInfo[] oM4PropertiesInfo = this.GetType().GetProperties();
    foreach (PropertyInfo oM4PropertyInfo in oM4PropertiesInfo)
    {
        if (oM4PropertyInfo.Name != "M4Controls")
        {
            /* Recuperar el método que migra dicha propiedad
             * (específico, o general en ausencia de este)
             */
            myMethod = (new M4MigrateMethods()).GetType().GetMethod(
                        "Migrate" + oM4PropertyInfo.Name,
                        new Type[] { typeof(M4Control),
                        typeof(M4Treeview), typeof(XmlNode),
                        typeof(PropertyInfo), typeof(M4ContextData)});
            // Invocar al método recuperado
            try
            {
                myMethod.Invoke(this, new object[] { oParentControl,
                    this, oM4ChildNode, oM4PropertyInfo, oM4Context });
            }
            catch (NullReferenceException)
            {
                string texto = "Error: la propiedad " +
                        oM4PropertyInfo.Name + " no pudo ser migrada ";
                M4Utils.M4AddToLog(texto);
            }
        }
    }

    // Invocar a la migración de cada uno de los controles contenidos
    foreach (M4Control oM4Control in this.M4Controls)
    {
        oM4Control.Migrate(this, oM4ChildNode, oM4Context);
    }
}
```

1
2
3
4

1	A cada control a migrar se le indica el contexto en el que se encuentra dentro del código, ya que la migración puede variar en función de dicho contexto
2	Para migrar un control del código XML de ejecución, almacenamos en un fichero la correspondencia de este control con su homólogo en XAML.
3	Dicho control dispone de ciertas propiedades, por lo que en su correspondencia en XAML también deberán aparecer migradas.
4	Para continuar el proceso, será necesario repetir estos pasos pero con los hijos contenidos en el control tratado.

En el ciclo 03 de este proyecto se ha conseguido realizar la migración de los siguientes controles (teniendo presente que la mayoría de estos controles aún no están completamente finalizados):

- M4Action_dataprops.cs
- M4Action_value.cs
- M4Button.cs
- M4Changer.cs
- M4Control.cs
- M4Evclick.cs
- M4Form.cs
- M4Groupbox.cs
- M4Image.cs
- M4Includemenubar.cs
- M4Includepanel.cs

- M4Itemlabel.cs
- M4Label.cs
- M4Menu.cs
- M4Menuitem.cs
- M4Menuparamsgroup.cs
- M4Mimic.cs
- M4Nodestatus.cs
- M4Panel.cs
- M4Patternbox.cs
- M4Presentation.cs
- M4ScrollPanel.cs
- M4Splittblock.cs
- M4Splitthorizontal.cs
- M4Splittvertical.cs
- M4Tab.cs
- M4Table.cs
- M4Tabstrip.cs
- M4Textarea.cs
- M4Textbox.cs

- M4Toolbar.cs
- M4ToolbarButton.cs
- M4ToolbarDataGroup.cs
- M4ToolbarEditGroup.cs
- M4Toolbarparamsgroup.cs
- M4ToolbarRootNavigationGroup.cs
- M4Toolbarseparator.cs
- M4TraductProp.cs
- M4Tree.cs
- M4Treenode.cs
- M4Treeview.cs

Además, para el correcto funcionamiento de la migración, serán necesarias las siguientes clases:

- M4Constants.cs → clase que contiene todos los atributos con sus valores por defecto.
- M4ContextData.cs → clase padre que contiene las variables de contexto de cada control.
- M4MigrateMethods.cs → clase que contiene los métodos de migración de todos los atributos.
- M4Parser.cs → clase con la que se comienza el proceso de migración.

- M4Utils.cs → clase que contiene funcionalidades complementarias que no están implicadas directamente en la migración de controles.

4.2 M4Control

Este componente es el encargado de dos tareas principales:

- Aportar la funcionalidad concreta a cada acción de cada evento.
- Creación de los controles personalizados de determinados objetos recibidos en el código XML de ejecución.

El motivo de haber personalizado ciertos controles es debido a que se tratan de controles con propiedades especiales asociadas, como la gestión de eventos, el acceso a base de datos, cálculos matemáticos en tiempo de ejecución.

Existen ciertas clases especiales para la gestión de eventos. Estas son

- IM4Action.cs
- IM4EventDispatcher.cs
- M4EventHandler.cs
- M4EventParams.cs
- M4EventParamsList.cs

Gracias a ellas se podrá generar en XAML una disposición arbórea de tal manera que se permita ofrecer la posibilidad de que para un mismo control, se disparen varios eventos. Para ello se utiliza el control *M4EventHandler*, que representa todos los eventos, y una lista de estos eventos representada por el control *M4EventParamsList*. A su vez, para cada evento es posible disponer de

una lista de acciones, representada por el control *M4EventParams*. Para ejecutar dichas acciones relativas a un evento, serán utilizados los controles propios de acciones como son *M4Action_dataprops* o *M4Action_value*, que ejecutarán una acción diferente en función al valor de la propiedad *Action*, *Dialog*, etc.

4.3 M4HelpLibrary

Este componente es el encargado de aportar la funcionalidad de ayuda necesaria, usada en el control de ayuda F1.

4.4 M4ImagesNet

Este componente es el encargado de aportar las imágenes necesarias en la presentación que se esté migrando. Dicho componente estará formado por una serie de directorios entre los que se encuentra *ResActiveDB*. Dicho directorio será el encargado de contener los ficheros de imagen que serán utilizados para la correcta visualización de la presentación migrada.

Si se desea visualizar una nueva imagen que esté contenida en la presentación pero que no se encuentre almacenada en el directorio *ResActiveDB*, será necesario añadirla. El proceso para añadir una imagen es el siguiente:

- Colocar la nueva imagen en el directorio *M4ImagesNet\ResActiveDB*.
- Desde el entorno MVS, sobre el directorio *ResActiveDB* elegir la opción "Add Existing Item". Elegir la imagen añadida anteriormente a dicho directorio y elegir la opción "Add".

- Arrastrar la nueva imagen al fichero *M4ActiveDashboard.resx*, contenido en el proyecto.
- Hacer *build* del proyecto.

4.5 WPFToolkit

Este componente es el encargado de contener las clases e iconos/imágenes necesarios para que se pueda mostrar la función de calendario en aquellas presentaciones que lo contengan.

4.6 XamlParser

Este componente es el encargado de hacer posible la inclusión de capacidades XAML que permitan la ejecución de los eventos. Para conseguir ejecutar dichos eventos será necesario parsear los objetos contenidos en el XAML de visualización que tienen asociados eventos, y para cada objeto que presenta un evento, incluir código XAML en el mismo con el fin de capacitar la ejecución de los eventos.

A continuación se muestra un ejemplo en el que se explica paso a paso con más detalle lo anteriormente mencionado.

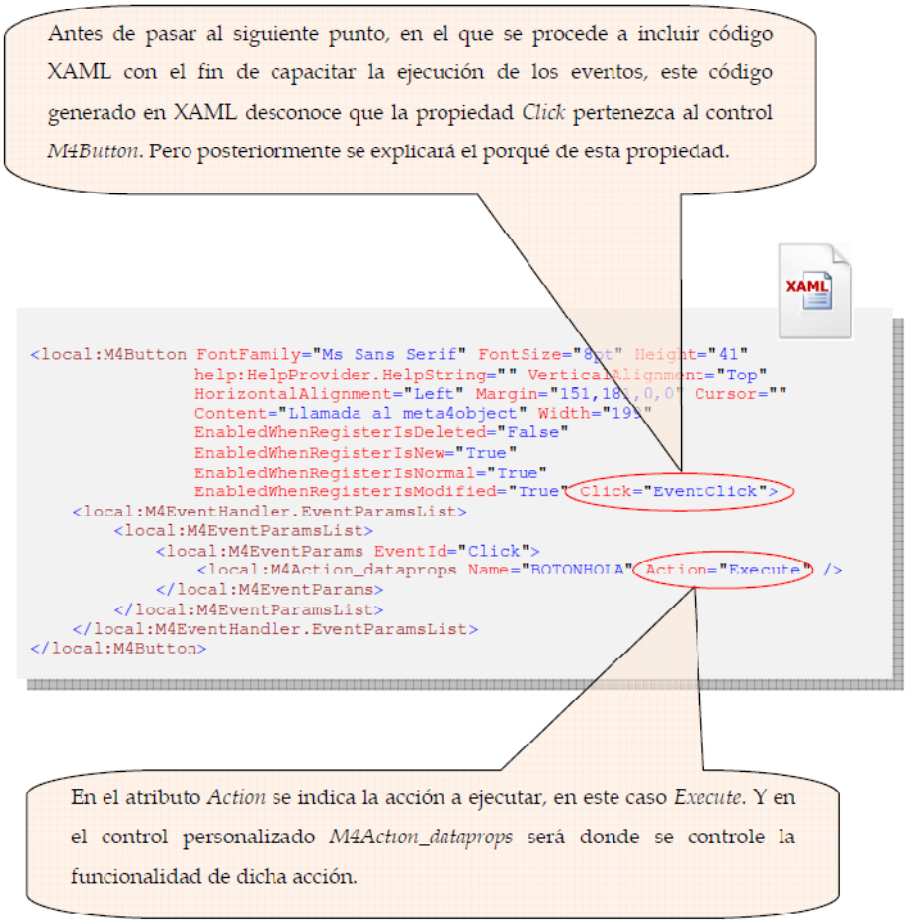
1. Cuando obtenemos el código XML de ejecución correspondiente al OBL de la presentación, le exponemos a un proceso de migración con el cual obtenemos el código XAML de Visualización. Este es un ejemplo en el que se dispone de un botón con un evento asociado:


```
<Button ALIAS ="Button1">
      <Grants>31</Grants>
      <Left>151</Left>
      <Top>181</Top>
      <Width>199</Width>
      <Height>41</Height>
      <Text>Llamada al meta4object</Text>
      <Evclick ALIAS ="Evclick1">
            <Grants>31</Grants>
            <Action_dataprops ALIAS ="Action_dataprops1">
                  <Grants>23</Grants>
                  <Iditem>BOTONHOLA</Iditem>
            </Action_dataprops>
      </Evclick>
</Button>
```

El código anterior consiste en un pedazo de código XML de ejecución el cual tiene una correspondencia directa con el código OBL recibido de la presentación. Se trata de un botón que tiene asociado un evento *Evclick*, el cual tiene asociada una acción *Action_dataprops*.

2. A continuación se procede a mostrar el código XAML de Visualización resultante tras el proyecto de migración. En él se puede comprobar cómo la migración del control *Button* se corresponde con un control personalizado *M4Button*. Además, se puede comprobar como todas las propiedades de dicho botón también han sido migradas.

   El evento *Evclick* ha sido migrado con la estructura arbórea explicada en el punto **4.2 M4Control**; en este caso con un solo evento para el control *Button*, y una sola acción asociada a dicho evento.



3. Por último, se muestra el pedazo de código correspondiente al XAML operativo, conseguido tras la inclusión del código necesario para controlar los eventos en al XAML de visualización anterior, aportando la posibilidad de ejecutar la funcionalidad requerida por el botón.

Los controles con eventos asociados tendrán un identificador de evento de nombre *Click*. Además, el nombre de la función que gestiona los eventos en cada control será *EventClick*. Es por ello por lo que anteriormente se disponía de la propiedad *Click* en el control *M4Button*. De esta manera lo que se consigue es bindar la propiedad *Click* de la clase *M4Button*, con el método *EventClick* de la **instancia** de la clase.

XAML

```
<local:M4Button FontFamily="Ms Sans Serif" FontSize="8pt" Height="41"
                help:HelpProvider.HelpString="" VerticalAlignment="Top"
                HorizontalAlignment="Left" Margin="151,181,0,0" Cursor=""
                Content="Llamada al metaobject" Width="149"
                EnabledWhenRegisterIsDeleted="False"
                EnabledWhenRegisterIsNew="True"
                EnabledWhenRegisterIsNormal="True"
                EnabledWhenRegisterIsModified="True">
    <XamlParser:ElementBinder.ElementList>
        <XamlParser:ElementList>
            <XamlParser:Event EventName="Click" EventHandler="EventClick" />
        </XamlParser:ElementList>
    </XamlParser:ElementBinder.ElementList>
    <local:M4EventHandler.EventParamsList>
        <local:M4EventParamsList>
            <local:M4EventParams EventId="Click" >
                <local:M4Action_dataprops Name="BOTONHOLA" Action="Execute" />
            </local:M4EventParams>
        </local:M4EventParamsList>
    </local:M4EventHandler.EventParamsList>
</local:M4Button>
```

En este caso, el identificador del evento tiene como nombre *Click*.

¿Qué aporta dicho proyecto XamlParser?

En los dos primeros ciclos del proyecto, la idea llevada a cabo consistía en generar y almacenar un archivo XAML, el cual sería ejecutado mediante un ejecutable generado por el propio proyecto. Todo esto quedaría empaquetado y grabado en el repositorio la compañía.

Actualmente, la línea de desarrollo a seguir consiste en generar un XAML en cliente justo antes de ser ejecutada la presentación, es decir: generar el XAML, ejecutarlo y el cliente, sin necesidad de almacenar dicho XAML, será capaz de visualizar directamente la presentación en WPF (.NET). En este sentido, el código generado en XAML deberá ser capaz de aportar toda la funcionalidad requerida por el cliente, por lo que la gestión de eventos debe ser controlada de algún modo en dicho XAML. Este modo de gestionar los eventos consiste en incorporar un código XAML para cada control con un evento asociado, que se encargue de realizar los correspondientes bindados a las acciones a ejecutar de cada evento.

5 Matriz de trazabilidad UR-SR.

MATRIZ DE TRAZABILIDAD DE REQUISITOS SR—UR		Fecha: 23/01/2009
PROYECTO: *ISDAC*		
URD ID.	SRD ID.	RESUMEN DEL RU
UR-CAP-01	SR-FUN-01 SR-FUN-02 SR-FUN-03	El sistema permitirá la conversión de una presentación determinada de la plataforma propiedad de la compañía a otra en WPF desde la plataforma propiedad de la compañía.
UR-CAP-02	SR-FUN-02 SR-FUN-03	El sistema deberá interpretar una presentación de la plataforma propiedad de la compañía representada en formato OBL
UR-CAP-03	SR-FUN-03 SR-FUN-04	El sistema deberá generar una representación en XAML a partir del OBL de la presentación original respetando los datos y las funcionalidades establecidas
UR-CAP-04	SR-FUN-03 SR-FUN-04 SR-NFN-11	El sistema creará el código fuente XAML y su code behind asociado de la presentación a partir de la representación XML.
UR-CAP-05	SR-FUN-05 SR-NFN-05	El sistema establecerá un entorno .NET para la compilación y presentación del código XAML.
UR-RES-01	SR-NFN-06	Los requisitos de migración de cada uno de los componentes de una presentación están especificados en el documento 'ISDAC_REQ_P01C03_20081020'
UR-RES-02	SR-NFN-07	Las recomendaciones para la migración de los componentes de las presentaciones se encuentran especificadas en el documento 'ISDAC_REC_P01C03_20081020'.
UR-RES-03	SR-NFN-10 SR-NFN-02	Los trabajos de migración realizados deberán poder ser utilizados como parte de futuros procesos y ciclos de migración.
UR-RES-04	SR-NFN-08	El proceso de construcción del sistema deberá estar alineado con el estándar de desarrollo de software PSS-05-0 de la ESA.
UR-RES-05	SR-NFN-09	Todo el desarrollo del sistema seguirá el estándar de codificación Coding Guidelines de la compañía.
UR-RES-06	SR-NFN-11	El desarrollo de los componentes migrados tenderá a incorporar tanta funcionalidad en código XAML como

		sea posible, en detrimento del código C#.
UR-RES-07	SR-NFN-12	El código XAML generado para cada presentación seguirá las recomendaciones del World Wide Web Consortium.
UR-RES-08	SR-FUN-01 SR-NFN-03 SR-NFN-04	El sistema estará destinado a cualquier usuario de la plataforma propiedad de la compañía.
UR-RES-09	SR-FUN-01 SR-NFN-03 SR-NFN-04	El proceso de conversión de una presentación deberá ser transparente al usuario
UR-RES-10	SR-FUN-01 SR-NFN-03 SR-NFN-04	La conversión de una presentación no requerirá ningún tipo de conocimiento ni de formación específica por parte del usuario
UR-RES-11	SR-NFN-05	El entorno de ejecución del sistema será un puesto de usuario o desarrollador de la plataforma propiedad de la compañía equipado con .NET 3.0 o superior
UR-RES-12	SR-FUN-03 SR-NFN-01	La comunicación entre el sistema y la plataforma propiedad de la compañía se realizará por medio de los interfaces estándar que ésta última proporciona.

III. Apéndice C. Plan de administración del proyecto software.

En el presente apéndice se recoge el Plan de Administración del Proyecto Software (PAPS), dónde se describe la técnica de administración del proyecto de desarrollo del plan de migración que permite traducir interfaces de usuario creadas con un código obsoleto, a interfaces de usuario con las que interactuar en un entorno .NET.

Esta tarea de administración consiste en recoger la organización del proyecto en cuanto a roles y límites organizacionales se refiere, así como el proceso técnico en el que se detallarán las entradas y salidas del proyecto, y las herramientas utilizadas para tal fin. A su vez, se mostrará el calendario y el presupuesto previsto para este proyecto.

ISDAC

Integración, Soporte y Difusión de Aplicaciones Complejas

PLAN DE ADMINISTRACIÓN DEL PROYECTO SOFTWARE

Versión:	0.01
Creación:	20-10-2008
Última Modificación:	30-03-2009
Preparado por:	Luis Miguel Esteban
Revisado por:	UC3M1

Resumen

El presente documento consiste en el Plan de Administración del Proyecto Software del proceso de reingeniería del software del proyecto KHAMIG.

Hoja de Estado del Documento

1. *ISDAC.* Integración, Soporte y Difusión de Aplicaciones Complejas. Software Specification Document.		
2. Referencia del documento: PAPS		
3. Versión	**4. Fecha**	**5. Motivación del Cambio**
0.01	20-10-2008	Primera versión interna del documento.

Tabla 1. Hoja de estado del documento.

Registro de Cambios

1. Versión	2. Fecha	3. Autor
0.01	20/10/2008	Luis Miguel Esteban
4. Descripción del Cambio		
• Primera versión interna del documento.		

Índice de Contenidos

Resumen ..... **i**
Hoja de Estado del Documento ..... **i**
Registro de Cambios ..... **ii**
Índice de Contenidos ..... **iii**
Índice de ilustraciones ..... **iv**
Índice de tablas ..... **v**
1 Introducción **1**
1.1 Propósito del documento 1
1.2 Ámbito del software 1
1.3 Definiciones, acrónimos y abreviaturas 2
1.3.1 Definiciones 2
1.3.2 Acrónimos 2
1.3.3 Abreviaturas 3
1.4 Referencias 3
1.5 Estructura del documento 3
2 Organización del proyecto **4**
2.1 Roles organizativos y responsabilidades 4
2.2 Límites organizacionales e interfaces 4
3 Proceso técnico. **5**
3.1 Entradas del proyecto. 5
3.2 Salidas del proyecto. 5
3.3 Tecnología y herramientas de desarrollo. 6
3.4 Funciones de apoyo al proceso. 7
4 Calendario y presupuesto **8**
4.1 Calendario. 8
4.1.1 Planificación. 10
4.1.2 Diagrama de Gantt. 13
4.1.3 Consideraciones. 16
4.2 Presupuesto. 17

Índice de ilustraciones

Ilustración 1 - Planificación (I) 10
Ilustración 2 - Planificación (II) 11
Ilustración 3 - Planificación (III) 12
Ilustración 4 -Planificación (IV) 13
Ilustración 5 - Diagrama de Gantt 13
Ilustración 6 - Diagrama de Gantt (I) 14
Ilustración 7 - Diagrama de Gantt (II) 14
Ilustración 8 - Diagrama de Gantt (III) 15
Ilustración 9 - Diagrama de Gantt (IV) 15

Índice de tablas

Tabla 1 - Coste temporal..... 17
Tabla 2 - Coste monetario..... 18

1 Introducción

El presente documento describe el *Plan de Administración* del proyecto de desarrollo de un plan de migración que permita traducir interfaces de usuario creadas con un código obsoleto, a interfaces de usuario con las que interactuar en un entorno .NET.

Esta sección aborda los objetivos tanto del plan de migración como de este documento en sí mismo. También recoge las definiciones y las referencias que aparecen en el resto del documento y una breve descripción del contenido de las secciones que lo conforman.

1.1 Propósito del documento

El *Plan de Administración del Proyecto Software* define los objetivos, los entregables y el ciclo de vida del proyecto de migración. También enuncia las principales actividades que marcan el desarrollo del proyecto, así como los recursos necesarios, su calendario y su presupuesto. Su redacción está basada en la aplicación de los estándares de Ingeniería de Software de la ESA [1] y [2].

1.2 Ámbito del software

El software a desarrollar, tomando como base los requisitos definidos en el presente documento, tiene como objetivo la traducción a WPF de una presentación OBL generada por la plataforma propiedad de la compañía para su lanzamiento en el entorno .NET. El producto obtenido es una solución concebida como un conjunto de proyectos en el entorno .NET que integrará el conjunto de la funcionalidad descrita. Así, el principal beneficio del citado producto es aglutinar la funcionalidad de traducción y muestra de la

presentación que se ha tomado como base para el P01C03. Dicha presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un .EXE ni efectuar almacenamientos en el repositorio de la compañía.

1.3 Definiciones, acrónimos y abreviaturas

En esta sección se recogen las definiciones, acrónimos y abreviaturas que aparecen en este documento.

1.3.1 Definiciones

Diseñador de presentaciones
Herramienta disponible en la plataforma propiedad de la compañía. Su fin consiste en elaborar presentaciones, bien gráficamente, bien desarrollando código manualmente.

Presentaciones
Interfaces de usuario creadas con un propósito, por ejemplo, un mantenimiento de empleados.

Controles / Objetos
Elementos que aparecen en el código referente a las presentaciones con el fin de representar un componente de la presentación.

1.3.2 Acrónimos

ESA
European Space Agency

MVS
Microsoft Visual Studio

1.3.3 Abreviaturas

No aplicable.

1.4 Referencias

[1] BSSC (96)2: Guide to applying the ESA software engineering standards to small software projects.

Disponible en: http://styx.esrin.esa.it/premfire/Docs/Bssc962.pdf

[2] PSS-05-05: Guide to the user requirements definition phase.

Disponible en: http://styx.esrin.esa.it/premfire/Docs/PSS0505.pdf

1.5 Estructura del documento

El presente documento se estructura de la siguiente manera. En primer lugar, la sección dos incluye la organización del proyecto en cuanto a roles y límites organizacionales se refiere. La sección tres consiste en el proceso técnico en el que se detallarán las entradas y salidas del proyecto, así como las herramientas utilizadas para tal fin. Por último, la sección cuatro mostrará el calendario y el presupuesto previsto para este proyecto.

2 Organización del proyecto

Esta sección define la estructura de gestión del proyecto, junto a las responsabilidades de cada uno de sus miembros.

2.1 Roles organizativos y responsabilidades

Debido al carácter académico del proyecto, como Proyecto Fin de Carrera de Ingeniería en Informática, todos los roles necesarios para su realización serán llevados a cabo por su autor, Luis Miguel Esteban Santiago.

2.2 Límites organizacionales e interfaces

El componente de migración automática construido en el P01C03 del proyecto ISDAC en su anualidad 2008 está ampliamente relacionado con el proyecto KHAPITAL y más concretamente al subproyecto KHAMIG. Ambos proyectos se sitúan en el ámbito de la colaboración entre la compañía y la Universidad Carlos III de Madrid. Las soluciones que se han adoptado en el subproyecto KHAMIG han supuesto un marco de actuación para las actividades desarrolladas en ISDAC. Sin embargo, se considera que ISDAC ha supuesto un cambio fundamental en la filosofía de la migración, ya que la antigua filosofía de almacenar el código WPF en el repositorio y construir y compilar un ejecutable se ha visto sustituida por un approach que aboga por la realización de las tareas de forma automática sin interacción con el repositorio de la compañía.

3 Proceso técnico.

En esta sección se establecen los detalles técnicos de la gestión del proyecto, entre los que se encuentran las entradas y las salidas del proyecto, así como las herramientas de desarrollo utilizadas para la consecución del mismo.

3.1 Entradas del proyecto.

La principal entrada al proyecto ha sido, por supuesto, el plan de migración definido conseguir la presentación migrada a tecnología .NET desde la presentación definida en código obsoleto.

A su vez, ha sido esencialmente necesario seguir un framework previamente definido en el anterior ciclo del proyecto (*Cabezas, PFC, Junio 2008*).

3.2 Salidas del proyecto.

La principal salida de este proyecto ha sido la herramienta migradora, la cual consta de los siguientes componentes:

- M4PNetBridge.
- TestInitAppWpf.
- M4HelpLibrary.
- M4ImagesNet.
- WPFToolkit.

- M4PresentationTranslator.
- M4DynamicControl.
- XAMLParser.

Además, estos son los documentos obtenidos en el proceso de migración:

- Documento de requisitos de usuario.
- Documento de especificación software.
- Plan de administración del proyecto software.

3.3 Tecnología y herramientas de desarrollo.

En cuanto a la tecnología empleada para la consecución del proyecto de reingeniería, ha sido necesario disponer del entorno Microsoft .NET Framework, y del sistema operativo Windows.

Las herramientas empleadas han sido:

- Microsoft Visual Studio 2008.
- Microsoft Visual SourceSafe 8.0.
- Paquete Microsoft Office 2007.
- Sistema de información desarrollado por el cliente.

3.4 Funciones de apoyo al proceso.

Teniendo en cuenta el tamaño del proyecto y el carácter académico del mismo, no se han definido formalmente funciones de gestión de la configuración, verificación y validación y aseguramiento de la calidad.

4 Calendario y presupuesto

Esta sección presenta la descomposición de las tareas del proyecto, las cuales serán mostradas en el calendario que marca la evolución de las mismas. A su vez, se calcula el presupuesto para la ejecución del proyecto que nos compete.

4.1 Calendario.

Todo proyecto con el que se pretenda llegar a un resultado idóneo, dispone de una planificación precisa y adecuada que debe ser llevada a cabo. Por lo tanto, realizar una buena planificación es una de las tareas iniciales más importantes que deben tenerse en cuenta. Lógicamente, la planificación puede cambiar durante el ciclo de vida del proyecto, por lo que no es una tarea estática, sino que con el paso del tiempo puede modificarse, siempre que dichas alteraciones sean apropiadas y permisibles.

En el caso del presente Proyecto Fin de Carrera, la planificación realizada inicialmente fue relativamente buena y no fue necesario modificarla en exceso, pero al tratarse de una migración de tecnología, muchas ocasiones exigían un mayor detenimiento en el análisis de ciertos objetos a migrar. Mientras que otras tareas para las que se estimaron un tiempo bastante amplio, no fue necesario hacer uso de todo ese tiempo y se consiguió finalizar el trabajo antes de tiempo. Gracias a poder haber seguido a tiempo dicha planificación, el resultado final consiguió ser el deseado.

Las ocupaciones con las que cuenta el proyecto son denominadas *tareas*:

- **La primera tarea** consiste en el estudio de la documentación del ciclo anterior, con el propósito de familiarizarme con el proyecto de reingeniería a realizar y entender a su vez el resultado que se pretendía obtener.

- **La segunda tarea** consiste en la integración del proyecto migrador con la aplicación propiedad de la empresa, que cuenta con la herramienta *Diseñador de presentaciones* que se requiere migrar.

- **La tercera tarea** se basa en el análisis de los componentes incluidos en la presentación objetivo que se pretende migrar, para saber el ámbito del proyecto y ser capaces de crear una planificación adecuada.

- **La cuarta etapa** es la más amplia, ya que necesita más del 85% del tiempo total del proyecto para su consecución. Lógicamente, esta etapa consiste en la migración de los objetos desde la tecnología propiedad de la empresa, a tecnología .NET, así como su correspondiente documentación.

A continuación se muestra la planificación de dichas tareas de las que consta el proyecto de migración, donde se observarán los nombres de las tareas y la duración de las mismas, así como su fecha de comienzo y fin. Si alguna de estas tareas necesita la anterior finalización de una de las tareas previas, dicha tarea prioritaria aparecerá como predecesora. Posteriormente se mostrará el correspondiente Diagrama de Gantt de dicha planificación.

4.1.1 Planificación.

		Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	✓	**Proyecto KHAMIG (Ciclo 3)**	**110 días**	**lun 29/09/08**	**vie 27/02/09**	
2	✓	Estudio documentación	5 días	lun 29/09/08	vie 03/10/08	
3	✓	**Integración con la aplicación**	**7 días**	**lun 06/10/08**	**mar 14/10/08**	2
4	✓	Análisis del ámbito del proyecto	3 días	lun 06/10/08	mié 08/10/08	
5	✓	Modificar M4SysTask	1 día	jue 09/10/08	jue 09/10/08	
6	✓	Crear M4NetBridge.dll con interfaz .COM	2 días	vie 10/10/08	lun 13/10/08	
7	✓	Convertir M4Translator a .dll	1 día	mar 14/10/08	mar 14/10/08	
8	✓	Pruebas	1 día	mar 14/10/08	mar 14/10/08	
9	✓	**Analisis presentación UC3M_FRAMEWORK_I**	**3 días**	**mié 15/10/08**	**vie 17/10/08**	
10	✓	Análisis de componentes incluidos en la presentación a migrar	3 días	mié 15/10/08	vie 17/10/08	
11	✓	**MIGRACIÓN OBJETOS**	**95 días**	**lun 20/10/08**	**vie 27/02/09**	3
12	✓	Documentos generales del proyecto (Scope, URD, SSD, etc.)	40 días	lun 05/01/09	vie 27/02/09	
13	✓	**Auxiliares**	**83 días**	**lun 20/10/08**	**mié 11/02/09**	
14	✓	TraductProp	2 días	lun 20/10/08	mar 21/10/08	
15	✓	Control	1 día	lun 20/10/08	lun 20/10/08	
16	✓	Constants	1 día	mar 21/10/08	mar 21/10/08	
17	✓	ContextData	1 día	mar 21/10/08	mar 21/10/08	
18	✓	Parser	2 días	mié 22/10/08	jue 23/10/08	
19	✓	Utils	2 días	jue 23/10/08	vie 24/10/08	
20	✓	Translator	1 día	vie 24/10/08	vie 24/10/08	
21	✓	MigrateMethods	78 días	lun 27/10/08	mié 11/02/09	
22	✓	**Presentation**	**1 día**	**lun 27/10/08**	**lun 27/10/08**	
23	✓	Análisis y documentación	1 día	lun 27/10/08	lun 27/10/08	
24	✓	Migración	1 día	lun 27/10/08	lun 27/10/08	
25	✓	**Form**	**3 días**	**lun 27/10/08**	**mié 29/10/08**	
26	✓	Análisis y documentación	1 día	lun 27/10/08	lun 27/10/08	
27	✓	Migración	3 días	lun 27/10/08	mié 29/10/08	
28	✓	**Button**	**4 días**	**mar 28/10/08**	**vie 31/10/08**	
29	✓	Análisis y documentación	3 días	mar 28/10/08	jue 30/10/08	
30	✓	Migración	3 días	mié 29/10/08	vie 31/10/08	

Ilustración 1 - Planificación (I)

31	✓	**Textbox**	**4 días**	**lun 03/11/08**	**jue 06/11/08**	
32	✓	Análisis y documentación	1 día	lun 03/11/08	lun 03/11/08	
33	✓	Migración	4 días	lun 03/11/08	jue 06/11/08	
34	✓	**Textarea**	**2 días**	**mar 04/11/08**	**mié 05/11/08**	
35	✓	Análisis y documentación	1 día	mar 04/11/08	mar 04/11/08	
36	✓	Migración	2 días	mar 04/11/08	mié 05/11/08	
37	✓	**Groupbox**	**2 días**	**jue 06/11/08**	**vie 07/11/08**	
38	✓	Análisis y documentación	1 día	jue 06/11/08	jue 06/11/08	
39	✓	Migración	2 días	jue 06/11/08	vie 07/11/08	
40	✓	**Includepanel y Panel**	**3 días**	**lun 10/11/08**	**mié 12/11/08**	
41	✓	Análisis y documentación	1 día	lun 10/11/08	lun 10/11/08	
42	✓	Migración	3 días	lun 10/11/08	mié 12/11/08	
43	✓	**Itemlabel y Label**	**3 días**	**mar 11/11/08**	**jue 13/11/08**	
44	✓	Análisis y documentación	1 día	mar 11/11/08	mar 11/11/08	
45	✓	Migración	3 días	mar 11/11/08	jue 13/11/08	
46	✓	**Image**	**1 día**	**vie 14/11/08**	**vie 14/11/08**	
47	✓	Análisis y documentación	1 día	vie 14/11/08	vie 14/11/08	
48	✓	Migración	1 día	vie 14/11/08	vie 14/11/08	
49	✓	**Toolbars (visualización)**	**10 días**	**lun 17/11/08**	**vie 28/11/08**	28
50	✓	Análisis	1 día	lun 17/11/08	lun 17/11/08	
51	✓	Toolbar	1 día	lun 17/11/08	lun 17/11/08	
52	✓	ToolbarButton	1 día	lun 17/11/08	lun 17/11/08	
53	✓	ToolbarDataGroup	4 días	lun 17/11/08	jue 20/11/08	
54	✓	ToolbarEditGroup	3 días	vie 21/11/08	mar 25/11/08	
55	✓	ToolbarRootNavigationGroup	2 días	mié 26/11/08	jue 27/11/08	
56	✓	Toolbarparamsgroup	1 día	vie 28/11/08	vie 28/11/08	
57	✓	Toolbarseparator	1 día	vie 28/11/08	vie 28/11/08	
58	✓	Documentación	4 días	mar 25/11/08	vie 28/11/08	

Ilustración 2 - Planificación (II)

59	✓	**Control de eventos**	**10 días**	**lun 01/12/08**	**vie 12/12/08**	
60	✓	Evclick	5 días	lun 01/12/08	vie 05/12/08	
61	✓	Action_dataprops	6 días	mié 03/12/08	mié 10/12/08	
62	✓	Action_value	2 días	mié 10/12/08	jue 11/12/08	
63	✓	Documentación	5 días	lun 08/12/08	vie 12/12/08	
64	✓	**Toolbars (funcionalidad)**	**10 días**	**lun 15/12/08**	**vie 26/12/08**	49;59
65	✓	Análisis	1 día	lun 15/12/08	lun 15/12/08	
66	✓	Nodestatus	3 días	lun 15/12/08	mié 17/12/08	
67	✓	Patternbox	1 día	jue 18/12/08	jue 18/12/08	
68	✓	ToolbarDataGroup	2 días	jue 18/12/08	vie 19/12/08	
69	✓	ToolbarEditGroup	3 días	vie 19/12/08	mar 23/12/08	
70	✓	ToolbarRootNavigationGroup	4 días	mar 23/12/08	vie 26/12/08	
71	✓	ToolbarButton	1 día	vie 26/12/08	vie 26/12/08	
72	✓	Toolbarparamegroup	1 día	vie 26/12/08	vie 26/12/08	
73	✓	Documentación	5 días	lun 22/12/08	vie 26/12/08	
74	✓	**Menús**	**2 días**	**lun 29/12/08**	**mar 30/12/08**	
75	✓	Análisis y documentación	1 día	lun 29/12/08	lun 29/12/08	
76	✓	Menu	2 días	lun 29/12/08	mar 30/12/08	
77	✓	Menuitem	2 días	lun 29/12/08	mar 30/12/08	
78	✓	Includemenubar	2 días	lun 29/12/08	mar 30/12/08	
79	✓	Menuparamsgroup	2 días	lun 29/12/08	mar 30/12/08	
80	✓	**Tabs**	**3 días**	**mié 31/12/08**	**vie 02/01/09**	
81	✓	Análisis y documentación	1 día	mié 31/12/08	mié 31/12/08	
82	✓	Tab	3 días	mié 31/12/08	vie 02/01/09	
83	✓	Tabstrip	3 días	mié 31/12/08	vie 02/01/09	

Ilustración 3 - Planificación (III)

84	✓	**Mimic**	**1 día**	**lun 05/01/09**	**lun 05/01/09**	
85	✓	Análisis y documentación	1 día	lun 05/01/09	lun 05/01/09	
86	✓	Migración	1 día	lun 05/01/09	lun 05/01/09	
87	✓	**Splitts**	**8 días**	**lun 05/01/09**	**mié 14/01/09**	
88	✓	Análisis	1 día	lun 05/01/09	lun 05/01/09	
89	✓	Splitblock	2 días	mar 06/01/09	mié 07/01/09	
90	✓	Splithorizontal	5 días	mié 07/01/09	mar 13/01/09	
91	✓	Splitvertical	3 días	lun 12/01/09	mié 14/01/09	
92	✓	Documentación	3 días	lun 12/01/09	mié 14/01/09	
93	✓	**Tree**	**4 días**	**jue 15/01/09**	**mar 20/01/09**	
94	✓	Análisis	1 día	jue 15/01/09	jue 15/01/09	
95	✓	Tree	3 días	jue 15/01/09	lun 19/01/09	
96	✓	Treenode	4 días	jue 15/01/09	mar 20/01/09	
97	✓	Treeview	4 días	jue 15/01/09	mar 20/01/09	
98	✓	Documentación	2 días	lun 19/01/09	mar 20/01/09	
99	✓	**ScrollPanel**	**3 días**	**mié 21/01/09**	**vie 23/01/09**	
100	✓	Análisis y documentación	1 día	mié 21/01/09	mié 21/01/09	
101	✓	Migración	3 días	mié 21/01/09	vie 23/01/09	
102	✓	**Table**	**3 días**	**lun 26/01/09**	**mié 28/01/09**	
103	✓	Análisis y documentación	3 días	lun 26/01/09	mié 28/01/09	
104	✓	Migración	2 días	mar 27/01/09	mié 28/01/09	
105	✓	**Listgroup**	**21 días**	**jue 29/01/09**	**jue 26/02/09**	
106	✓	Análisis	5 días	jue 29/01/09	mié 04/02/09	
107	✓	Listgroup	2 días	mar 03/02/09	mié 04/02/09	
108	✓	Function	2 días	jue 05/02/09	vie 06/02/09	
109	✓	Extends	1 día	lun 09/02/09	lun 09/02/09	
110	✓	Action_preload	1 día	lun 09/02/09	lun 09/02/09	
111	✓	Funcionalidad	12 días	mar 10/02/09	mié 25/02/09	107;108;109;110
112	✓	Documentación	5 días	vie 20/02/09	jue 26/02/09	

Ilustración 4 -Planificación (IV)

4.1.2 Diagrama de Gantt.

En primer lugar se muestra el diagrama de Gantt con las tareas resumidas, para poder visualizar claramente el tiempo que es necesario emplear para cumplimentar cada una de ellas. Posteriormente se mostrarán los diagramas de Gantt con las tareas disgregadas.

Ilustración 5 - Diagrama de Gantt

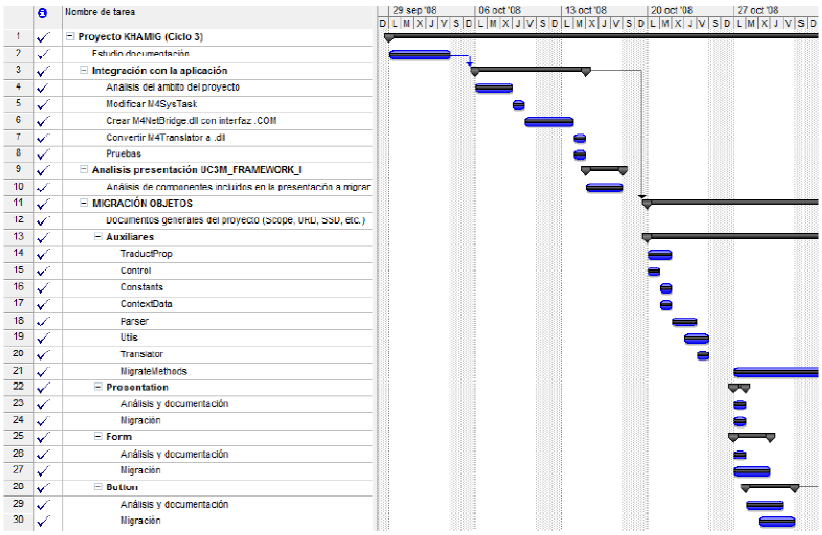


Ilustración 6 - Diagrama de Gantt (I)

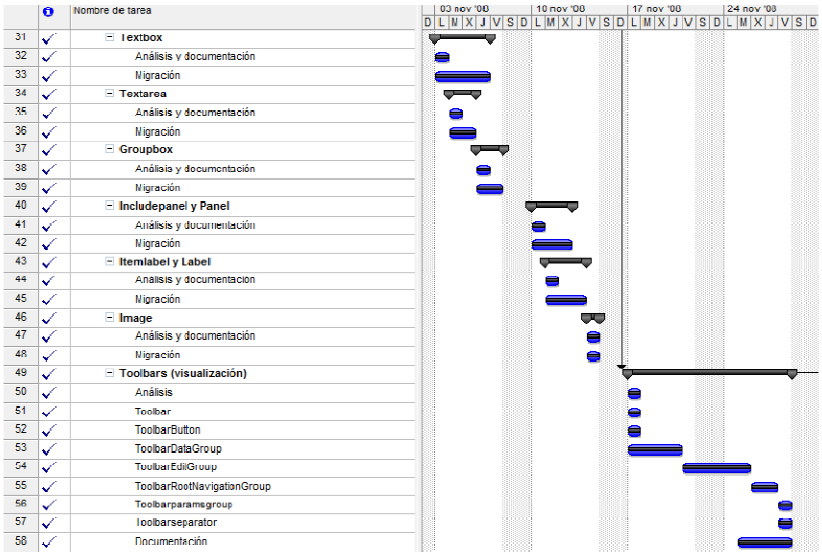


Ilustración 7 - Diagrama de Gantt (II)

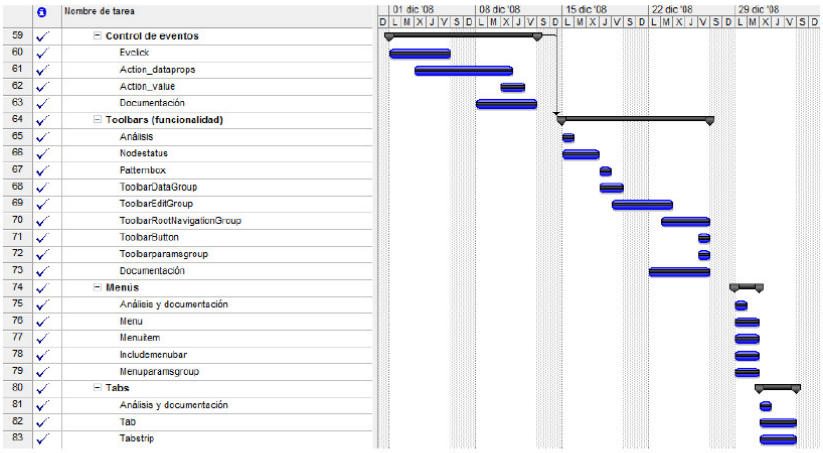


Ilustración 8 - Diagrama de Gantt (III)

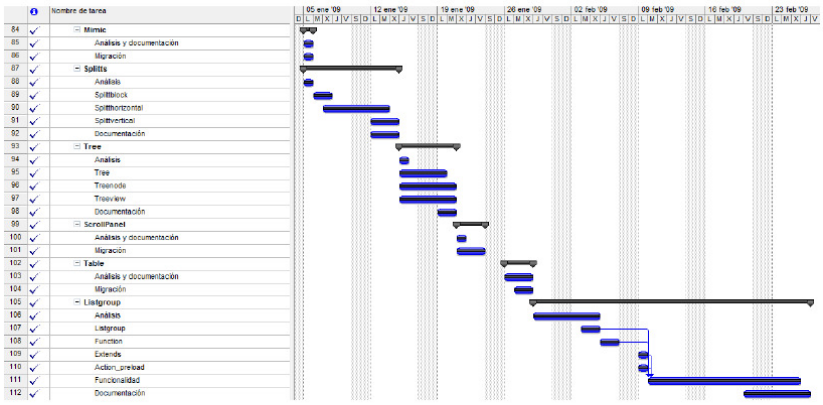


Ilustración 9 - Diagrama de Gantt (IV)

4.1.3 Consideraciones.

La tarea *MigrateMethods* contenida en el apartado *Auxiliares*, en *Migración de objetos*, tiene una duración de 78 días ya que esta tarea consiste en la creación de una clase en C# encargada de albergar los métodos de migración de las propiedades de todos los objetos migrados durante el proceso de migración. En este sentido, dicha clase deberá ser continuamente modificada hasta la finalización de la migración.

Como puede observarse en los anteriores diagramas, todas las tareas planificadas se encuentran finalizadas. Esto no significa que el objeto en cuestión haya sido migrado en su completitud, sino que la tarea de migración estipulada por el equipo de desarrollo para dicho control ha sido finalizada con éxito. Aunque es cierto que gran parte de los objetos mostrados están totalmente migrados, muchos otros requieren de una revisión y un análisis en mayor profundidad, acciones que serán tratadas en el siguiente ciclo del proyecto.

Por último, se considera conveniente comentar que para realizar la migración de ciertos objetos, resulta ventajoso haber migrado ciertos objetos previamente. Sin embargo, estos últimos no aparecen en la planificación como acciones predecesoras de los primeros, ya que para realizar la migración no es estrictamente necesaria la previa migración de dichos objetos.

4.2 Presupuesto.

Concluido el calendario, se procede a ofrecer un presupuesto, ya que éste depende lógicamente del tiempo de vida del proyecto. La fecha de inicio del ciclo desarrollado fue el 29 de Septiembre de 2008, y su fecha de finalización el 27 de Febrero de 2009, lo que suponen 110 días de trabajo en los 5 meses de duración de esta fase del proyecto. El número total de horas a trabajar en estos tres meses ha sido: 880 horas.

Plazo	29 de Septiembre de 2008	27 de Febrero de 2009
Número total de días	110 días	
Número total de horas	**880 HORAS**	

Tabla 1 - Coste temporal

Para calcular el presupuesto habría que sumar al coste monetario de los recursos humanos (en este caso, mi propia persona), el coste del software utilizado. El coste software consistiría básicamente en la adquisición de equipos y licencias de tecnología y herramientas, explicadas estas dos últimas en los dos siguientes apartados. Sin embargo, los equipos utilizados, así como la tecnología y herramientas empleadas han supuesto un coste nulo, debido a que la Universidad Carlos III y la empresa en la que ha sido desarrollado el proyecto han proporcionado dichas utilidades. De esta manera, el coste monetario está basado en los recursos humanos.

IVA	Rol	Dedicación	Coste	TOTAL
Sin IVA	Ingeniero del software	880 horas	30 € / hora	**26.400 €**
Con IVA (16%)	Ingeniero del software	880 horas	34,8 € / hora	**30.624 €**

Tabla 2 - Coste monetario

IV. Apéndice D. Manual de usuario.

En el presente apéndice se recoge el Manual de Usuario del sistema migrador desarrollado, dónde se describe la situación de un usuario cuyo deseo sea la migración de una presentación en OBL, a una presentación con la que interactuar en el entorno .NET.

ISDAC

Integración, Soporte y Difusión de Aplicaciones Complejas

MANUAL DE USUARIO

Versión:	0.01
Creación:	20-01-2009
Última Modificación:	06-04-2009
Preparado por:	Luis Miguel Esteban
Revisado por:	UC3M1

Resumen

El presente documento consiste en el Manual de Usuario de la herramienta desarrollada en el proyecto software, consistente en el proceso de reingeniería del software del proyecto KHAMIG.

Hoja de Estado del Documento

1. *ISDAC.* Integración, Soporte y Difusión de Aplicaciones Complejas. Software Specification Document.		
2. Referencia del documento: UserManual		
3. Versión	**4. Fecha**	**5. Motivación del Cambio**
0.01	20-01-2009	Primera versión interna del documento.

Tabla 1. Hoja de estado del documento.

Registro de Cambios

1. Versión	2. Fecha	3. Autor
0.01	20/01/2009	Luis Miguel Esteban
4. Descripción del Cambio		
• Primera versión interna del documento.		

Índice de Contenidos

Resumen i
Hoja de Estado del Documento i
Registro de Cambios ii
Índice de Contenidos iii
Índice de ilustraciones iv
1 Introducción 1
1.1 Propósito del documento 1
1.2 Ámbito del software 1
1.3 Definiciones, acrónimos y abreviaturas 2
1.3.1 Definiciones 2
1.3.2 Acrónimos 3
1.3.3 Abreviaturas 3
1.4 Referencias 3
1.5 Estructura del documento 3
2 Descripción general 4
3 Proceso preparativo de la migración. 5
3.1 Instalación de la tecnología 5
3.2 Diseñador de menús. 5
3.3 Ejecución desde MVS. 6
4 Proceso de migración. 7
4.1 Diseñador de menús. 7
4.2 Ejecución de la tarea para la migración. 9

Índice de ilustraciones

Ilustración 1 - Plataforma propiedad de la compañía 7
Ilustración 2 - Diseñador de menús 8
Ilustración 3 Tarea (Identificador de la presentación a migrar) 9
Ilustración 4 – Menú con la tarea ya asociada 10

1 Introducción

El presente documento describe el *Manual de Usuario* del proyecto de desarrollo de un plan de migración que permita traducir interfaces de usuario creadas con un código obsoleto, a interfaces de usuario con las que interactuar en un entorno .NET.

Esta sección aborda los objetivos tanto del plan de migración como de este documento en sí mismo. También recoge las definiciones y las referencias que aparecen en el resto del documento y una breve descripción del contenido de las secciones que lo conforman.

1.1 Propósito del documento

El *Manual de usuario* surge con la necesidad de establecer una serie de pasos dirigidos al usuario, de modo que pueda seguirlos con el fin de asegurar la migración de la presentación.

1.2 Ámbito del software

El software a desarrollar, tomando como base los requisitos definidos en el presente documento, tiene como objetivo la traducción a WPF de una presentación OBL generada por la plataforma propiedad de la compañía para su lanzamiento en el entorno .NET. El producto obtenido es una solución concebida como un conjunto de proyectos en el entorno .NET que integrará el conjunto de la funcionalidad descrita. Así, el principal beneficio del citado

producto es aglutinar la funcionalidad de traducción y muestra de la presentación que se ha tomado como base para el P01C03. Dicha presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un .EXE ni efectuar almacenamientos en el repositorio de la compañía.

1.3 Definiciones, acrónimos y abreviaturas

En esta sección se recogen las definiciones, acrónimos y abreviaturas que aparecen en este documento.

1.3.1 Definiciones

Diseñador de presentaciones
Herramienta disponible en la plataforma propiedad de la compañía. Su fin consiste en elaborar presentaciones, bien gráficamente, bien desarrollando código manualmente.

Presentaciones
Interfaces de usuario creadas con un propósito, por ejemplo, un mantenimiento de empleados.

Controles / Objetos
Elementos que aparecen en el código referente a las presentaciones con el fin de representar un componente de la presentación.

Diseñador de menús
Herramienta disponible en la plataforma propiedad de la compañía. Su fin consiste en definir menús que estarán compuestos por tareas, donde se indicará la presentación a migrar.

Tareas
Elemento de un menú donde se indica el identificador único de una presentación que se pretende migrar.

1.3.2 Acrónimos

ESA
European Space Agency

MVS
Microsoft Visual Studio

1.3.3 Abreviaturas

No aplicable.

1.4 Referencias

[1] Esteban Santiago, Luis Miguel. Proyecto Fin de Carrera. 2009.

1.5 Estructura del documento

El presente documento se estructura de la siguiente manera. Tras la sección uno desarrollada, en primer lugar, la sección dos incluye una descripción general acerca de la tecnología necesaria para hacer posible el proceso de migración [1]. La sección tres consiste en la descripción de una serie de pasos para preparar la migración. Por último, la sección cuatro mostrará visualmente cómo realizar el proceso de migración a tecnología .NET.

2 Descripción general

El proyecto de reingeniería desarrollado establece que cualquier presentación será regenerada en formato WPF y permitirá su aprovechamiento a través de la plataforma .NET sin necesidad de compilar un ejecutable (.EXE) ni efectuar almacenamientos en el repositorio de la compañía. Por ello, el usuario simplemente necesitará:

- ✓ El framework .NET.
- ✓ Las librerías que permiten la migración.
- ✓ La plataforma donde se encuentra el Diseñador de presentaciones, y donde se encuentra a su vez el apartado que ofrece la migración de la presentación deseada.

El sistema ha sido desarrollado para máquinas con sistema operativo Windows, por lo que, además, el usuario necesitará disponer de dicho entorno para efectuar la migración.

3 Proceso preparativo de la migración.

El usuario hará posible la migración de las presentaciones si sigue los pasos indicados en esta sección.

3.1 Instalación de la tecnología.

El primer paso, lógicamente, será disponer de la tecnología indicada en el apartado 2. A partir de entonces, procedemos a realizar los siguientes pasos.

3.2 Diseñador de menús.

En la plataforma propiedad de la compañía existe un apartado denominado "Herramientas de desarrollo". Como hijo directo de este apartado, se tiene un subapartado "Menús". Será aquí donde el usuario deberá indicar la presentación que desea migrar, en el apartado de "Tareas".

En la propia plataforma, en el apartado "Herramientas de desarrollo", se generará una tarea con el nombre que el usuario haya especificado, la cual al ser ejecutada levantará la presentación migrada correspondiente a la presentación que el usuario indicó en el paso anteriormente explicado en este mismo apartado.

3.3 Ejecución desde MVS.

Por ahora, debido a que el proyecto de migración no ha llegado a su conclusión total, para el correcto funcionamiento de la migración será necesario ejecutar el proyecto de migración desde Microsoft Visual Studio, con la finalidad de cargar todos los recursos textuales y gráficos, así como el entorno de ejecución.

De este modo, este paso actuará como paso previo del anteriormente explicado, levantando la plataforma propiedad de la compañía donde se encuentra el menú con la tarea asociada que indica la presentación que el usuario pretende migrar.

4 Proceso de migración.

En este apartado se muestra, mediante una serie de imágenes, cómo llevar a cabo la migración de una presentación a tecnología .NET.

4.1 Diseñador de menús.

En la plataforma propiedad de la compañía existe un apartado denominado "Herramientas de desarrollo". Como hijo directo de este apartado, se tiene un subapartado "Menús".

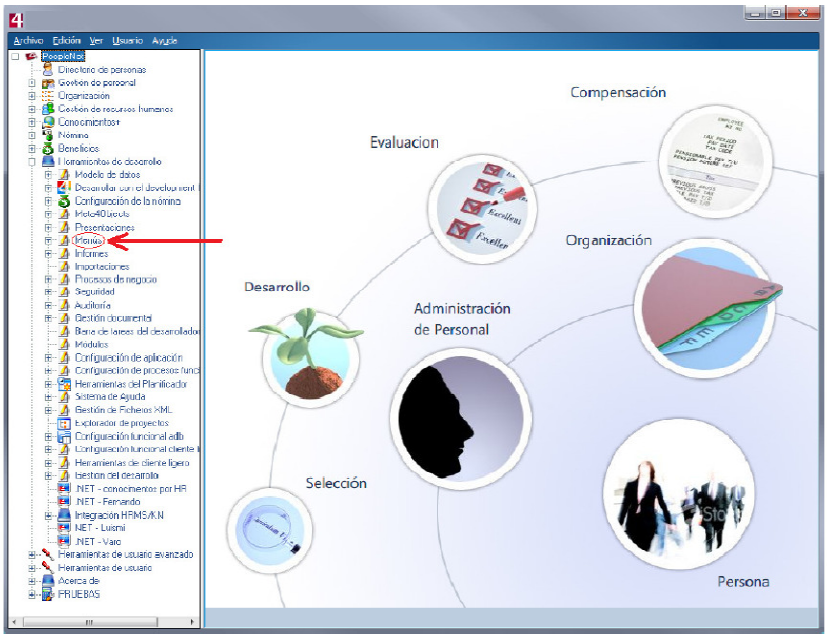


Ilustración 1 - Plataforma propiedad de la compañía

El usuario accederá al diseñador de menús para proceder a indicar la presentación que desea migrar, en el apartado de "Tareas".

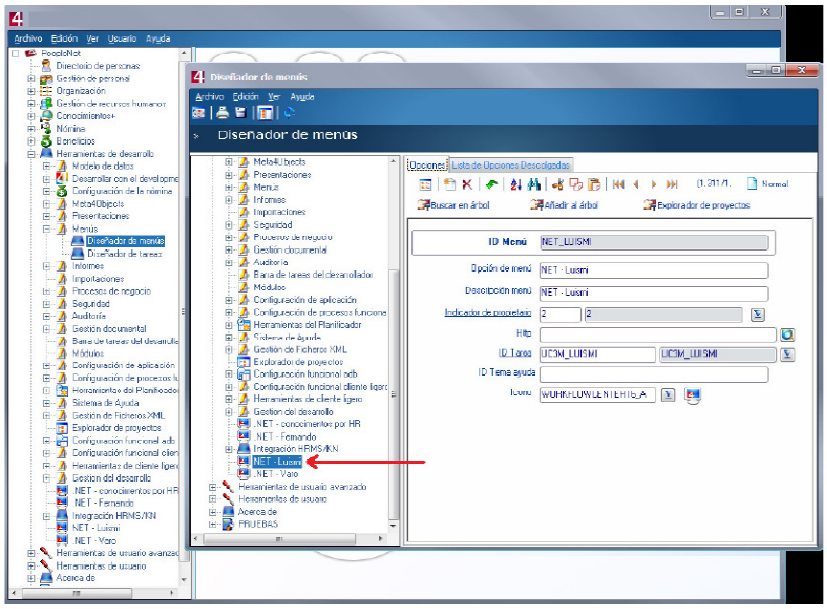


Ilustración 2 - Diseñador de menús

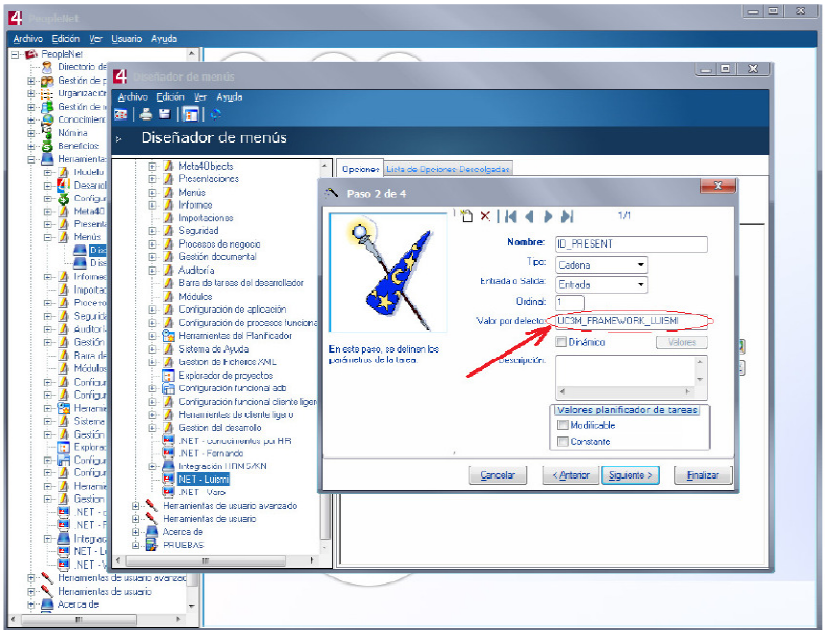


Ilustración 3 - Tarea (Identificador de la presentación a migrar)

En la propia plataforma, en el apartado "Herramientas de desarrollo", se generará una tarea con el nombre que el usuario haya especificado, la cual al ser ejecutada levantará la presentación migrada correspondiente a la presentación que el usuario indicó en el paso anterior.

4.2 Ejecución de la tarea para la migración.

Una vez creada la tarea, el usuario ejecutará desde Microsoft Visual Studio el proyecto de migración, debido a los fines explicados anteriormente.

Se levantará la plataforma y aparecerá el menú creado por el usuario con la tarea asociada que indica la presentación que el usuario pretende migrar. Una vez ejecutada dicha tarea, se procede a la migración (transparente al usuario) de la presentación.

Ilustración 4 – Menú con la tarea ya asociada

En el momento en el que la presentación aparece migrada en el entorno .NET, el usuario podrá interactuar con ella del mismo modo que lo hacía con la presentación generada con el sistema *legacy*.

CAPÍTULO VII. BIBLIOGRAFÍA.

Bibliografía.

- *Adeoti-Adekeye, W. B. "The importance of management information systems." Library Review 46, 1997.*
- *Argyris, C. "Management Information Systems: The Challenge to Rationality and Emotionality." Management Science 17, 1971.*
- *Barreu, D. "The hidden costs of implementing and maintaining information systems." The Bottom Line: Managing Library Finances 14, 2001.*
- *Bass, Len, Paul Clements and Rick Kazman. "Software Architecture in Practice". Addison-Wesley, 1998.*
- *Behling, Robert, Chris Behling, and Kenneth Sousa. "Software re-emgineering: concepts and methodology." Industrial Management & Data Systems, 1996.*
- *Bergey, John, Liam O'Brien, and Dennis Smith. "DoD Software Migration Planning." Technical Note, 2001.*
- *Bergey, John, Dennis Smith, and Nelson Weiderman. "DoD Legacy System Migration Guidelines." Technical Report, 1999.*
- *Broderick, R. F., and J. W. Boudreau. "Human Resource Management, Information Technology,and the Competitive Edge. Center for Advanced Human Resource Studies", Cornell University, ILR School, 1991.*
- *Cabezas, Proyecto Fin de Carrera, "Marco para la reingeniería de productos de software", 2008.*
- *Colomo-Palacios, García-Crespo & Ruano-Mayoral. "EuroSPI", 2008*
- *De Pablos Heredero, "C. Dirección y Gestión de los Sistemas de Información en la Empresa". Madrid: ESIC Editorial: Universidad Rey Juan Carlos, Servicio de Publicaciones, 2001.*

- *Dragandis, F., and G. Mentzas. "Competency based management: a review of systems and approaches." Information Management & Computer Security 14, 2006.*

- *Duff, N. M., and M. G. Assad. "Information Management: An Executive Approach. Oxford". University Press, 1980.*

- *ESA Board for Software Standarisation and Control. "Guide to applying the ESA software engineering standards to small software projects". European Space Agency, 1996.*

- *ESA Board for Sotfware Standarisation and Control. "ESA Software Engineering Standards ESA PSS-05-0. Issue 2". European Space Agency, 1991.*

- *Gardner, B. "A fruitful investment." Computer-world, 1993.*

- *Hammer, and Champy. "Reengineering the Corporation: A Manifesto for Business Revolution". HarperCollins, 1994.*

- *Hooper, J. W., and R. O. Chester. "Software Reuse: Guiudelines and Methods". New York: Plenum Publishing Corporation, 1991.*

- *Kadiyala, R., and B. H. Kleiner. "New developments concerning business information systems." Management Research News 28, 2005.*

- *Katrib, Del Valle, Sierra, Hernández. "Windows Presentation Foundation". 2007*

- *Kozaczynski, W., and N. Wilde. "Understanding How Programs Use Data: An Aid to Re-Engineering Transaction Processing Systems". Chicago, Illinois: Andersen Consulting, 1989.*

- *Langemo, M. "Records management/word processing–a needed team effort". Records Management Quarterly 14, 1980.*

- *Laudon, K. C., and J. P. Laudon. "Management Information Systems: New Approaches to Organization and Technology". 5 ed. London: Prentice Hall International, 1998.*

- *Laudon, K. C., y J. P. Laudon. "Sistemas de información gerencial: Organización y tecnología de la empresa conectada a la red". 6 ed. México: Pearson Educación, 2002.*

- *Melián-González, S. "Improving human resources management: some practical questions and answers". International Journal of Contemporary Hospitality Management 16, 2004.*

- *Ngai, E. W. T., and F. K. T. Wat. "Human resource information systems: a review and empirical analysis." Personnel Review, 2006.*

- *Poutanen, H. "Human Resource Management and Human Resource Information Systems in Organizations".*

- *Premerlani, W., y M. Blaha. "An approach for reverse engineering of relational data-bases". Communications of the ACM, 1994.*

- *Roger S. Presuman. "Ingeniería de Software, un enfoque práctico", 5ª ed.*

- *Sasso, William. "Cognitive Processes in Program Comprehension: An Empirical Analysis in the Context of Software Reengineering". 1996.*

- *Serrano, Miguel A., Doris L. Carver, and Carlos Montes de Oca. "Reeingineering legacy systems for distributed enviroments." The Journal of Systems and Software, 2002.*

- *Tannenbaum, S. I. "HRIS: user group implications." Journal of Systems Management 41, 1990*

Recursos electrónicos.

- *M.A. Sicilia, artículo electrónico sobre Técnicas del mantenimiento del software, http://cnx.org.*
- *Manual electrónico sobre la plataforma .NET, 2004.*
- *MSDN, 2008.*
- *MSDN, 2009.*
- *Real Academia de la Lengua Española, http://www.rae.es.*
- *www.microsoft.com.*
- *www.w3counter.com, 2009*