UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Magíster en Ingeniería de Software

Testing de migración de aplicaciones distribuidas a entornos Web

Tesis

Autora: C.C. María Elena Ciolli

Director: Mgter. Ing. Juan Francisco Giró

Codirector: Dr. Gustavo Rossi

- Diciembre de 2007 -

A mi familia y alumnos de la Universidad

AGRADECIMIENTOS

Un agradecimiento especial al Mgter. Ing. Juan Francisco Giró, por haber aceptado la dirección de mi tesis y por haberme brindado su ayuda y aliento durante todo su desarrollo, y al Dr. Gustavo Rossi, por aceptar codirigirla y allanar el camino para su concreción.

Un gracias y mi amor incondicional a mi familia, ya que sin su cariño, apoyo y colaboración esta tesis no hubiera sido posible.

A mis alumnos de la Universidad por su interés permanente en los temas desarrollados aquí y en su aliento para que pudiera finalizar este estudio.

RESUMEN

La aplicación de la tecnología computacional a lo largo de ya varias décadas ha conducido a sistemas de información grandes y complejos, muchos de los cuales parecen haber alcanzado un punto límite en sus posibilidades de adecuación y evolución. Si bien este problema viene siendo advertido desde hace mucho tiempo, el crecimiento exponencial de la Web y la necesidad de mantener competitividad en un mundo dinámico y cambiante le dieron al problema una trascendencia adicional. En efecto, las organizaciones deben hacer un efectivo aprovechamiento del potencial que ofrece la Web en plazos perentorios, pero al mismo tiempo, gran parte de su conocimiento, experiencia y reglas del negocio están concentrados en estos sistemas de información que parecen inmunes a su adaptación.

Por esta razón, durante los últimos años se comprueba un creciente número de experiencias de migraciones a la Web de aplicaciones distribuidas no basadas en la Web, convirtiéndose este tema en un importante campo de investigación y práctica profesional de la Ingeniería de Sistemas.

Es así que se vienen presentando y ensayando diferentes propuestas, se desarrollaron productos y se dispone de amplia bibliografía para la migración de sistemas apoyados en plataformas mainframes tradicionales a entornos abiertos. Sin embargo, no hay tanta información referida a la realización de las pruebas de migración y mucho menos sobre las características de las validaciones de las aplicaciones en entornos distribuidos sobre la Web.

Esta comprobación sirvió de estímulo para el trabajo que se presenta, en el que se aborda el problema de migración de sistemas a la Web desde el enfoque de testing, estudiando en especial las especificaciones de requerimientos de interfases con el usuario y funcionales.

Para ello se formula una metodología para el análisis lógico y físico de las aplicaciones distribuidas a migrar a entornos Web, y se la pone en práctica aplicándola a un caso de estudio. Este caso corresponde a un sistema distribuido desarrollado mediante el uso de una metodología de análisis y diseño estructurado y la aplicación migrada a la Web fue desarrollada mediante el uso de alguna metodología basada en UML.

El trabajo realizado se apoya en el conocimiento previo sobre testing de regresión, testing de caja negra y testing de interfaces gráficas con el usuario. Asimismo, dado que el ciclo de vida del testing está embebido dentro del ciclo de vida del software, el enfoque propuesto promueve la reutilización de casos de pruebas existentes, resultantes de la trazabilidad con los casos de uso planteados para el sistema distribuido, y la automatización de las pruebas unitarias, de integración y de regresión.

INDICE DE CONTENIDOS

INDICE DE CONTENIDOS (Cont.)

CAPITULO 1. INTRODUCCIÓN

La mayor parte de los grandes sistemas de información que están hoy funcionando en las organizaciones de nuestro país fueron desarrollados en los años ochenta. La irrupción de las tecnologías relacionadas con Internet, el paradigma de objetos, los componentes distribuidos y la nueva mentalidad empresarial que intenta ofrecer mejores servicios a sus clientes, han provocado que la información que permanece en los viejos sistemas y que es totalmente aprovechable sea objeto de diversos tratamientos a los fines de su recuperación. Más aún, en muchos casos se trata de conocimiento y experiencia sobre reglas del negocio que no están disponibles en ningún otro medio y esto lo convierte en un problema que puede considerarse crítico.

La situación tecnológica en los años 80 sólo permitía una arquitectura física y lógica restringida a la oferta de los grandes fabricantes de software y hardware, quienes suministraban ambos componentes, por lo que se obtenía en consecuencia una dependencia total del cliente. Los sistemas fueron luego sometidos a un mantenimiento constante, normalmente indocumentado, lo que en muchos casos originó degradación de las aplicaciones y, por ende, un servicio deficiente para el usuario.

Se llega así al concepto de “sistemas heredados” (Legacy Systems) que admiten diversas definiciones según el punto de visto considerado. Así Ulrich (1994) hace referencia a “sistemas independientes construidos en una era tecnológica anterior que disponen de precaria documentación”, Brodie (Brodie y Stonebraker, 1995) los define como “todo sistema de información que se resista significativamente a su modificación y evolución, para cubrir cambios en sus requerimientos”, y Bennett (1995) habla de “grandes sistemas parcialmente desconocidos y vitales para las organizaciones”.

Ante una situación como la descrita en los párrafos precedentes, las organizaciones se vieron en la necesidad de tomar una determinación que puede ser resumida como un “cambio del sistema”, para adecuarlo a las nuevas necesidades, y que puede concretarse a través de una de las opciones siguientes:

a) Hacer reingeniería o migrar el sistema.

b) Abandonar el sistema y sustituirlo por otro nuevo.

c) Optar por una solución híbrida entre las dos anteriores.

Como denominador común de todas las opciones anteriores, la organización debió plantearse la inclusión, dentro del proyecto correspondiente, de una buena gestión de la evolución del nuevo software que se produzca, con el objetivo de no volver a caer en la situación de la que buscaban salir. Las pequeñas y grandes aplicaciones distribuidas, desarrolladas antes de la irrupción de la Web, que basaron su proceso de construcción en ambientes cliente – servidor estaban orientadas a la conexión en red de ordenadores personales (clientes), con servidores. En la mayoría de los casos, la herramienta de desarrollo permitía a equipos de programadores crear aplicaciones diseñadas con interfaces gráficas de usuario (GUI), y con acceso a información de base de datos locales o en servidores de red.

En este contexto, desde los 80 y hasta principios de los 90, maduraron diversos métodos de análisis y diseño de software. Cabe mencionar el uso en gran escala del principio de ocultación de la información; la aplicación del análisis y diseño estructurado para desarrollo de aplicaciones concurrentes, distribuidas y de tiempo real; el modelado de entidades del dominio del problema en términos de eventos; el uso de tareas concurrentes asociadas a entidades y los métodos de análisis y diseño orientado a objetos (Jacobson, Rumbaugh & Booch, 2000).

Tal como señala Piattini (2004), estos tipos de arquitecturas ofrecen una serie de inconvenientes; entre los cuales se destacan: a) falta de flexibilidad de los sistemas, debido al acoplamiento existente entre las aplicaciones y la base de datos, lo que origina que una modificación en la base repercuta en la necesaria modificación de las aplicaciones asociadas; b) dificultades de migración, por la dependencia con el fabricante; c) problemas de escalabilidad, al incrementar el número de usuarios y d) dificultad de reutilización de componentes, debido al acoplamiento entre el nivel de presentación (interfaz del usuario) y los sistemas de gestión de base de datos.

En tanto, como evolución importante de la computación distribuida cliente/servidor, surge la tecnología Web; y, en consecuencia con ello, el interrogante sobre la conveniencia de migrar las aplicaciones existentes de tipo cliente-servidor a dicha arquitectura; para lo cual resulta necesario revisar algunas de las características diferenciadas que presentan estas tecnologías. En este sentido, se observa que en las arquitecturas tradicionales, la interfaz del sistema se instala en la computadora del usuario final; mientras que la arquitectura basada en la Web transforma la interfaz de búsqueda existente, que es el explorador Web, generalmente Netscape o Internet Explorer, en la interfaz del usuario final. Ésta es la razón por la cual los exploradores mencionados se han convertido en el estándar de la mayoría de las PC. Estos navegadores fueron diseñados, en principio, como herramientas de software con soporte gráfico, con el fin de desplazarse a través de nodos de información mediante el hipertexto. Con el tiempo, el navegador resultó una herramienta de trabajo para el acceso a otros sistemas, y fue sustituyendo paulatinamente a herramientas que estaban diseñadas para tareas específicas, tales como clientes de correo, usenet, etc.

Así, el navegador convirtió su funcionalidad al evolucionar desde un dispositivo de acceso a documentos y gráficos a un escritorio (en terminología Windows), desde donde el usuario accede, de forma transparente, a multitud de aplicaciones. Por ejemplo, por medio de un navegador, es posible acceder a un sistema central, a sistemas de información, a bases de datos y repositorios de contenidos, sin necesidad de cambiar de herramienta y mediante la acción de “navegar”. En todos los casos, la Web no sólo resulta una herramienta fundamental de comunicación, sino que brinda la posibilidad al usuario final de realizar múltiples operaciones sin moverse de su casa o de su lugar de trabajo, a costos cada día más accesibles.

Sin embargo, tras el nacimiento de la tecnología Web, surgió como problema la necesidad de que los productos de migraciones a la Web desde aplicaciones distribuidas no Web cumplan al menos con las especificaciones de estas últimas. El problema es aún más complejo, ya que una aplicación Web requiere no sólo corrección en la funcionalidad, sino también una buena interfase tanto en la presentación como en las opciones de corrección y accesibilidad de los contenidos.

Existen productos y experiencia comprobada para realizar pruebas de migración de sistemas a diferentes plataformas operativas y de motores de bases de datos, incluyendo la consideración de costos, eficiencia, integridad, seguridad, velocidad de respuesta, cantidad de usuarios, carga del sistema, entre otros datos. Como ejemplos, pueden citarse las experiencias de los ya mencionados sistemas heredados desde plataformas origen tales como mainframes IBM y UNISYS hacia sistemas abiertos en Windows, UNIX o Linux. En este contexto, también puede mencionarse la conversión de sistemas desarrollados bajo lenguajes propietarios (RPG, COBOL, Natural, etc.) hacia Java u otros sistemas de código abierto.

En todos estos casos, los sistemas son convertidos manteniendo exactamente la misma funcionalidad que los originales, sin modificar los procedimientos de la organización que los utiliza, pero tal como ya fue anticipado, muy poco se menciona de la consolidación y características de las pruebas y validaciones de las aplicaciones en entornos distribuidos a la Web.

Entre los ejemplos de migraciones exitosas de grandes sistemas distribuidos a la Web, pueden citarse, entre otros, los que corresponden a las siguientes aplicaciones:

- Reserva de pasajes a través de sistemas, tales como *SABRE* o *AMADEUS* a los que se accede mediante portales de agencias de viajes o compañías aéreas.
- Operaciones bancarias on-line, que permiten a los usuarios operar sobre sus cuentas bancarias, en un nuevo concepto denominado HomeBanking que evita la concurrencia de los clientes a las instituciones.
- Sistemas de autogestión de alumnos de instituciones educativas, para la inscripción en materias y exámenes, modificación de datos personales, etc.

A los fines de este trabajo, se considera el caso de aquellas aplicaciones distribuidas no basadas en la Web, para las cuales se dispone de especificación, diseño y código, y para las cuales se determinó la conveniencia de construir una aplicación basada en la Web que preservara las principales propiedades de la aplicación original, tales como son su especificación, funcionalidad y propiedades de la interfase gráfica con el usuario. Para ello, se enfoca este estudio desde el punto de vista del testing, a partir de la consideración de todos los aspectos a cubrir en el mismo. Se definen criterios y estrategias para validar especificaciones, funcionalidades e interfases.

1.1. Objetivos Generales

En el presente trabajo se determinan los siguientes objetivos generales:

1. Estudiar la reutilización de requisitos de aplicaciones distribuidas a aplicaciones Web, a partir de la trazabilidad de casos de uso de ambas aplicaciones.
2. Definir metodologías de trabajo para la prueba de los resultados de las migraciones a la Web de aplicaciones distribuidas no realizadas bajo entorno Web, a partir de la consideración de las propiedades que se quieren preservar de estas últimas.
3. Documentar la información recopilada en función de los resultados obtenidos luego de realizar las pruebas de migración.

1.2. Objetivos Específicos

A los fines de alcanzar los objetivos generales propuestos, se plantean los siguientes objetivos específicos:

1. Construir metamodelos para abstraer propiedades en común de modelos de aplicaciones Web y de modelos de aplicaciones tradicionales, los que sirven de base para mapear casos de uso utilizados en aplicaciones distribuidas no Web a casos de uso de aplicaciones basadas en tecnología Web.
2. Formular una metodología de análisis para la migración de aplicaciones distribuidas a entornos Web, basada en un enfoque de testing con reutilización de casos de prueba y que prevea la utilización de herramientas automáticas para la ejecución y reuso de los casos de prueba generados.
3. Aplicar la metodología propuesta tomando como caso de estudio el Sistema de Gestión Académica de una institución universitaria a efectos de comprobar su desempeño.

Con referencia al primero de los objetivos específicos planteados, cabe aclarar que la formulación de metamodelos se justifica por proporcionar un contexto para resolver las transformaciones y representación explícita de correspondencias entre los distintos modelos que se presenten en el trabajo en estudio. Así, mediante esta plataforma se pretende definir una base formal que permita representar los modelos de ambas arquitecturas e integrarlos. Asimismo, teniendo en cuenta que el ciclo de vida del testing

está embebido dentro del ciclo de vida del software, el enfoque de prueba promoverá la reutilización de casos de prueba existentes, y la automatización de la creación de un repositorio con los casos de pruebas realizados. Si bien ello puede, inicialmente, parecer un trabajo arduo, la ejecución de estas pruebas de regresión permitirá sistematizar el testing y evitar la redundancia en el proceso de detección de errores.

1.3. Estructura de la Tesis

El Capítulo 2 presenta el problema de la migración de sistemas a la Web y el modo de gestionar su evolución. Asimismo, plantea algunos de los interrogantes que se formulan previo a la realización de una migración de una aplicación cliente/servidor a Web, las posibles soluciones a los mismos y una exposición de los principales beneficios que se obtienen con este proceso.

El Capítulo 3, por su parte, realiza una comparación de dos enfoques, el Testing clásico y el Web testing; y enuncia las dificultades que presenta este último como consecuencia de la naturaleza de esta plataforma.

En el Capítulo 4 se formula una metodología para el análisis lógico y físico de un sistema a migrar de arquitectura cliente/servidor a Web y propone un enfoque de Testing de migración basado en la corrección funcional o testing de caja negra y la reutilización de requisitos en el proceso de migración a la Web. Dicho proceso de migración se presenta guiado por los casos de uso, y se utiliza la notación UML para la representación de los modelos de requisitos y pruebas. En base a la comparación de los modelos de requisitos se plantea un modelo de reutilización e interacción de los casos de uso y su trazabilidad para los casos de prueba derivados de los mismos, lo que origina un modelo de configuración de pruebas del cual se realiza un análisis detallado de sus componentes.

El Capítulo 5 aplica la metodología de migración tomando como caso de estudio el sistema de Autogestión de Alumnos del Instituto Universitario Aeronáutico. A lo largo de este capítulo, se da cuenta de un estudio comparativo y se aplican las metodologías utilizadas para el desarrollo del sistema antiguo y del actual. En este desarrollo, se centra el foco de atención en la reutilización de requisitos en el proceso de migración a la Web, y se propone un metamodelo de la construcción de la interfaz de ambas aplicaciones.

El Capítulo 6 plantea la aplicación del enfoque funcional del testing de migración al caso de estudio presentado en el capítulo anterior, mediante la reutilización y estudio comparativo de los casos de uso/casos de prueba principales de ambos sistemas. Permite visualizar, convenientemente, las diferencias existentes entre los mismos, para lo cual se presenta una clasificación de las condiciones que les dan origen y la posibilidad de su reutilización. Asimismo se realiza un estudio comparativo de la interfaz de ambos sistemas considerando las diferencias tecnológicas existentes.

En el Capítulo 7 se presenta una herramienta automática para la generación de datos de prueba y su entrada a las pruebas unitarias de los sistemas, complementada con una herramienta robot para las pruebas de integración y regresión, y, por último una herramienta para la verificación automática de enlaces en la aplicación web.

En el Capítulo 8 se realiza una reseña de los trabajos relacionados con esta tesis muchos de los cuales sirvieron de base para la elaboración de esta propuesta.

Por último, se presentan las conclusiones a las que se arriba tras el desarrollo propuesto, los resultados alcanzados con la investigación teórico/práctica realizada, y los aportes devengados gracias a la realización de la tesis.

CAPÍTULO 2. EL PROBLEMA DE LA MIGRACIÓN

2.1. Introducción

A los fines de introducir el tema desarrollado en este capítulo, resulta importante precisar el alcance de la palabra *migración*. Una de las acepciones del término hace referencia a la acción de convertir los programas de un lenguaje a otro, habitualmente desde lenguajes como el Cobol hacia el Java, lo que en este caso implica cambiar el paradigma de construcción de las aplicaciones desde un modelo procedimental hacia un modelo orientado a objetos.

En una interpretación más abarcativa, se hace referencia a la migración de un sistema de computación cuando se lo traslada de una plataforma a otra, lo que puede involucrar cambios de arquitectura y/o de tecnología, y normalmente lleva implícita la necesidad de reescribir los programas en un lenguaje diferente.

Si solo se considera la conversión de los lenguajes de los programas, en el mercado existen traductores de código que tienen la finalidad de contribuir a facilitar esta tarea. Sin embargo, estos cumplen una función esencialmente sintáctica, normalmente pobre desde el punto de vista semántico, y sus resultados se alejan demasiado del objetivo deseado. La traducción del código sin cambio en el paradigma conduce a programas monolíticos, ineficientes y difícilmente mantenibles. Por el contrario, si se considera el concepto de migración en su interpretación más amplia, el problema adquiere la dimensión de un proyecto de ingeniería de software y debe ser tratado en consecuencia, para lo cual se presentan diferentes alternativas.

El concepto de migración de un sistema no está taxativamente definido y en muchos casos se lo confunde con el de reingeniería, por lo que, para comenzar, es necesario aclarar el alcance de ambos términos. Se entiende como reingeniería a la casi completa reconstitución y reimplementación de un sistema, sin que haya necesariamente un cambio de plataforma o ambiente de operación. Por el contrario, la migración evita el redesarrollo completo del sistema al usar todos los antecedentes disponibles (requerimientos, diseños, etc.) y siempre implica un cambio en el ambiente de operación. Por lo tanto, al hablarse de migración se está haciendo referencia a la necesidad de trasladar un sistema a una nueva plataforma manteniendo sus funcionalidades y provocando mínimo impacto en su operación.

A los fines de comprender la importancia de este planteo se reitera la situación que enfrentan muchas organizaciones en la actualidad: la necesidad de trasladar aplicaciones informáticas críticas para el negocio, y que necesitan ser adaptadas para su funcionamiento a los canales que ofrecen las nuevas tecnologías, tales como Internet.

2.2. Características y consecuencias del problema

Tal como ya fue expuesto, la sucesiva evolución de la tecnología computacional y el paso del tiempo han conducido a que muchos sistemas informáticos incorporen un conjunto de características no deseadas y desafortunadas que son las siguientes:

- Operan sobre hardware obsoleto, que es lento y costoso de mantener.
- El mantenimiento del software también es costoso y lento, principalmente por la falta de documentación y de conocimiento de la estructura interna del sistema.
- Los esfuerzos de integración se ven muy limitados por la ausencia de interfases.
- Por las razones señaladas son sistemas muy difíciles de expandir.

En respuesta a estos problemas se han propuesto diversas soluciones que pueden ser agrupadas en las siguientes tres categorías:

a) Reconstrucción

La reconstrucción implica reescribir las aplicaciones existentes, y dependiendo de la documentación y conocimiento disponible sobre el sistema actual, puede tratarse desde una reingeniería hasta el rediseño de un sistema completamente nuevo. Esto último ya fue referido como abandono del sistema para su sustitución por otro nuevo.

b) Encapsulamiento

Con encapsulamiento se hace referencia al desarrollo de una envoltura de software (wrapper) sobre la aplicación existente, con el fin de dotarlo de interfases con componentes periféricos que permitan sacarlo de su aislamiento.

c) Migración

La migración de un sistema de información tiene por finalidad su traslado a un nuevo ambiente operativo, conservando su funcionalidad y datos originales. En todos los casos se persigue posibilitar el mantenimiento y posterior adecuación a nuevos requerimientos.

Dado un problema concreto de un sistema que reúna las cualidades antes señaladas, muchas veces tipificado como sistema *heredado*, no es siempre posible decidir cuál es la solución más conveniente y en muchos casos lo apropiado es una combinación de ellas. Sin embargo, es muy poco probable que la completa substitución del sistema sea una verdadera opción y la solución práctica del problema suele hallarse entre el encapsulamiento y la migración. La primera es muchas veces reconocida como una solución de compromiso o de corto plazo y se reconoce que la última, no siempre posible, es la que verdaderamente representa solidez y previsibilidad futura.

En efecto, en situaciones donde por diferentes motivos se descartan las opciones de reconstrucción y de encapsulamiento, la migración del sistema a un ambiente abierto se convierte en la mejor alternativa. Si bien esta es la opción más compleja, las ventajas que se obtienen a largo plazo justifican ampliamente el esfuerzo que será requerido.

Aquí debe reconocerse que un trabajo de migración es normalmente un proyecto de ingeniería de sistemas, que por su importancia merece el calificativo de crítico. Esto es así tanto por la relevancia de los entornos migrados (datos y aplicaciones), que deberán ofrecer finalmente la misma eficiencia y operatividad que ofrecían en el entorno anterior, como así también por la necesidad de hacer mínimo el impacto en todos los niveles de la organización. Se hace referencia aquí al objetivo de enfrentar un cambio de cultura tecnológica, para el que habrá que prever recursos técnicos y humanos, y que deberá ser acompañado del necesario entrenamiento del personal y usuarios.

Además, durante el proceso de cambio del sistema será muy importante prever cuál será la gestión de su evolución posterior; con el fin de evitar que la situación presente vuelva a repetirse o al menos resulte menos traumática. La gestión de la evolución debe consistir en el ofrecimiento de una respuesta rápida, preparada y eficiente a los cambios que se produzcan en el entorno, ya sean de índole tecnológica o de gestión del propio negocio.

2.3. Estrategias de migración

Las estrategias de migración reconocen los dos enfoques siguientes:

a) Habilitación gradual

La nueva aplicación es construida gradualmente en la plataforma de destino, haciéndose cargo en forma progresiva de las funcionalidades de la aplicación original, por lo que en este proceso ambas aplicaciones están integradas en un

único sistema con una transferencia gradual de responsabilidades de una a otra. Con este enfoque la información está duplicada y es necesario un importante esfuerzo de coordinación para asegurar la integridad y consistencia de los datos.

b) Habilitación súbita

La aplicación original mantiene todas sus prestaciones mientras la aplicación en la nueva plataforma es construida, implementada y probada. Las bases de datos de esta última son progresivamente actualizadas hasta el momento en que se decide la transferencia del control, momento en que la aplicación original queda desafectada y sus bases de datos quedan como referencia únicamente para consulta.

Se debe tener en cuenta que antes del desarrollo del nuevo sistema, es imprescindible tener una comprensión intensiva del sistema a ser migrado.

En cualquier sistema a ser migrado, algunas características son comunes con todo proyecto de ingeniería de software, tales como metodología de desarrollo, testing y selección del modelo de bases de datos. Otras, son específicas de la migración, por lo que se puede clasificarlas en dos grandes categorías: aquellas que conciernen al sistema a migrar, y, las específicas del sistema migrado, para lo cual es necesario entender las características intrínsecas de los datos, las interfases y las aplicaciones involucradas, en cualquier proceso de migración.

Consecuentemente, antes de tomar cualquier decisión sobre la estrategia de migración, se debe realizar un estudio intensivo a los efectos de cuantificar los riesgos y beneficios, con el fin de justificar acabadamente la migración a un nuevo sistema, según lo proponen Espiñeira y Sheldon (2005).

2.4. Los pilares de todo el proceso de migración

Una migración debe apoyarse en tres pilares básicos, a saber: 1) una metodología, 2) un conjunto de herramientas y 3) técnicas de pruebas y personalización.

La ***metodología*** garantiza, en primer lugar, un procedimiento sistemático que asegura que el trabajo realizado sea controlable y sus resultados predecibles. En segundo lugar, que se dispone de un repositorio con toda la información necesaria para abordar la migración: cadenas de programas, programas fuente, estructura de bases de datos, librerías de funciones, etc. En tercer lugar, contempla la obtención del modelo de negocio a migrar, a partir de la información contenida en el repositorio, y considera además la realización de los planes de prueba de las aplicaciones migradas. Por último, define las reglas de generación del código migrado, conforme a los estándares establecidos, las librerías de funciones usadas y cualquier otra consideración de interés.

Las ***herramientas de migración*** permiten obtener un modelo del negocio a migrar, que lo hace independiente de los lenguajes de las aplicaciones, con lo cual el modelo obtenido resultará válido en caso de ser necesarias futuras migraciones a otras tecnologías. Estas herramientas deben permitir, también, la incorporación de las reglas básicas del negocio a los efectos de obtener aplicaciones optimizadas para su funcionamiento en el entorno informático existente en una empresa.

Las ***técnicas de pruebas y personalización*** incorporan las reglas de generación introducidas por la metodología a los fines de obtener aplicaciones funcional y operativamente fiables y las optimizan para su funcionamiento en el entorno informático existente en la empresa.

La utilización de estos tres pilares permite asegurar el éxito del proyecto, manteniendo los plazos y costos de realización dentro de las previsiones.

2.5. La gestión de la evolución

La evolución de un sistema es un concepto amplio; abarca desde una simple modificación para corregir un error de un programa hasta una reimplantación completa del mismo. Con el paso del tiempo, se producen tres cambios en el desarrollo de sistemas que afectan a los aspectos de su evolución:

- La importancia del reflejo de la arquitectura en la documentación del sistema.
- La aparición del paradigma de objetos.
- Los componentes distribuidos.

Aunque en un principio se pensó que el paradigma de objetos sería la solución al problema de los sistemas heredados, se descubrió pronto que los problemas se agravaban al adaptar sistemas no concebidos para este paradigma a lenguajes como el C++, mediante el mal uso de mecanismos tales como la encapsulación y, sobre todo, de la herencia. Al fin y al cabo, se repite el mismo problema de siempre: el paradigma puede tener beneficios si al desarrollarlo se aplican técnicas de ingeniería. En caso contrario, vuelven a aparecer, nuevamente, los problemas del mantenimiento del sistema.

En el caso de desarrollos orientados a objetos, la falta de experiencia del personal, el empleo de varios lenguajes para el desarrollo de un mismo sistema y la no incorporación de nuevas tecnologías (UML, CORBA), proporcionan una base errónea para la construcción de sistemas con las mismas deficiencias que los actuales. Se presenta un nuevo tipo de sistemas heredados a los cuales es necesario someter a técnicas de reingeniería para su mantenimiento y evolución.

Como en los sistemas heredados tradicionales, a la hora de obtener la arquitectura del sistema los problemas se producen por una documentación insuficiente, falta de modularidad con un alto grado de acoplamiento entre clases y funcionalidad duplicada en diferentes implementaciones. La falta de experiencia en la construcción de programas produce una nula o mala utilización de la herencia, operaciones que se definen fuera de la clase correspondiente, violación de la encapsulación y clases mal utilizadas (escribir C++ con estilo C). Para que pueda ser aplicada una reingeniería orientada a objetos, debe cumplir con los requisitos: diseño independiente del lenguaje de implementación, desarrollo escalable y herramientas que soporten las técnicas aplicadas.

Ahora bien, a los fines de decidir el proceso de migración del sistema es necesario descubrir el núcleo del mismo (funcionalidades y datos), utilizando análisis para averiguar el nivel de abstracción necesario y la técnica más adecuada. En un sistema puede aplicarse esta técnica si se cuenta con un mínimo de recursos, entre los que deben figurar las descripciones de su arquitectura, modelos de dominio, documentación del diseño, programas de prueba, datos de prueba y su documentación, especificaciones de la interfase, herramientas, código y procesos. Es importante conocer los detalles de la arquitectura y del diseño, además de las restricciones de ingeniería y del dominio de aplicación.

El planteamiento de la evolución de los sistemas debe hacerse tanto para sistemas reingeniados como para los nuevos desarrollos, según lo plantean Bisbal et al (1997). Conceptualmente se debe preservar la estructura del sistema incorporando los cambios del entorno. Para ello, resulta necesario establecer un marco de trabajo disciplinado capaz de incorporar los cambios en el sistema en el momento más adecuado, que preserve su arquitectura y asegure la transmisión continua de conocimiento. Esto requiere un fuerte compromiso entre la ingeniería del software y la ingeniería de negocio de la organización y, posiblemente, un cambio en el entorno académico para asumir la inclusión en los currículos de los nuevos paradigmas.

2.6. Interrogantes para migrar a la Web

A continuación se presentan algunos de los interrogantes que se debe plantear una organización antes de realizar una migración de una aplicación Cliente/Servidor a la Web, agrupados según los distintos aspectos con los que éstos se relacionan.

a) Metas

- ¿Cuál es el objetivo y su motivación?, ¿es una necesidad o simplemente un deseo?
- ¿Qué debe realizar el sitio Web?
- ¿Cómo interactuará el sitio Web con las aplicaciones existentes?, ¿a través de procesos, de datos u otro tipo de integración?
- ¿Cuál será el futuro de las aplicaciones existentes?

b) Diseño Web

- ¿Cuál es la apariencia prevista para el sitio?
- ¿Tienen los elementos de diseño gráfico un gran impacto sobre el negocio? ¿Se requiere contenido estático o dinámico?
- ¿Quiénes pueden acceder al sitio?
- ¿Cuáles son los requerimientos de seguridad?
- ¿Cómo es el flujo de las páginas Web?
- ¿Se harán ingresos de datos o sólo reportes?

c) Recursos

- ¿Cuál es el presupuesto necesario?
- ¿Con qué soporte organizacional se debe contar?
- ¿Cuál es el nivel de conocimientos que deben poseer los programadores?, ¿requieren entrenamiento?
- ¿Es el entrenamiento un objetivo organizacional?

d) Técnico

- ¿Cuál es el sistema operativo para el servidor?
- ¿Qué servidor de aplicaciones se debe usar?
- ¿Qué servidor Web se debe usar?
- ¿Qué lenguaje de programación utilizar?
- ¿Cómo es el nivel de conocimiento de los programadores, nuevo o con experiencia existente?

e) Perspectivas de negocio

- ¿Se prevén cambios para los usuarios existentes?
- ¿Quiénes serán los nuevos usuarios?
- ¿Existen nuevos requerimientos?

- ¿Cómo es el impacto de los nuevos requerimientos sobre las características anteriores?

Fundamentalmente, una de las principales razones que se esgrimen para migrar a la Web la constituye el hecho de que los sistemas y las aplicaciones basados en Web hacen posible que una gran cantidad de usuarios pueda acceder a las mismas independientemente del lugar donde se encuentre.

Así, cuando los sistemas crecen en funcionalidad, y los usuarios que acceden a los mismos también, es impensable hacer frente a estos desafíos con los sistemas distribuidos tradicionales. Es necesario, no obstante, tener en cuenta los interrogantes planteados anteriormente para poder realizar el proceso de migración de estos sistemas a la Web, siguiendo un enfoque disciplinado a los efectos de la construcción de una arquitectura sólida que pueda ser eficientemente mantenible y configurable en su evolución.

2.7. Problemas con la arquitectura Cliente/Servidor

Una vez que se ha logrado, de acuerdo a la metodología aplicada, realizar una buena especificación de requerimientos para el sistema a migrar, debe considerarse prioritaria la selección de una buena arquitectura para el mismo.

El objetivo de esta nueva arquitectura es el de facilitar el mantenimiento y posibilidad de escalabilidad del nuevo sistema, de modo que el mismo no se transforme solamente en una extensión del sistema cliente-servidor.

La arquitectura cliente servidor intenta equilibrar el proceso de una red entre computadoras especiales como son los servidores y, aquellas que envían, a través de una interfase gráfica de usuario (GUI) consultas a una base de datos que se encuentra en un servidor, y que se visualizan a través de la interfase. Generalmente, cuando la red que soporta esta arquitectura distribuida es una red de área local (LAN), la lógica de la aplicación cliente reside en cada estación de trabajo de acuerdo al perfil del mismo, por eso se lo denomina FAT CLIENT, o cliente pesado.

Se mencionan a continuación algunos de los problemas más comunes encontrados en las aplicaciones distribuidas tradicionales:

- Programación para un solo cliente (Windows).
- No está preparado para la Web.
- Control no centralizado.
- Generación de cuellos de botella en la base de datos.
- Consume mucho recurso.
- Limitado a recursos de hardware.
- Código embebido en los objetos.
- Falta de control de las conexiones a las bases de datos.
- Los clientes tienen administración del negocio.
- Fallas en la seguridad.

Asimismo, cabe mencionar que al momento de recoger los datos y las aspiraciones del cliente durante la fase del estudio preliminar surge un punto de decisión en el que resulta necesario considerar diferentes aspectos referentes a las aplicaciones a migrar, tales como:

- Lenguajes de programación de las aplicaciones.

- Organización de los datos.
- Expectativas de evolución de la aplicación.
- Frecuencia e importancia de los cambios futuros.
- Necesidad de modernización y agilidad ante futuros cambios.
- Existencia de productos de emulación en la plataforma abierta.

Ahora bien, y tal como ya fue mencionado, según el factor que se considere existen dos alternativas posibles:

- Reubicación de la aplicación en la nueva plataforma utilizando productos de emulación.
- Transformación/migración de la aplicación a un nuevo entorno de programación.

Como el nuevo entorno de programación para la Web exige un conocimiento profundo de nuevas tecnologías, es necesaria una previa capacitación de los recursos humanos disponibles, así como la adquisición de nuevo hardware (servidores y Workstations) para poder efectuar un desarrollo acorde a las exigencias de las *NTIC*'s (nuevas tecnologías de la información y las comunicaciones).

2.8. Posibles soluciones

Las *NTIC*'s exigen primariamente entornos distribuidos seguros donde muchos usuarios puedan acceder simultáneamente y eficientemente a una diversidad de recursos de datos y aplicaciones.

A los fines de solucionar todos estos problemas, se presentan en el mercado varias alternativas. A continuación, se detallan algunas de ellas:

- Software emulador, que permite la conexión a los aplicativos, pero sigue siendo Cliente/Servidor.
- Software para migrar los aplicativos Cliente/Servidor a la Web, para algunos casos son buenos; pero generan cajas negras y además código innecesario porque la interfaz Windows es diferente a la Web, y los costos son elevados.
- Adquirir un producto en Web llave en mano, lo que crea dependencia de la empresa.
- Realizar desarrollo desde cero con otras herramientas.
- Software de seguridad para permitir el "acceso seguro" desde varias aplicaciones clientes a las "aplicaciones servidoras" a través del broker existente en la/s máquinas clientes.

En síntesis, una sesión típica de migración debe comprender:

- Análisis de las plataformas, aplicaciones e información que son el objeto de la migración.
- Formalización de reglas de conversión.
- Uso del "aplicador de reglas". La documentación generada sirve para añadir nuevas reglas al sistema y puntualiza los sitios donde se deben hacer las transformaciones semánticas.

- Análisis de la información trasformada, mediante el uso de herramientas para garantizar una migración que aproveche mejor los recursos disponibles en la nueva plataforma.
- Considerar la necesidad de una buena fase de testeo del software a implementar, teniendo en cuenta varios factores tales como funcionalidad, cambios en el hardware, en el sistema operativo y, en toda otra tecnología que fue pasible a modificaciones en el sistema migrado.

En cualquiera de estas alternativas se plantea la necesidad de una metodología para el proceso de migración, que considere al menos las siguientes etapas fundamentales:

1. Justificación del por qué de la migración
2. Comprensión exhaustiva del sistema a migrar
3. Desarrollo del nuevo sistema
4. Testing
5. Implementación de la Migración

En todas estas etapas debe considerarse seriamente la posibilidad de "reutilizar" todos los artefactos de software disponibles (requisitos, documentación, pruebas, etc.) a los efectos de minimizar los recursos asignados y ampliar la brecha con los beneficios obtenidos por el proceso de migración.

2.9. Beneficios del proceso de migración

Es esencial que para el éxito del proceso de migración, se cumpla con la funcionalidad requerida dentro del dominio de aplicación establecido, para lo cual el usuario debe comprender el alcance de la misma y entender que el sistema anterior satisfacía parcialmente los requerimientos especificados e implementados para el nuevo sistema migrado.

De esta forma, los costos involucrados en el proceso de migración deben ser sopesados contra los beneficios logrados, teniendo en cuenta además una estimación de la posibilidad de fallas durante el desarrollo y la implementación del mismo.

Entre los principales beneficios asociados al proceso de migración, cabe citarse:

- Reducción de costos. En una arquitectura Web, las tareas de administración y mantenimiento del software se realizan en un solo punto y no en cada uno de los clientes.
- Mejora de la productividad: un entorno más amigable tanto para los desarrolladores como para los usuarios y el uso de nuevas funcionalidades.
- Mayor accesibilidad: posibilidad de integración con portales corporativos con el único requisito de disponibilidad de un navegador o un dispositivo wireless.
- Se puede acceder en este caso, en tiempo real, a información y herramientas antes sólo disponibles para minorías a través de terminales específicos. Gracias a la tecnología Web el acceso se realiza a esos mismos sistemas desde cualquier terminal a través del navegador.
- Mantenimiento de la inversión: se conservan y reutilizan los conocimientos esenciales de los desarrolladores y usuarios sobre los actuales desarrollos, por lo que el proceso de migración aprovecha al máximo las capacidades existentes.
- Posibilidad de reutilización del código actual y de la documentación existente a la migración.

Por último puede citarse la integración de los sistemas migrados a la Web con los otros sistemas o aplicativos de los usuarios en línea y los recientes servicios ofrecidos por la Web 2.0 (wikis, blogs, foros, ecommerce, etc).

CAPÍTULO 3. TESTING EN LA WEB

3.1. Introducción

En la última década se asistió a un profundo cambio en la operatividad de los sistemas de computación. En efecto, las aplicaciones convencionales de escritorio se caracterizaban por ser estáticas y fuertemente dependientes de bibliotecas especiales en su propio entorno de ejecución. Esto se ha venido modificado, en gran medida debido a la fuerte tendencia hacia las aplicaciones Web, que se caracterizan por ser desplegables en un browser o navegador, por su naturaleza dinámica y por estar fuertemente apoyadas en estándares que permiten su visualización en forma independiente. Paralelamente, se viene comprobando un crecimiento vertiginoso de desarrollos y uso de aplicaciones y sistemas Web cada vez más complejos y sofisticados.

Debe reconocerse que son diversos los cambios de enfoque entre estos dos tipos de aplicaciones, que abarca desde la manera en que se desarrollan hasta la forma en que se prueban y validan. No obstante, la gran difusión y mayor complejidad de estas nuevas aplicaciones no fue acompañada de los mecanismos adecuados para garantizar la calidad de los sistemas, a pesar de la creciente dependencia a nivel social y económico que se tiene de ellos.

Es así que la escasa calidad en las aplicaciones basadas en la Web ha venido generando una preocupación creciente entre los miembros de la comunidad científica y técnica involucrada en el desarrollo de estos sistemas. Así pues, en los últimos años surgieron varias iniciativas con el objetivo de poner cierto orden, y de las discusiones sostenidas en congresos y talleres especializados ha surgido una nueva disciplina denominada *Ingeniería Web*.

Murugesan (2001) fue uno de los primeros en promover el reconocimiento de una nueva disciplina asociada a la Web y definir la Ingeniería vinculada a ella: "*La ingeniería Web consiste en el establecimiento y uso de principios de ingeniería y administración sólidos y científicos, y enfoques disciplinados y sistemáticos para el desarrollo exitoso, la implementación y el mantenimiento de alta calidad de sistemas y aplicaciones basados en la Web*". Esta definición reconoce la necesidad de un nuevo campo de la ingeniería de software, que conducirá a un enfoque ordenado y sistemático que permitirá aplicar las técnicas y herramientas más adecuadas para desarrollar aplicaciones Web robustas y también disponer de los medios apropiados para su evaluación.

Para comenzar, debe aquí reconocerse que el desarrollo de aplicaciones Web posee determinadas características que lo distinguen del desarrollo de aplicaciones tradicionales. En efecto, en estas aplicaciones cobran gran importancia aspectos tales como el uso intensivo de lenguajes interpretados, la interfase con el usuario, el diseño gráfico, la animación, la administración de hipertextos y la dinámica del despliegue de las aplicaciones, entre otras. A estas deben sumarse otros aspectos más convencionales, como son el modelado y diseño de estos sistemas, su simulación, las técnicas de administración y recuperación de información y la siempre necesaria gestión de proyectos, todo lo cual hace a la Ingeniería Web una disciplina verdaderamente multidisciplinaria.

Acorde a esto, los procesos de testing que deben ser aplicados en todo el ciclo de desarrollo de estos sistemas no son una simple extensión de los métodos tradicionales de testing. Por el contrario, se trata de procesos con identidad propia que deben contemplar tanto los aspectos particulares como los convencionales ya antes mencionados de estos desarrollos.

Llegado a este punto, parece conveniente definir el concepto de *testing,* como el proceso de evaluación de un producto de software, tanto para comprobar que opera

correctamente (verificación) como también para asegurar que sus prestaciones son acordes a sus especificaciones (validación).

En este Capítulo se introduce el problema del testing de aplicaciones Web, revisando las condiciones que permiten verificar y validar sistemas en diferentes entornos, identificando los principales tipos de testing y presentando finalmente las condiciones necesarias para desarrollar estos procesos.

3.2. Naturaleza de los desarrollos Web y sus consecuencias en el Testing

Naturalmente, las estrategias para la evaluación de las aplicaciones Web adoptan los principios básicos establecidos para las pruebas del software convencional, aplicándose técnicas utilizadas en los sistemas orientados a objetos. Sin embargo, teniendo en cuenta que las aplicaciones Web residen en una red y operan con distintos sistemas operativos, navegadores, plataformas de hardware y protocolos de comunicación, los procesos de testing deben incorporar otros recursos para adecuarse a este nuevo entorno de trabajo.

Más aún, debe considerarse que no solo se trata de productos diferentes, sino que se llega a ellos a través de procesos que son también diferentes. Con respecto a esto último y para marcar las diferencias entre las fases de un desarrollo clásico y el de una aplicación Web resulta muy ilustrativa la tabla propuesta por Andrea Macintosh (2000), que se presenta a continuación (Tabla 1):

Tabla 1 - Diferencias entre las fases del Ciclo de Vida de desarrollo de software de una aplicación clásica y una aplicación Web.

Actividad	Aplicación clásica	Aplicación Web
Definición de Requerimientos	Basado en detalladas especificaciones.	Las especificaciones no son estrictas. Se basan en notas, discusiones o ideas.
Planeamiento	Basado en experiencias de proyectos anteriores.	Basado en la disponibilidad de tiempo.
Análisis y Diseño	Procesos separados formalmente.	Procesos que se efectúan en paralelo.
Implementación	Desarrollo secuencial de componentes.	Procesos iterativos sobre sitios Web.
Integración	Ensamblado de componentes	Habilitación progresiva de servicios Web.
Evaluación	Se prueba la funcionalidad en contra las especificaciones	Se basa preferentemente sobre la funcionalidad deseada.
Puesta en régimen	Transferencia de información a un soporte de instalación.	Transferencia a Web Server.
Mantenimiento	En promedio, 1 a 3 años.	En promedio, 4 a 6 meses.

Como puede apreciarse, las fases del Ciclo de Vida de un desarrollo Web presentan características semejantes a las propuestas por la mayoría de las llamadas metodologías ligeras, como es el caso del Extreme Programming (XP). Una de las diferencias sustanciales entre ambos enfoques se da en las fases de análisis y diseño, donde en las aplicaciones clásicas estas fases están conceptualmente y formalmente separadas, mientras que en el marco de una aplicación Web éstas son tareas paralelas, con un enfoque que podría identificarse como evolutivo. Por otra parte, los ciclos de vida de las aplicaciones Web son sustancialmente más cortos que las aplicaciones clásicas y esto queda reflejado en los períodos de mantenimiento.

A efectos de completar el enfoque comparativo de la tabla anterior, pareció conveniente resumir los conceptos de Robert Glass (2000) relativos a las diferentes fases del ciclo de vida del software, que son presentados en la Tabla 2:

Tabla 2 – Algunas características de las fases del Ciclo de Vida de desarrollo de software de una aplicación Web.

Actividad	Desarrollo Web
Definición de Requerimientos	Una aplicación Web posee requerimientos livianos, debido a que cualquier requerimiento se restringe a las posibilidades de un browser.
Planeamiento	Los proyectos Web son cortos. En promedio, un proyecto Web no demora más de tres meses y está a cargo de un equipo de cuatro personas.
Implementación	El desarrollo de una aplicación Web es rápido y poco granular. La cantidad de herramientas de desarrollo Web existentes proveen un desarrollo simple y rápido, dejando de lado los editores de texto y la edición manual de código HTML. El desarrollo de una aplicación Web responde al concepto de "*rapid development*" o desarrollo rápido, donde se prima un desarrollador rápido por sobre profesionales del software.
Evaluación	El testing de una aplicación Web nunca es lo suficientemente minucioso, aun cuando existen herramientas que automatizan el proceso. Estas herramientas son de buena calidad, pero su elevado costo restringe su utilización. En la práctica, un proyecto Web se encuentra siempre en proceso de testing. Esto es debido a su naturaleza dinámica y a que resulta muy simple modificar programas escritos en lenguaje interpretado, como un script o código HTML. El testing debe incluir técnicas que determinen la aceptación visual por parte del usuario, ya que una aplicación Web es altamente orientada al contenido y la visualización.
Operación	Una aplicación Web no establece una interacción directa entre el cliente y el servidor. No se posee una certeza de quién es el cliente que está interactuando con el sistema.

También debe advertirse que el término genérico de "aplicación Web" engloba un espectro muy amplio de posibilidades, mostrándose en la Tabla 3 algunas de las principales características de las dos posibles situaciones extremas:

Tabla 3 – Principales aspectos que distinguen las aplicaciones Web simples y avanzadas

Aplicación Web Simple	Aplicación Web Avanzada
Contenidos mayoritariamente estáticos, representados por información textual.	Grandes volúmenes de información, heterogénea, representada dinámicamente y cambiante en el tiempo.
Objetivo esencialmente informativo.	Medio principal de comunicación entre una organización y sus usuarios.
Mínimos requerimientos de seguridad.	El análisis de riesgos y la seguridad son una condición principal de diseño y operación.

Interactividad y accesibilidad limitada.	Integración con bases de datos y otros sistemas de identificación y búsqueda de datos.

Obsérvese que, a pesar de todo lo expuesto, la comprensión de la naturaleza de los desarrollos Web requiere aún de otras consideraciones, algunas de ellas insospechadas, que van más allá de la mera comparación de productos y procesos. Puede citarse como ejemplo a la naturaleza de la información procesada, que en los sistemas clásicos está mayormente representada por datos numéricos operados en forma transaccional, con relativamente escasa información textual. Esto facilita su estructuramiento, normalización, clasificación y búsqueda. Por el contrario, los sistemas de información Web operan con mayor cantidad de información textual y gráfica, a lo que se agrega el video y audio, que la hace mucho más difícil de estructurar, normalizar y clasificar.

Todo esto permite anticipar que cada fase del Ciclo de Vida de las aplicaciones Web da lugar a evaluaciones que involucran nuevos factores a probar y la necesidad de disponer de nuevas técnicas para realizar estas pruebas.

3.3. Prerrequisitos y Planificación del Web testing

Previo a dar comienzo al proceso de testing, resulta necesario tener en cuenta una serie de acciones tales como:

1. Reunir el equipo de testing.
2. Preparar documentación que incluya: requerimientos que fueron aceptados por el cliente y el equipo de testing, el diseño funcional completo y especificaciones internas del diseño, entre otros.
3. Crear un plan de proyecto, con la identificación de los aspectos de alto riesgo.
4. Diseñar del ámbito del testing, en el cual se determina los tiempos para cada fase así como una priorización de los elementos a ser probados.
5. Verificar el ambiente de testing: hardware y software.
6. Crear test scripts.
7. Desarrollar un método para el seguimiento de un problema.
8. Decidir el número de iteraciones a realizar.

El plan de testing de un proyecto software es un documento propuesto por ANSI/IEEE Standard 829-1983 (ver Anexo III), en donde se describe el proceso de testing. Los elementos más importantes de este plan son:

- Establecer los objetivos para cada fase del testing
- Establecer las responsabilidades de cada actividad del testing
- Determinar la disponibilidad de las herramientas, facilidades bibliotecas para el testing
- Establecer los procedimientos y estándares a ser utilizados en el planeamiento y conducción del testing y el reporte de los resultados
- Establecer tanto el criterio de finalización del testing como el criterio de éxito para cada test.

La estructura de un plan de Web testing es la siguiente:

1. Introducción
2. Objetivos y tareas

3. Riesgos
   3.1. Identificar y priorizar áreas de riesgo: por ejemplo, cuando los casos de prueba no sean documentados.
   3.2. Identificar dependencias de pruebas sobre actividades externas: en caso de estar relacionadas con hitos del mismo proyecto.
   3.3. Planes de contingencia para los riesgos: cursos de acción a seguir en caso de problemas.
4. Estrategia de testing
   4.1. Testing funcional: contempla aspectos como el procedimiento para diseñar casos de prueba, definir condiciones y validación de estos casos.
   4.2. Testing de GUI: es necesario considerar aspectos como: botones, menús, toolbars, formularios, fuentes, etc.
   4.3. Testing de usabilidad: aspectos navegacionales, ayuda, links, desempeño del sistema, seguridad, entre otros.
5. Requerimientos de hardware y software: en cuanto browser, sistemas periféricos, software antivirus, etc.
6. Requerimientos del entorno: principalmente se refiere a realizar pruebas en dos tipos de workstation: primero en aquellas que no poseen ningún software aparte de la aplicación que se desea probar y segundo, en otras workstation donde existan otros tipos de aplicaciones, de manera de probar las interacciones entre ellas.
7. Planificación de pruebas y del proyecto
8. Recursos: humanos, financieros y temporales.
9. Procedimiento de pruebas: tipos de estrategia a seguir
10. Procedimiento de control
11. Resultados de pruebas
12. Aprobación: criterios de aceptación

3.4. Estrategias de prueba convencionales

Las pruebas constituyen el último eslabón desde el cuál se puede evaluar la calidad del software e identificar errores, por lo que a esta etapa muchos autores la consideran “destructiva”. Sin embargo, se debe tener en cuenta que la calidad se incorpora en el software durante todo el ciclo de vida del mismo.

Por lo tanto, es posible equiparar el proceso de prueba del software bajo la misma estrategia que el proceso de ingeniería del software representado por el modelo de espiral.

Así, como se muestra en la figura 1, la *prueba de la unidad* comienza en el vértice de la espiral y se concentra en cada unidad del software, avanza hasta desplazarse hacia afuera, a lo largo del espiral, hasta llegar a la *prueba de integración*, donde se realiza el diseño y la construcción de la arquitectura del software. Una vuelta más afuera de la espiral se efectúa la *prueba de validación*, donde se validan los requisitos establecidos en el análisis de requisitos del software, comparándolos con el software construido.

Finalmente se llega a la etapa de *prueba del sistema* donde se prueban como un todo el software y otros elementos del sistema.

En todo este proceso se recorre el espiral hacia fuera, de forma tal que en cada vuelta se ensancha el alcance de la prueba. Al comienzo, la prueba se realiza en cada componente individual, asegurando que cada unidad funciona de manera apropiada (prueba de unidad), a continuación deben integrarse y ensamblarse todos los componentes para formar el paquete de software a probar (prueba de integración).

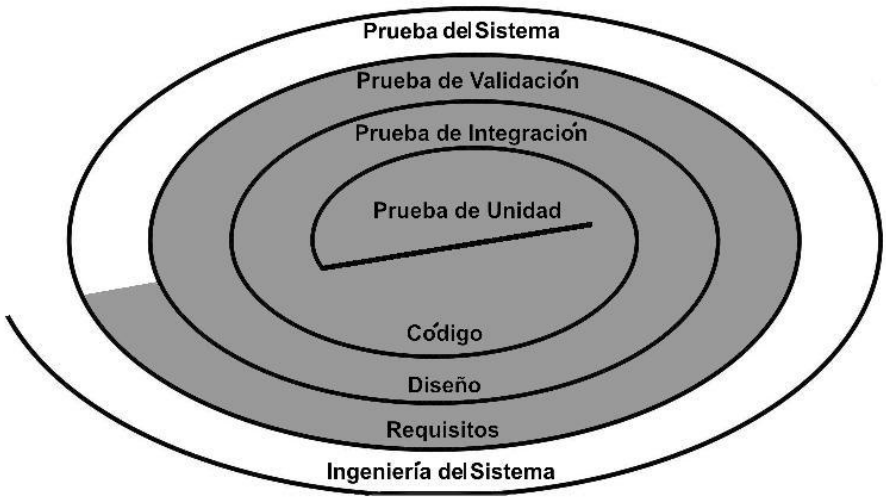


Figura 1– Proceso de pruebas en espiral

Una vez que se ha integrado el software se aplica un conjunto de pruebas de alto nivel para evaluar los criterios de validación establecidos durante el análisis de requisitos (prueba de validación). Una vez validado el software, debe combinarse con otros elementos del sistema (por ejemplo, hardware, personas, bases de datos), para verificar que todos los elementos encajen apropiadamente a los fines de lograr el buen desempeño del sistema.

En general, la mayoría de los equipos de software elige este enfoque incremental de la prueba iniciado con las pruebas de unidades individuales, pasando por las pruebas diseñadas para lograr la integración de las unidades y, finalizando con pruebas que se realizan sobre el proceso construido.

A continuación se desarrollan cada uno de estos tipos de prueba fundamentales.

3.4.1. Testing de unidad

Es la de más bajo nivel. Centra el proceso de verificación en la unidad más pequeña del diseño de software: el módulo.

Para ello se debe tener en cuenta la lógica del procesamiento interno del componente y en las estructuras de datos dentro de los límites del mismo. Esto significa que se prueban las condiciones límites para asegurar que todas las instrucciones del módulo se hayan ejecutado al menos una vez. Se deben diseñar casos de prueba para descubrir errores originados por cálculos incorrectos, comparaciones erróneas o flujos de control inadecuados. Consiste básicamente en una prueba estructural o de caja blanca, lo cual requiere conocer el diseño interno de la unidad, y de una prueba de especificación, o de caja negra, basada en el comportamiento externamente visible de la unidad.

Normalmente ambas pruebas son necesarias y se complementan entre sí, y se pueden llevar a cabo en paralelo para múltiples módulos.

Las pruebas que se realizan como parte de la prueba de unidad comprenden: interfaces de los módulos, estructuras locales, condiciones límites, caminos independientes y manejo de errores.

Como en general un componente no es un programa independiente, suelen utilizarse programas controladores para aceptar los datos de ingreso, enviarlos al componente y mostrar los resultados, y programas de resguardo para reemplazar los módulos subordinados al componente que se está probando. En muchos casos se evita utilizar estos programas, que originan una sobrecarga de trabajo, posponiéndolos para la prueba de integración. En este caso, se simplifica la prueba de unidad y, se reduce el número de casos de prueba, atendiendo sólo una función del componente.

Respecto al manejo de errores, no basta con recargar los componentes para que realicen esta tarea. Muchas veces la descripción del error no es clara o no corresponde al error encontrado, puede causar la intervención del sistema operativo antes de que se dispare el manejador de errores, o el procesamiento de la condición de error es incorrecto.

Yourdon (Yourdon, 1975), propone que se configuren rutas de manejo de errores o en su defecto, que se finalice el procesamiento cuando ocurra un error. Cuando todos los componentes hayan sido probados individualmente, y, corregidos los errores encontrados, puede realizarse la siguiente prueba que es la de integración.

3.4.2. Testing de integración

El objetivo de esta prueba es tomar los módulos que han sido probados como unidades, para probar las interacciones entre ellas y construir una estructura del programa de acuerdo a lo estipulado por el diseño. Existen varios tipos de estrategias para realizar esta integración:

- Big Bang: se combinan todos los módulos por anticipado y se prueba el programa en conjunto. La corrección es difícil, puesto que es complicado aislar las causas al tener que analizar el programa completo. Se descubren errores que no estaban expuestos en las pruebas más simples, y el proceso continúa en un ciclo que parece interminable.
- Incremental: es la antítesis del enfoque del “big bang”, donde el programa se construye y prueba en pequeños incrementos, en los cuales resulta más fácil aislar y corregir los errores, ya que es probable que se prueben por completo las interfaces y por lo tanto puede aplicarse un enfoque sistemático para las pruebas.

  Existen dos enfoques dentro de la estrategia incremental que reflejan distintos enfoques de la integración del sistema: descendente y ascendente.

  En la integración descendente, los componentes de los niveles más altos del sistema se integran y prueban antes de que se complete su diseño e implementación. Los módulos se integran al descender por la jerarquía de control, empezando por el módulo principal, y los módulos subordinados se incorporan en la estructura de dos maneras posibles, primero en profundidad o primero en anchura. La primera opción integra todos los módulos subordinados a una ruta de control elegida de antemano, mientras que la segunda incorpora todos los componentes directamente subordinados en cada nivel, desplazándose horizontalmente por la estructura.

  En la integración ascendente, los componentes de los niveles bajos se integran y prueban antes de que se desarrollen los componentes de los niveles altos. Este

último enfoque no requiere que el diseño esté completo, por lo que se puede comenzar en una etapa inicial del proceso de desarrollo del sistema.

Por lo general, los sistemas se prueban utilizando una combinación de ambos tipos de pruebas, que se conoce como prueba sandwich.

La prueba de integración, cualquiera sea el enfoque utilizado, se basa primordialmente en la prueba de casos de uso, mediante la cual se pueden identificar los estímulos entre el usuario y el sistema, y, entre los objetos del sistema. Los casos de uso que extienden o son incluidos en otros casos de uso, se prueban después de verificar los casos de uso básicos donde éstos se insertan.

A través de las pruebas de integración se verifica que las unidades de software operan correctamente cuando se combinan en la aplicación. Normalmente estas pruebas se van realizando por etapas, englobando progresivamente más y más módulos en cada prueba. Pueden plantearse desde un punto de vista estructural o funcional.

Las interacciones entre casos de uso es una de las áreas más difíciles de probar. Ello se debe al enorme número de combinaciones generadas por un caso de uso, incluso de tamaño moderado. Por esto, es casi imposible realizar pruebas exhaustivas de las interacciones entre casos de uso. Sin embargo, se pueden utilizar ciertas técnicas para encontrar áreas en las que es probable que la interacción entre casos de uso provoque problemas. El área más fácil es la precondición del caso de uso. Si el caso de uso tiene una determinada precondición, ¿hay algún caso de uso que deje al sistema en un estado en que se viole la precondición? Hay dos formas de que esto ocurra. La primera es cuando se saltea un paso en el proceso. El segundo escenario es en el que el sistema puede ingresar en un estado extraño, es decir, cuando se cumple una condición de excepción. Unir los casos de uso que presenten estos tipos de problema es una manera simple de probar las interacciones. Una vez que están unidos los casos de uso, hay que seleccionar las suites de prueba correspondientes a los casos de uso. Esta propuesta se muestra en la figura 2.

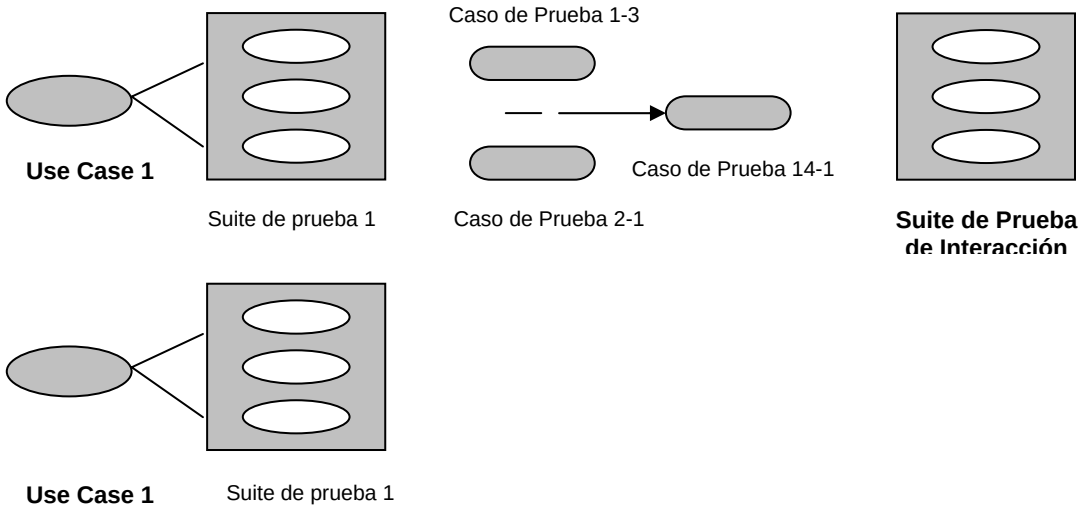


Figura 2 - Interacciones entre casos de uso

En la figura anterior, se observa un escenario con dos casos de uso, se busca en qué caso de uso se viola la precondición de prueba del caso de uso anterior, y al localizarlo, se unen los dos casos de uso para probar su interacción. Un ejemplo en un sistema de facturación:

- Al validar el cliente para facturar con cuenta corriente se dio de baja la cuenta corriente.

- El producto seleccionado no existe porque NO fue dado de alta.

Existen otras formas de interacción aparte de las precondiciones que necesitan ser probadas. Una técnica es construir tablas de doble entrada (Matriz CRUD) entre casos de uso y objetos. Si se combinan estas matrices, se puede ver cómo los casos de uso se relacionan entre sí mediante los objetos que tocan.

Específicamente, se buscan situaciones en las que el objeto es leído, actualizado o borrado en un caso de uso y creado en otro. ¿Qué pasa cuando el objeto no existe? Esta condición se llama "Leído antes de Creado" (RC). Otra condición que se busca es cuando un caso de uso elimina un objeto que otro caso de uso está leyendo o actualizando. Esta condición se llama "Leído después de Borrado" (RD). Los casos de prueba de interacción son creados combinando los casos de prueba funcionales que involucran RCs y RDs., como puede verse en la tabla 4.

Tabla 4 – Casos de prueba con objeto leído y borrado

Use case/ Objeto	Use Case: Recepción de Materiales	Use Case: Facturar
Producto	C/A o (la responsabilidad del objeto en ese Use Case)	Read
Objeto2	R	D

En este proceso de testing: existen dos condiciones, como puede verse en la tabla 5:

- RC (*read before creation*)
- RD (*read after deletion*): cuando se borra un objeto que otro *use case* necesita leer o actualizar.

Tabla 5 – Casos de prueba con objeto leído después de borrado

	Use case1	Use case2	Use case3
Use case1			RD objeto 2
Use case2	RC objeto 1		
Use case3	RC objeto 1	RC objeto2	

En la tabla 6 se muestra cómo se combinan los casos de prueba que están involucrados en los RC o RD

Tabla 6 – Combinación de casos de prueba con RC y RD

	Recibir Materiales	Facturar	Registrar Cta. Cte.
Recibir Materiales			
Facturar	RC producto		RC cuenta corriente
Registrar faltantes	RC producto		

Una vez identificados los casos de uso relacionados con los mismos objetos que se leen y actualizan, para realizar las pruebas de interacción, se debe tomar como "condiciones" de las mismas a los distintos casos de prueba de los casos de uso que se están verificando. De esta forma, se completarán las planillas de casos de prueba de Interacción de los distintos módulos, señalando en cada condición a qué caso de uso se hace referencia. También cada condición deberá incluir los valores de prueba a utilizar.

La forma de evaluar los resultados de las pruebas de interacción es la misma que la utilizada para las demás pruebas en las que se realiza, de ser necesario, la tipificación de errores y la asignación de prioridades.

Una vez corregido el error se realizará un nuevo ciclo de testeo que será completo o no dependiendo del impacto que tengan las correcciones efectuadas.

3.4.3. Testing de regresión

La detección de errores durante la prueba de integración, involucra correcciones del software, lo que provoca cambios en la configuración de software ya sea en la documentación, el programa o los datos. El testing de regresión es la actividad que ayuda a asegurar que los cambios (debido a pruebas u otros motivos) no introduzcan un comportamiento no deseado o errores adicionales.

La aplicación de una prueba de regresión consiste en ejecutar nuevamente el mismo subconjunto de pruebas ya aplicado, para asegurarse que los cambios efectuados no hayan producido efectos colaterales no deseados.

Este tipo de pruebas pueden ser realizadas manualmente, mediante una nueva realización de un subconjunto de todos los casos de prueba o el uso de herramientas automáticas de reproducción de captura. Estas herramientas permiten capturar casos de prueba y los resultados para la siguiente reproducción y comparación. Son importantes los aportes de Bertolino (2003) y Harrold (2002), sobre los test de regresión a los fines de minimizar el costo de las sucesivas repeticiones de los mismos.

A medida que avanza la prueba de integración, se incrementa el número de pruebas de regresión, por lo que el conjunto de estas pruebas debe diseñarse para incluir una o más clases de errores en las funciones principales del módulo que ha presentado el cambio.

3.4.4. Testing de validación

Luego de la prueba de integración, el software está completamente ensamblado como un paquete; se han encontrado y corregido los errores, y se pueden comenzar a realizar las pruebas de validación. La validación del software se consigue mediante una serie de pruebas que demuestran la conformidad con los requisitos detallados en la especificación de requerimientos. La especificación de requisitos contiene una sección denominada Criterios de validación, cuya información constituye la base del enfoque de prueba de validación, el plan de prueba define las clases de prueba que se aplicarán, y un procedimiento de prueba define los casos de prueba específicos.

Tanto el plan como el procedimiento sirven para asegurar que se cumplen todos los requisitos funcionales, las características de comportamiento, los requisitos de desempeño, que la documentación es correcta, y que se cumple con otros requisitos tales como portabilidad, compatibilidad, recuperación de errores, etc.

Un elemento importante del proceso de validación es la revisión de la configuración. La intención de la revisión es asegurarse de que todos los elementos de la configuración del software se han desarrollado apropiadamente, se han catalogado y están suficientemente detallados para soportar la fase de mantenimiento.

3.4.5. *Testing de aceptación*

Por último, se llevan a cabo las pruebas de aceptación, para permitir que el cliente valide todos los requisitos. Las realiza el usuario final en lugar del responsable del desarrollo de sistema y pueden variar en cuanto a rigurosidad y duración, según las necesidades y los recursos existentes. En este ámbito, pueden citarse las llamadas pruebas alfa y beta.

Las pruebas alfa son aquellas que se llevan a cabo en el lugar de desarrollo pero por un cliente, el que usa el software de manera natural con el desarrollador como observador, quien registra los problemas y errores encontrados. Las pruebas beta son efectuadas por los usuarios finales y en el lugar de trabajo. Por lo general, el desarrollador no está presente, lo que le confiere a la prueba el carácter de una simulación en vivo del software. El usuario es el que registra los errores y se los comunica al desarrollador.

En este etapa, se acompaña al usuario (cliente) tanto en las pruebas que realiza en ambiente de laboratorio (pruebas alfa), como en las que realiza en ambientes de trabajo reales (pruebas beta). Para las pruebas alfa, se invita al cliente al entorno de desarrollo o se publica la aplicación en un server público. Se trabaja en un entorno controlado y el cliente siempre tiene un experto disponible para ayudarlo a usar el sistema y para analizar los resultados. Las pruebas beta se ejecutan después de las pruebas alfa y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallos (reales o imaginarios); los que informa al desarrollador.

3.4.6. Testing del sistema

Al incorporarse el software a otros elementos del sistema, como ser por ejemplo nuevo hardware, es necesaria la realización de otras pruebas cuyo propósito primordial es ejercitar, en profundidad el sistema a implementar, para verificar que se han integrado adecuadamente todos los elementos del sistema.

La prueba del sistema abarca una serie de pruebas de propósito diferente que se detallan a continuación:

- Recuperación: es una prueba del sistema que provoca fallas en el software y verifica que la recuperación se realice apropiadamente. Si el propio sistema la realiza, se evalúan que sean correctos los mecanismos de respaldo, de recuperación de datos y el arranque del sistema.
- Seguridad: comprueba que los mecanismos de protección incorporados en el sistema lo protejan de accesos impropios.
- Resistencia: estas pruebas están diseñadas para enfrentar a los programas con situaciones anormales, a tal fin, se ejecuta el sistema de tal forma que requiera una cantidad y una frecuencia anormal de recursos.
- Desempeño: permiten probar el desempeño del software en tiempo de ejecución, dentro del contexto de un sistema integrado. Suelen vincularse con pruebas de resistencia y, requerir hardware o software adicional para efectuar mediciones puntuales de recursos.

3.4.7. Testing funcional

También denominado de “caja negra” porque el comportamiento del sistema se puede determinar estudiando las entradas y salidas de la prueba del sistema. El nombre de testing funcional es debido a que al testeador o probador sólo le interesa la funcionalidad del sistema y no su implementación. Las pruebas funcionales o de caja negra se derivan de la especificación del sistema, y se realizan casos de prueba cuya

selección se basa en el conocimiento del dominio para identificar aquellos casos que con mayor probabilidad revelen defectos del sistema.

Según Boris Beizer (Beizer, 1995), una estrategia de prueba o técnica de prueba es un método sistemático usado para seleccionar o generar casos de prueba. Resulta efectiva si los test incluidos en ella revelan bugs en el objeto testeado.

El enfoque que se da a este trabajo se basa en la estrategia basada en el comportamiento, también llamada *Prueba de Caja Negra* o *prueba funcional*. Se la denomina así porque considera el sistema como una caja negra, cuyo comportamiento es determinado únicamente a partir de sus entradas y salidas.

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y a estudiar la salida, sin preocuparse de lo que pueda estar realizando el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc.) No obstante, pueden resultar útiles en cualquier módulo del sistema. Estas pruebas se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea.

El problema que se presenta en las pruebas de caja negra no suele estar en el número de funciones proporcionadas por el módulo -que siempre es un número muy limitado en diseños razonables-; sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio -por ejemplo, un entero-.

A la vista de los requisitos de un módulo, se sigue una técnica algebraica conocida como "clases de equivalencia". Esta técnica trata cada parámetro como un modelo algebraico donde unos datos son equivalentes a otros. Si logramos definir un rango excesivamente amplio de posibles valores reales a un conjunto reducido de clases de equivalencia, entonces es suficiente probar un caso de cada clase, pues los demás datos de la misma clase son equivalentes.

El problema consiste, entonces, en identificar clases de equivalencia; tarea para la que no existe una regla de aplicación universal; sin embargo hay recetas para la mayor parte de los casos prácticos:

Durante la lectura de los requisitos del sistema, se localizarán una serie de valores singulares, que marcan diferencias de comportamiento. Estos valores son claros candidatos a marcar clases de equivalencia: por abajo y por arriba. Una vez identificadas las clases de equivalencia significativas en el módulo en cuestión, se procede a seleccionar un valor de cada clase, que no esté justamente al límite de la clase. Este valor aleatorio, hará las veces de cualquier valor normal que se le pueda pasar en la ejecución real.

La experiencia muestra que un buen número de errores aparecen en torno a los puntos de cambio de clase de equivalencia. Hay una serie de valores denominados "frontera" (o valores límite) que conviene probar, además de los elegidos en el párrafo anterior. Usualmente se necesitan 2 valores por frontera, uno justo abajo y otro justo encima.

3.4.7.1. Limitaciones de las pruebas de caja negra

Lograr una buena cobertura con pruebas de caja negra es un objetivo deseable; pero no suficiente a todos los efectos. Un programa puede pasar con holgura millones de pruebas y sin embargo tener defectos internos que surgen en el momento más inoportuno (Murphy no olvida). En la tabla 7 se presenta una síntesis del enfoque de caja negra.

Este enfoque se aplica de igual forma a los sistemas estructurados que a los orientados a objetos. En el caso de sistemas desarrollados para la Web, el testing funcional involucra una evaluación de todos los aspectos del sitio donde existe código o scripts involucrado, desde la búsqueda de links defectuosos hasta probar formularios Web y scripts. Dos estrategias utilizadas para probar los primeros son el ingreso de información real (real-world data) y el ingreso de información errónea (denominada "extreme" data).

Tabla 7– Enfoque de caja negra

CAJA NEGRA
Orientada al comportamiento
Enfocada a la funcionalidad del software
Detecta errores de: ▪ Funcionalidad incorrecta ▪ Funciones ausentes ▪ Estructura de datos
Tipos: ▪ Partición equivalente ▪ Análisis de valores límite ▪ Pruebas según la experiencia ▪ Tabla de decisiones

3.5. Testing de arquitecturas cliente/servidor

Los métodos de prueba expuestos en el punto anterior, pueden aplicarse para todas las arquitecturas y aplicaciones, pero en determinados entornos deben utilizarse enfoques especiales para realizar las pruebas.

Este es el caso de la arquitectura cliente servidor, donde la naturaleza distribuida de la misma, la complejidad de la comunicación en red, la existencia de una base de datos a la que acceden múltiples clientes, y las características especiales de los servidores, hacen que la prueba de sistemas bajo esta arquitectura sea considerablemente más compleja e implique mayor tiempo y costo en su ejecución.

En general pueden especificarse los siguientes enfoques de prueba para esta arquitectura:

- Prueba de funcionalidad de la aplicación que se realiza en forma independiente.
- Pruebas del servidor en cuanto a la coordinación y manejo de datos, así como su desempeño.
- Pruebas de la base de datos para asegurar la exactitud e integridad de los datos almacenados.

- Pruebas de transacción de acuerdo a los requisitos especificados y a los tiempos de respuesta y volumen
- Pruebas de la red para verificar la comunicación entre los distintos nodos que la componen y que el tráfico por la misma no acuse errores.
- Pruebas de perfiles de usuario de acuerdo a los requerimientos estipulados en los casos de uso.
- Pruebas de interfaces gráficas de usuario (GUI), que, de acuerdo a su mayor o menor complejidad debe validarse no sólo el contenido sino también su diseño y amigabilidad.

3.5.1. Pruebas de Interfaces de Usuario

Las pruebas de interface y de reportes asociadas a un determinado caso de uso tienen como base los requerimientos del usuario, los que tienen lineamientos generales o estándares y condiciones especiales para algunos atributos según el caso de uso. Para los lineamientos estándares se utilizará un checklist con los distintos ítems a tener en cuenta y se completará cada uno con el valor. A saber,

- **OK o SI**: La pantalla o reporte cumple con lo definido para este ítem
- **NO OK o NO**: La pantalla o reporte no cumple con lo definido para este ítem
- **NO APLICA**: No está definido este ítem para esta pantalla o reporte o no existe en la misma.

Para las condiciones especiales se deberán especificar los datos ya mencionados de atributo o condición a validar, el valor con el que se realizará la prueba y el resultado esperado. Una vez ejecutada la prueba, se registrarán los resultados de la misma forma que en las pruebas funcionales indicando, en caso de haberlo, el tipo y prioridad del error. Una vez corregido el error se procederá a realizar un nuevo ciclo de testeo.

A través del checklist se verifica entre otras cosas:

- Modos de ventana
- Menús
- Tamaño estándar de controles
- Redacción

Este checklist se entrega a los programadores en el momento del desarrollo.

3.6. Testing de arquitecturas orientadas a objetos

En general la prueba orientada a objetos es similar a la de los sistemas convencionales, pero algunas características propias de esta arquitectura hacen que deban aplicarse las técnicas en forma particular.

Cada uno de los elementos de estos sistemas, subsistemas y clases, realiza funciones que ayudan a cumplir con los requisitos del sistema, comenzando con una revisión de los modelos construidos y, una vez generado el código, poder efectuar pruebas orientadas a las clases y sus colaboraciones y completar con los subsistemas donde éstas se integran.

Pueden aplicarse los métodos de prueba de caja blanca para las operaciones definidas para las clases, y, fundamentalmente, considerar los métodos de prueba de caja negra a través de los casos de uso, los cuales proporcionan la base para el diseño de casos de pruebas de caja negra basadas en el estado.

Suelen aplicarse también técnicas de pruebas basadas en fallas, diseñando casos de prueba específicos que revisen el diseño y el código.

En el caso específico de estas pruebas orientadas a objeto, las pruebas de integración se aplican a los atributos y operaciones de los objetos, para lo cual las pruebas ejercitan determinados valores de los atributos para determinar el comportamiento correcto de los objetos.

3.7. Testing de arquitecturas Web

La estrategia de prueba de aplicaciones Web sigue los principios de todas las pruebas de software convencionales, y, en especial aplica las técnicas mencionadas en los sistemas orientados a objeto.

Esta estrategia es una suma de actividades que se realizan durante todo el proceso de la ingeniería Web, y, abarca tanto revisiones como pruebas ejecutables, a los fines de descubrir errores en el contenido, la funcionalidad, la facilidad de uso, la navegación, el desempeño y la seguridad de las aplicaciones Web.

Nguyen (Nguyen, 2003), plantea distintos tipos de errores que se producen en ambientes Web:

- Problemas que se evidencian en el lado del cliente a través de su interfaz, donde se ve el síntoma del error, no el error en sí.
- Debido a que las aplicaciones Web se implementan en configuraciones diferentes, resulta muchas veces difícil reproducir un error fuera del ambiente donde se originó.
- Puesto que las aplicaciones Web se ejecutan en una arquitectura distribuida cliente/servidor, los errores deben rastrearse en las tres capas de esta estructura: cliente, servidor o en la red en sí.
- Algunos errores se producen debido a un diseño o codificación incorrecta y deben investigarse hacia dentro de los componentes de la aplicación.
- Existen errores debidos al ambiente operativo donde se realizan las pruebas y otros son atribuibles a la carga dinámica de recursos o al tiempo de ejecución.

En consecuencia, puede apreciarse que en las pruebas de aplicaciones Web, el ambiente desempeña un rol importante a tener en cuenta para el diagnóstico de errores encontrados durante las mismas.

A continuación se caracteriza el comportamiento de distintos tipos de testing dentro de un entorno Web.

3.7.1. Testing de desempeño

El desempeño de un sitio Web es afectado principalmente por la configuración del servidor, la configuración del cliente, por la red y por la frecuencia de peticiones. Sin embargo, dos son los factores que siempre influyen en la variación de la configuración del servidor: fallas no planificadas de componentes del sistema y mantenimiento.

La idea central es que si se está ofreciendo un sistema de alto desempeño, existe la necesidad de simular algunas fallas inesperadas para determinar si el sitio cumple con los requerimientos globales de desempeño. En este caso, resulta útil y necesario diseñar pruebas que simulen las fallas de los componentes principales de la arquitectura del sitio y del servidor (desconectar un procesador, desconectar un cable, apagar el servidor, deshabilitar los servicios HTTP o FTP, entre otros). Para el mantenimiento del sistema, resulta una buena práctica el simular un proceso de mejora a los componentes de la arquitectura, como por ejemplo, el mejorar la versión del servidor Web o desconectar temporalmente el sistema para efectuar un respaldo.

Las pruebas de desempeño deben ser diseñadas con diferentes configuraciones, esto es, pruebas con diferentes velocidades de máquinas, diferentes browsers, diferentes métodos de acceso a Internet y diferentes velocidades de conexión.

3.7.2. Testing de usabilidad

El testing de usabilidad se ocupa de identificar y entender todas las respuestas y problemas que un usuario pueda tener mientras trabaja con la aplicación. La usabilidad es una amplia disciplina que debe tener en consideración aspectos de diseño, facilidad de uso y facilidad de aprendizaje, entre otros. De la misma manera, necesita conocer los requerimientos mínimos del usuario como: funcionalidad mínima requerida; limitaciones -ancho de banda, tipo de navegador, nuevas interfaces, plataformas, pluggins-; preferencias del usuario en cuanto a gráficos y textos; hábitos del usuario; sistemas existentes e información específica del tipo de usuario -edad, nivel de estudios, etc.-.

Tal como lo señala Ricardo Baeza (2005), a los fines de evaluar una aplicación Web, existen varias técnicas, desde las muy sencillas hasta las altamente sofisticadas. Pueden realizarse las pruebas tanto en ambientes controlados como en el lugar mismo donde se va a usar el sistema. Del mismo modo, puede hacerse una evaluación automatizada, o evaluar el sistema por usuarios reales. En realidad, todo depende de los recursos disponibles al momento de hacer la evaluación. Se debe destacar, en todo caso, el hecho de que las pruebas de usabilidad son costosas en términos de tiempo y personal requerido para realizarlas.

El tipo más común de pruebas de usabilidad son las pruebas de prototipos de alta fidelidad, también conocidas como pruebas de productos finales (Spool, 2004). Estas pruebas se realizan en productos que están en su última fase de desarrollo, prácticamente listos o ya en uso. Cualquier problema mayor detectado, se arregla en la próxima versión del producto.

En general, el test de usabilidad se utiliza a través del ciclo de vida del desarrollo del producto. En las etapas tempranas del proceso de desarrollo, el test de una versión previa o del producto de un competidor proporciona referencias muy útiles al equipo de diseño. En las etapas medias, el test valida el diseño e informa sobre las posibilidades de refinamiento del mismo. En etapas posteriores, el test asegura que el producto alcanza los objetivos del diseño.

3.7.3. Testing de compatibilidad

Este tipo de testing asegura que el sitio Web funciona correctamente en todo tipo de clientes, browsers, versiones de browser, plataformas y diferentes máquinas. Desafortunadamente, el esfuerzo de normalización de los diferentes estándares, desde HTML al DOM, todavía no garantiza un comportamiento igual entre los browsers y requiere un esfuerzo de test adicional. Es imprescindible que el sitio Web ofrezca el mismo grado de funcionalidad con todos los browsers del mercado, o al menos con las versiones más habituales de Netscape e Internet Explorer.

Asimismo, es real que los browsers no se comportan de manera igual en Windows, MacOS o UNIX. Es por esto necesario probar el sitio con la mayoría de las plataformas del mercado y todas sus versiones (Windows, MacOS, Linux, Solaris, HP-UX, AIX, Irix, etc).

3.7.4. Testing de seguridad

El aspecto relacionado con la seguridad resulta uno de los más importantes, debido a la magnitud de las transacciones que se realizan sobre Internet. Por ello, en el desarrollo del producto, es vital cubrir un conjunto mínimo de condiciones de seguridad que engloben tanto la privacidad del usuario como la estabilidad del servicio. Mecanismos como el intercambio de claves públicas, más algoritmos de encriptación han permitido

cubrir el primer punto. La estabilidad del servicio debe ser cubierta en general por el uso de sistemas de respaldo y redundancia de información (sistemas RAID en otros niveles).

El Testing de seguridad vela por el cumplimiento de estas primitivas de funcionamiento, las cuales se pueden resumir en:

- Autenticación: validar que quien hace las acciones sea efectivamente quien dice ser (ingreso a un sistema).
- Integridad: la información debe ser correcta independiente de errores o inconsistencias externas.
- Privacidad: la información debe ser visible sólo por los usuarios autorizados.
- No-repudio: el servicio debe estar en línea, más aún si se trata de servicios críticos (un banco en línea por ejemplo).
- Disponibilidad: el servicio debe estar disponible aún en presencia de fallas (uso de sistemas RAID, clusters, entro otros).

3.7.5. Testing de escalabilidad

Los tests de escalabilidad ayudan a caracterizar el sistema, a identificar cuándo y cómo se degrada el acceso al sitio Web, en función del número de accesos. Para esto es posible crear escenarios de navegación: en base a casos de uso reales y con la ayuda de herramientas se simula hasta miles de transacciones simultáneas.

Su máxima expresión es la prueba de estrés (stressing), por la que se prueba el sistema en los límites extremos para determinar su nivel de tolerancia y si ocurre algún tipo de falla. También se mide el volumen de datos intercambiados, las tasas de fallo o el tiempo total de transacción a los fines de otorgarle una imagen real de la experiencia de un usuario.

3.7.6. *Testing de aceptación visual*

Este tipo de testing asegura que el sitio tenga la apariencia deseada. Esto incluye revisar la integración gráfica, así como la simple confirmación de que el sitio “se ve bien”. En este punto es importante probar que entre las páginas del sitio exista consistencia visual y gráfica. Las pruebas deben ser realizadas en distintos entornos como: distintas tarjetas gráficas, frecuencias de refresco, resolución de colores (256, 32 y 16 bits) y de pantalla (1024*768, 800*600, etc.), entre otras.

Es importante señalar, que ésta es una de las primeras pruebas que el usuario inconscientemente efectúa puesto que, en teoría, éste puede retener un número fijo de conceptos. Diversos son los ejemplos de sitios Web sobrecargados de componentes que provocan el rechazo inmediato por parte del usuario.

3.7.7. Testing de disponibilidad de servicio

Se simula durante un período amplio de tiempo un tráfico regular, sin sobrecarga. Las pruebas diseñadas permiten comprobar cuánto tiempo el sistema trabajará con un rendimiento óptimo, o si el servicio está sufriendo una degradación lenta.

3.7.8. Testing de idioma

Un aspecto al que muchas veces se le otorga poca importancia es el idioma en que será utilizada la aplicación. Por ello resulta importante realizar pruebas reales en los idiomas que se desea, donde no sólo exista una prueba superficial sino que se prueben las funcionalidades del sitio en condiciones de uso reales, con el sistema operativo y el browser de los idiomas seleccionados.

3.7.9. Testing de motores de búsqueda

En caso que el sitio Web posea un motor de búsqueda, es necesario probar si entrega los resultados correctos y los más semejantes al patrón de búsqueda. Para ello, las pruebas deben estar orientadas a las expresiones regulares aceptadas por el motor. Otro factor a probar es la velocidad de procesamiento del motor.

3.7.10. Testing de accesibilidad

Este tipo de testing, según el AQA Focus Document (2002), se propone facilitar que personas con discapacidades puedan navegar y ejecutar las acciones deseadas en el sitio Web. Para analizar los problemas de accesibilidad que las páginas de un sitio Web van a generar en estos usuarios, se intenta simular el modo en que ellos van a acceder a las mismas. Para ello, se emplean los mismos navegadores alternativos que estas personas utilizan, o un programa que simule su funcionamiento.

La accesibilidad no es de interés únicamente para las personas con discapacidad sino que mejora el acceso a la Web en general. Con el fin de facilitar el acceso y la navegación en éstas y otras condiciones, el WAI (Web Accessibility Initiative) perteneciente al W3C desarrolló un conjunto de pautas o reglas básicas de accesibilidad que deben ser revisadas y probadas.

3.7.11. Testing de prueba

Este tipo de testing hace referencia a la capacidad de un sistema de soportar las nuevas pruebas, después de sufrir una serie de cambios importantes en su configuración. Por ejemplo, si a un sitio Web -que ya pudo haber sido probado- se le efectúa un proceso de mejora a un componente importante, es vital que el proceso de testing aplicado a la versión anterior -antes del cambio- soporte esta acción, o que el proceso de mejora sea compatible y consistente con el plan de testing.

3.7.12. Testing de script

Los elementos de script asociados a una página Web, y por ende a un sitio Web, se pueden clasificar en tres categorías:

- Código visible: el cual es accesible públicamente por los usuarios desde el browser. Por ejemplo: HTML, Javascript y VBscript.
- Código generador: el cual es ejecutado en el servidor Web y tiene como salida código visible (asp, php, perl, jsp), y no puede ser accedido en forma directa por el usuario.
- Componentes incrustados: los cuales son ejecutables en el browser (con alguna extensión o plug-in previamente instalada), sin que el usuario tenga acceso a su código fuente. Por ejemplo: applet de java, video Quicktime o un archivo de Realmedia.

Para la primera categoría es factible aplicar técnicas de caja blanca, por contar con el código de forma directa. Esto también es aplicable al código generador en caso de contar con acceso a él. Aún así, un usuario podría aplicar técnicas de caja negra al recibir como salida código visible ante ciertos parámetros de entrada. Los componentes incrustados, por otra parte, sólo pueden recibir pruebas de caja negra por no contar con el código fuente.

3.8. Características del Proceso de Web testing

Las actividades de testeo pasan a ser parte de todo el ciclo de vida del producto e incluyen también como métodos de testeo a las revisiones, auditorias, walk throughs e inspecciones que normalmente pertenecerían a la etapa de validación y verificación. En

la figura 3 se incluye el modelo en V ampliado/modificado mostrando cómo se integra la validación al ciclo de vida de desarrollo del software.

En el costado izquierdo del Modelo en V se puede ver las fases de Captura de Requerimientos, Análisis y Diseño y Diseño de HW y SW y Codificación y, en el lado derecho del modelo se muestran las actividades de testeo que deben llevarse a cabo después que se ha completado la Implementación.

.En la etapa de análisis y captura de requerimientos, se obtienen todas las especificaciones del usuario que serán contrastadas en la etapa de testing de aceptación.

Sucede de manera análoga con el diseño, el cual entregará las pautas y escenarios para probar que el sistema responde a las especificaciones de diseño establecidas. En la etapa del diseño del hardware y software, se provee los escenarios que serán probados en la etapa de la integración de los módulos, para finalmente llegar al testing unitario que prueba cada módulo por separado.

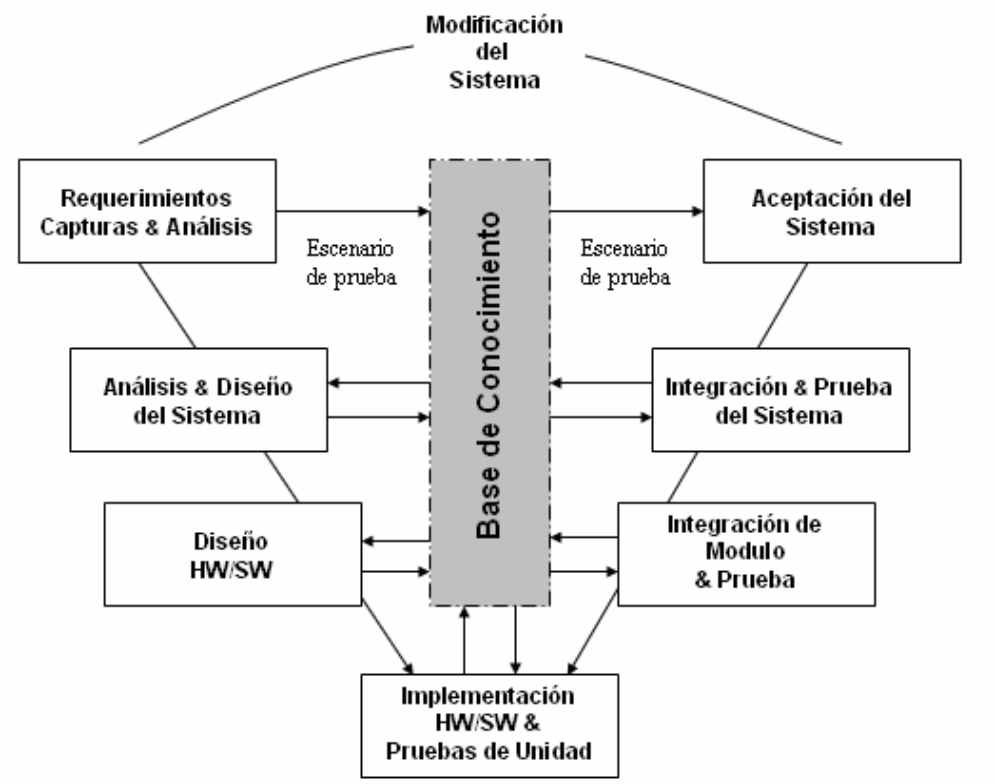


Figura 3– Diagrama V

Sintetizando:

- La prueba unitaria centra sus actividades en verificar la lógica del módulo (caja blanca), y, la especificación de las funciones que debe realizar el mismo (caja negra).
- La prueba de integración debe tener en cuenta la agrupación de módulos así como las interfaces entre componentes de la arquitectura de software.
- La prueba del sistema debe focalizarse en el cumplimiento de los objetivos indicados para el sistema.

- La prueba de aceptación permite que el usuario verifique si el producto final se ajusta a los requisitos establecidos de antemano con su participación.

La propuesta de un grupo de doctores del "Illinois Institute of Technology", Ilene Burnstein, Taratip Suwannasart y C.R. Carlson desde 1996, es la de aplicar un Modelo de Madurez específico del proceso de testeo, no incluido dentro del modelo CMM, en el cual una organización pueda calificar su estado actual del proceso de evaluación y pruebas del software, y comenzar a organizar sus actividades de testeo con un sistema de mejoramiento continuo. Al modelo desarrollado, lo han llamado TMM, Testing Maturity Model, o Modelo de Madurez de Testeo, en obvia referencia al CMM, al cual no alegan reemplazar sino más vale complementar, ya que utilizan un esquema de implementación similar.

Por medio de este modelo las organizaciones que desarrollan software pueden entrenar a un grupo interno de valorización que calificaría la madurez de la organización en su proceso de testeo. Con esta evaluación la dirección puede tomar la decisión de comenzar un proceso de mejoramiento del testeo que mejore las capacidades de pruebas de los equipos de desarrollo. También los clientes pueden utilizar este modelo para definir su rol en el proceso de testeo de un software que han adquirido. A partir de este modelo surgieron otros como: Test Process Improvement (TPI), Test Organization Maturity (TOM), etc., pero haciendo un proceso de comparación entre ellos los dos más aplicables son: SW – TMM y TPI.

Como criterios de selección del SW-TMM se pueden citar:

- su facilidad de comprensión y uso,
- permitir a las organizaciones realizar sus evaluaciones internas,
- la habilidad de proveer una línea base de la función de testeo y una alternativa de mejora
- la capacidad de ser usado para aplicaciones de telecomunicaciones, basada en la Web, y otras
- su facilidad de ser utilizado en conjunción con SW-CMM, aunque igual puede ser usado aún si la organización no tuviera definido su modelo de madurez del proceso de desarrollo, pero su interés o su debilidad está focalizada en el testeo de software.

3.9. Condiciones especiales para el Web testing

Tras los fundamentos expuestos en el transcurso de este documento, resultan las siguientes conclusiones y comentarios:

Debido a que las aplicaciones en la Web residen en una red e interoperan con distintos sistemas operativos, navegadores, plataformas de hardware y protocolos de comunicación, existe una fuerte dependencia entre las acciones llevadas a cabo por los usuarios y el entorno de ejecución.

Los modelos de contenido y de diseño deben ser revisados convenientemente a los fines de encontrar errores en: consistencia del contenido, ortografía, representaciones gráficas, referencias cruzadas, navegación de páginas, hiperenlaces, etc.

Como no siempre es posible probar los componentes aislados de la aplicación Web, se considera que la prueba unitaria mínima la representa la página Web, a través de su contenido, proceso y enlaces encapsulados.

La estrategia para la prueba de integración depende de la arquitectura elegida para la aplicación Web: si se ha diseñado con una estructura jerárquica lineal, es posible integrar las páginas Web como se integran los módulos en el testing convencional, en

cambio si se utiliza una arquitectura en red o una jerarquía híbrida, la prueba de integración es similar al enfoque orientado a objetos.

Las pruebas de regresión se aplican para asegurar que no haya efectos secundarios ante cualquier modificación de la aplicación Web.

A continuación se prueba la aplicación Web ensamblada, esto es la prueba de validación o aceptación para obtener una funcionalidad global y un contenido al igual que en el testing tradicional, pero focalizándose en el usuario y en los requisitos de interacción con el mismo.

Para comprobar la compatibilidad de las distintas configuraciones posibles, se implementa la aplicación Web en distintos entornos operativos y protocolos y se prueba en ellos, registrándose las diferencias encontradas y analizando las mejoras a realizar. Por último se selecciona un grupo de usuarios que abarque todos los roles posibles y se evalúan los resultados de su interacción con el sistema.

La constante evolución de las aplicaciones Web hace que el proceso de validación y verificación sea de mejoramiento continuo, por lo que se utilizan en la práctica pruebas de regresión derivadas de las pruebas desarrolladas cuando se diseñaron las aplicaciones Web originales.

CAPÍTULO 4. ENFOQUE FUNCIONAL DE TESTING DE MIGRACIÓN

4.1 Introducción

Ya en la introducción de este trabajo se abordó, en forma general, el problema de la migración de sistemas. Ahora, en razón de que este trabajo se ocupa del testing de migración de aplicaciones distribuidas a entornos Web, y previo a desarrollar los principios fundamentales del mismo, es necesario formular la siguiente pregunta: ¿se está probando el producto migrado correcto? Tal fundamental interrogante sólo podrá responderse mediante un profundo conocimiento del producto y del proceso de desarrollo que se concretó en la versión cliente servidor anterior, en especial, a partir de conocer las funcionalidades del producto a migrar que se han de testear.

Aquí debe tenerse en cuenta que en la mayoría de las aplicaciones cliente servidor fue utilizada la metodología estructurada, y por lo tanto, es necesario utilizar las herramientas de modelado del análisis estructurado de Yourdon (1993) para conocer las estructuras de estos sistemas en su versión original. Debe observarse que las metodologías están estrechamente relacionadas con la construcción del software y su ciclo de vida. Por lo tanto, resultaría un error emprender un desarrollo o su posterior validación sin una metodología que gobierne estos procesos. Como es sabido, esta lección fue aprendida con un elevado costo y a ello se debe la existencia de la ingeniería de software.

Con respecto a la aplicación Web, la mayoría de los problemas del diseño Web pueden simplificarse si se utiliza la notación UML. Desde que UML fue adoptado como el lenguaje estándar para el modelado, se utiliza como medio de expresión de los diferentes modelos que se crean durante el desarrollo de los sistemas. Como estos desarrollos están guiados por los casos de uso, los mismos se emplean para definir los requisitos funcionales del sistema, y, en general, todas las etapas del proceso se articulan en torno a los casos de uso identificados. Gracias a la definición de su metamodelo y a los mecanismos de extensión incluidos, UML ofrece la oportunidad de establecer un marco común para la representación de especificaciones de requisitos, de desarrollo y de pruebas.

Los perfiles UML incorporan mecanismos de extensión (estereotipos, valores etiquetados y restricciones) que permiten personalizarlos para distintas aplicaciones y tecnologías y han sido utilizados para representar la arquitectura del sistema a migrar y la del sistema migrado.

En este capítulo se presenta en primer término una metodología para el conocimiento del sistema a migrar, que obviamente se constituirá en la referencia obligada para el proceso de validación. Luego, se presenta una metodología para el conocimiento del nuevo sistema que ya ha sido migrado, y para esto último se pone el foco en la arquitectura cliente servidor, en la tecnología Web y en las reglas de negocio que deben ser reusadas para abordar el proceso de migración.

Basado en el aprendizaje previo sobre el sistema objeto de estudio, se propone un enfoque de testing de migración con una visión funcional sustentada en el reuso de casos de prueba. Éstos resultan de la trazabilidad con los casos de uso desarrollados de acuerdo a la metodología de migración presentada.

Se profundiza también en la identificación de las condiciones de prueba que resultan de las precondiciones existentes, y, en el proceso iterativo e incremental para el desarrollo de las pruebas, para lo cual se establece un seguimiento de las mismas a través de los conceptos de ciclo y versión.

En el Anexo I se presenta un estudio comparativo de las distintas metodologías utilizadas para el desarrollo de sistemas.

Para finalizar se desarrollan los distintos tipos de pruebas funcionales entre las que se destacan las pruebas de interface con el usuario y las pruebas de integración.

4.2 Metodología propuesta de análisis lógico y físico del sistema a migrar

Se presenta a continuación la metodología para conocer los detalles de la arquitectura y diseño del sistema anterior, además de las limitaciones tecnológicas y de las reglas de negocio existentes en el momento de su construcción. La aplicación de esta metodología en un caso de estudio se desarrolla en el capítulo 5.

4.2.1 Reconstrucción de la especificación de requerimientos

Se asigna mucha importancia a la reconstrucción de la especificación de requerimientos del sistema a ser migrado. Obviamente, la profundidad con la que pueda cumplirse esta tarea dependerá de cada caso, y la antigüedad del sistema será seguramente uno de los factores determinantes. En efecto, en la medida que la antigüedad sea mayor se dispondrá de menos testimonios y se caerá en el caso de los ya descriptos “sistemas heredados”. Sin embargo, cualquiera sea el caso tratado, todo el esfuerzo que pueda ser realizado para la reconstrucción de la especificación de requerimientos estará ampliamente justificado y será de gran valor en las próximas etapas.

a) Estudio preliminar

Realizar un relevamiento exhaustivo del sistema original, utilizando para ello técnicas de entrevistas a personas que hayan participado del mismo, desde el área técnica al área operativa (stakeholders). Recopilar toda la documentación disponible, incluyendo manuales de procedimientos, y analizar las reglas de gestión del sistema, características generales, perfiles, procesos, etc.

b) Reconstrucción

Clasificar y ordenar toda la información testimonial y documental que pueda haberse obtenido en la etapa anterior.

4.2.2. Descripción de la dimensión funcional

La dimensión funcional es el eje de las etapas de análisis y diseño, tanto de las metodologías estructuradas como orientadas a objetos, por lo que su correcta definición es esencial en todo el proceso de validación de un sistema.

a) Elaboración del diagrama de contexto

Establecer las fronteras del sistema con los otros sistemas y agentes externos que se vinculan con el mismo mediante un Diagrama de Contexto. Este diagrama de contexto es un caso especial de diagrama de flujo de datos (DFD), denominado de nivel “0”, que representa globalmente al sistema como una sola burbuja y donde un proceso único define la frontera o marco del análisis con el sistema externo. Así se definen las interfaces del sistema con el resto del universo.

b) Análisis de comportamiento del sistema

Confeccionar la lista de eventos o acontecimientos que estimulan al sistema y a los cuales debe darse respuesta, precisando en cada caso la naturaleza del estimulo y el tipo de respuesta esperado. Una vez completada esta lista se podrán establecer los escenarios que componen los distintos subsistemas del sistema a migrar y se completará el estudio de su comportamiento.

c) Construcción de diagramas de flujo de datos - DFD

A partir de los estímulos identificados se deben desarrollar diagramas que permitan estudiar los flujos de datos y sus relaciones y transformaciones con los distintos procesos asociados. El diagrama de flujo de datos (DFD) es una técnica que representa el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida. Cada proceso, representado por una burbuja, puede refinarse en otros DFD's de menor nivel para mostrar un mayor detalle.

d) Realización del modelo de casos de uso

El objetivo de esta tarea es especificar cada caso de uso identificado en el análisis del comportamiento del sistema, representando gráficamente los escenarios involucrados. Para cumplir con este objetivo se construye el Modelo de casos de uso de trazo grueso de la aplicación distribuida, identificándose las relaciones entre los mismos.

4.2.3. Descripción de la dimensión estática del sistema anterior

Normalmente, el modelo funcional conduce a una base sólida para completar luego la dimensión estática del sistema.

a) Reconstrucción del modelo conceptual

El modelo conceptual se utiliza para obtener una descripción del dominio del problema real, no es una descripción del diseño del software. Es usado como base para una vista unificada de los datos y se representa con un diagrama gráfico de estructura estática, con las distintas entidades que componen el diseño lógico del sistema, sus relaciones y cardinalidad.

4.2.4. Descripción de la interfaz del usuario

Como ya fue descrito con anterioridad, la interfaz con el usuario es uno de los aspectos que sufre gran impacto en la migración de sistemas a entornos Web y por lo tanto esta interfaz debe ser descripta con el mayor detalle y cuidado.

a) Análisis del modelo de interfaz del usuario

Representar la interfaz de usuario, mostrando las distintas pantallas que intervienen en la aplicación a migrar y realizando un diagnóstico sobre su presentación y contenido. Considérese que esta interfaz debe permitir la captura de datos con velocidad y reducción de errores, ya que por un lado se tiene un sistema físico o informático y por el otro a una persona que desea interaccionar con él, a través de instrucciones concretas. La interfaz de usuario es la herramienta que entiende a ambos y es capaz de traducir los mensajes que se intercambian por medio de un acceso amigable, congruente con sus necesidades y adecuado a principios ergonómicos.

La interfaz de usuario puede responder a distintos tipos: interfaces de lenguaje natural, preguntas y respuestas, menús, de llenado de formas, lenguajes de comandos e interfaces, gráficas de usuario (GUI), etc.

b) Construcción del metamodelo del sistema

La definición de metamodelo implica que es un modelo con la capacidad de almacenar diferentes modelos. En este caso, el metamodelo muestra cómo deben ser representados los mecanismos de la aplicación en una arquitectura de dos capas.

Se presentan, a través del uso del lenguaje unificado de modelado (UML), las interacciones entre el usuario cliente y la base de datos, mediante la utilización de modelos de formularios y de componentes, para lo cual existe una entidad central que despliega la información al usuario mediante la carga de forms.

Un form es un programa con los elementos necesarios para que el cliente pueda interactuar con la base de datos. Posee un conjunto de objetos que permiten, entre otras cosas, desplegar nuevos formularios, cajas de texto para ingresar datos, combos para seleccionar información existente, controles Data-Windows para acceso a datos y conexiones con Bases de Datos que facilitan la organización e interacción. El llamado de un form a otro es modelado por la asociación que se invoca a sí mismo.

Los forms pueden ser estáticos o dinámicos mediante el uso de componentes de otros fabricantes, tales como librerías .dll, .ocx, entre otros. También contiene componentes generados por el propio equipo de desarrollo.

La organización en forms es representada por una asociación jerárquica, cuyo destino es un conjunto de entidades forms dependientes unas de otras. La subdivisión en forms tiene una asociación unaria, es decir, cada form puede recargar una y sólo una childform por vez, constituyendo esto una desventaja con respecto a las web page. Cuando un botón en un form fuerza la carga de otro form, el form de destino se convierte en el form activo.

La interfaz existente entre los forms y la BaseDatos, es la que permite el uso del SGBD. El acceso se produce mediante una clase de componentes que provee los drivers nativos de conexión.

4.2.5. Descripción de la arquitectura física y del software de base

La arquitectura software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura software debe ser implementada en una arquitectura física, que consiste simplemente en determinar qué tecnología tendrá asignada cada tarea del sistema informático.

a) Arquitectura física

Al describirse la arquitectura física del sistema a migrar deben considerarse dos aspectos principales: 1) tipo y capacidad de unidades centrales, dispositivos de almacenamiento, características de monitores y otros periféricos de entrada y salida (lectoras de códigos de barra, scanners, impresoras, tickeadoras, etc) y 2) Interconectividad entre los equipos (red LAN, WAN, etc). En la figura 4 puede verse el esquema descripto:

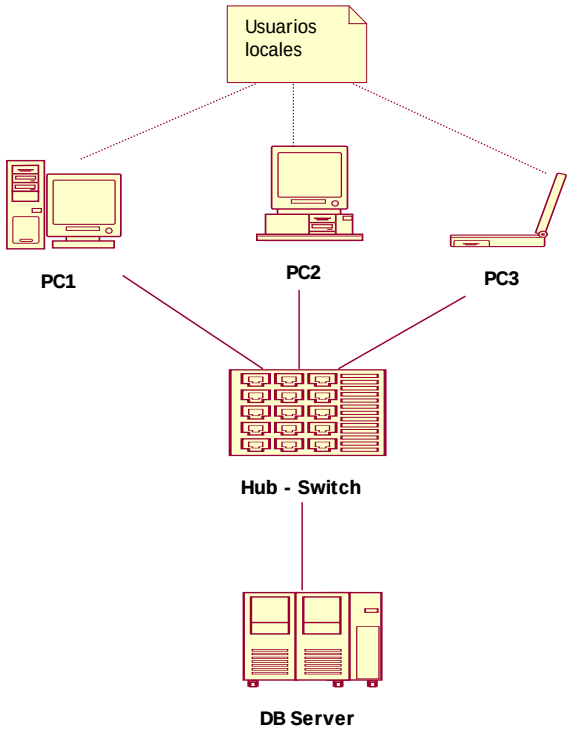


Figura 4 – Gráfico de componentes distribuidos del sistema de Autogestión

b) Arquitectura del Software de Base y diagrama de componentes

Describir la plataforma, incluyendo sistema operativo, interfaz gráfica, motor de base de base de datos (local o remota), uso de API y librerías propias del Sistema Operativo, otras librerías, componentes OCX, componentes DLL propias y de otros fabricantes.

Un componente representa una parte del sistema que puede ser implementado por uno o más elementos tales como ficheros binarios, ejecutables o scripts, y puede exponer un conjunto de interfaces que representan servicios que proporcionan los elementos que residen en el componente. También pueden observarse en el diagrama de componentes las relaciones de dependencia entre los mismos mediante flechas discontinuas.

Como se expresara con anterioridad, la aplicación a migrar constituye un sistema Client-Server, en un modelo bicapa, en el que sus clientes son FAT CLIENT, porque tanto los componentes de presentación como los de gestión de datos se encuentran del lado del cliente, generando dificultades en el mantenimiento; en tanto que el manejo de las transacciones las realiza el DTS (Data Transaction Server) del lado del servidor.

4.2.6. Análisis de las limitaciones del modelo anterior

Describir las limitaciones o restricciones que se presentan en el sistema a migrar, tales como el empleo de determinadas metodologías de desarrollo, lenguajes de programación, normas particulares, restricciones de hardware, de sistema operativo etc. Debe además considerarse que, con mucha frecuencia, la migración de un sistema arrastra expectativas referidas a mejoras operativas y/o funcionales que resultan ajenas a la propia migración pero que también deben ser consideradas.

Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas.

La arquitectura de dos capas presenta dificultades en la migración, por su dependencia con el fabricante del entorno operativo, el alto grado de acoplamiento entre las aplicaciones del cliente y la base de datos, y los problemas de escalabilidad, ya que este tipo de arquitectura no presenta buena performance en su implementación en los sistemas donde crece el número de usuarios interactivos.

En consecuencia, debido a la fuerte dependencia entre la aplicación residente en el cliente y la base de datos, al cliente se lo denomina como ya se ha mencionado, “cliente pesado” (fat client).

4.3 Metodología propuesta de análisis lógico y físico del sistema migrado

Otro de los pilares básicos en el proceso de migración es el conocimiento de las nuevas reglas de negocio que se presentan así como las características tecnológicas necesarias para el sistema en la Web. A tal fin se presenta a continuación una metodología para conocer los detalles de la arquitectura y diseño del sistema migrado a la Web.

4.3.1. Adecuación de la especificación de requerimientos

a) Estudio preliminar

Descripción de los servicios generales que debe brindar el nuevo sistema, incluyendo aquellos que ofrecía el sistema anterior, con el detalle de las nuevas reglas de gestión y la tecnología a utilizar para la migración.

b) Reutilización de requisitos en el proceso de migración a la Web

Se propone el reuso de los requisitos que se trasladan al nuevo sistema y la identificación de los requisitos nuevos y aquellos sujetos a cambios en el nuevo sistema. A los fines de abordar el proceso posterior de reutilización de requisitos, resultó interesante el estudio comparativo de la metodología Web realizado por María José Escalona de la Universidad de Sevilla y Nora Koch de la Universidad de Munich (2002), que se sintetiza a continuación. El mismo plantea que el desarrollo de sistemas de aplicaciones Web agrupa una serie de características que lo hacen diferente del desarrollo de otros sistemas. Por un lado, hay que tener en cuenta que en el proceso participan roles muy diferentes de trabajadores: analistas, clientes, usuarios, desarrolladores, diseñadores gráficos, expertos en multimedia y seguridad, etc.

Por otro lado, la presencia en estos sistemas de una estructura de navegación a través de sus páginas, obliga a un desarrollo preciso en este aspecto que garantice que el usuario no se “pierde en el espacio navegacional del sistema”.

4.3.2. Adecuación de la descripción funcional

a) Revisión del modelo funcional

En función del rediseño del modelo de casos de uso de la nueva aplicación y de las reglas de gestión acordadas, deben introducirse modificaciones al esquema funcional, que se reflejará en el nuevo modelo de casos de uso y en las especificaciones que surjan del mismo.

b) Realización del modelo de casos de uso de la aplicación web

Las nuevas reglas de gestión que resultan de los requerimientos formales del nuevo sistema, plantean un nuevo modelo de casos de uso que atienda las modificaciones y agregados a la funcionalidad existente en la aplicación GUI.

En este nuevo diagrama de casos de casos de uso puede observarse el reuso de la funcionalidad de la aplicación anterior y, la incorporación de nueva funcionalidad a la aplicación migrada, acompañada de un reordenamiento de la misma por la inserción de la nueva tecnología.

4.3.3. Adecuación de la descripción de la dimensión estática

a) Revisión del modelo conceptual

Se revisa el modelo conceptual de la aplicación anterior a los efectos de establecer las nuevas entidades que lo componen, las que responden a la redefinición de las funcionalidades existentes, a las nuevas reglas de negocio relevadas y a las exigencias de la nueva tecnología a implementar.

Para superar las limitaciones existentes en el sistema a migrar, debe comprobarse la existencia de las nuevas reglas de gestión con la interacción entre todos los actores que juegan un rol importante en el sistema, estas reglas deben estar escritas y aprobadas por los responsables de las áreas involucradas.

Por otro lado, debe constituirse un equipo multidisciplinario como se plantea en el punto anterior, que disponga de los conocimientos necesarios de la nueva tecnología a aplicar.

4.3.4. Descripción de la interfaz del usuario en la aplicación Web

a) Análisis del modelo de interfaz del usuario

Las páginas Web originaron la aparición de las interfaces Web, interfaces gráficas de usuario con elementos comunes de presentación y navegación que pronto se convirtieron en estándares. Este tipo de interfaces deben servir de intermediarias entre los usuarios genéricos, no acostumbrados generalmente al uso de aplicaciones informáticas, y los sistemas de información y procesos transaccionales que corren por debajo, debiendo posibilitar la localización de la información deseada, el entendimiento claro de las funcionalidades ofrecidas, la realización práctica de tareas específicas por parte de los usuarios y la navegación intuitiva por las diferentes páginas que forman el sitio Web.

Buscando una homogeneidad entre los millones de páginas Web que existen actualmente en Internet, el diseño de las mismas ha evolucionado con el tiempo hacia un esquema general perfectamente definido, ofreciendo un conjunto de componentes gráficos y funcionales similares que hacen posible que para cualquier usuario que accede a un sitio Web, la comunicación pueda hacerse en forma rápida y efectiva.

La interfaz del usuario en la tecnología Web es el “browser”, por lo que la misma debe contemplar, de acuerdo al usuario al que va dirigido, un diseño arquitectónico basado en la funcionalidad y en la navegabilidad.

Una ventaja significativa en la construcción de aplicaciones web, es que soporten las características de los browsers estándar, independientemente de la versión del sistema operativo instalado en el cliente. Esto significa que, en lugar de crear clientes para Windows, Mac OS X, GNU/Linux, entre otros, la aplicación es escrita una vez y es mostrada de la misma forma en todos los entornos desde donde se accede.

Adicionalmente, la habilidad de los usuarios a personalizar muchas de las características de la interfaz (como tamaño y color de fuentes, tipos de fuentes, etc.) puede interferir con la consistencia de la aplicación Web.

Otra aproximación es utilizar Macromedia Flash o Java applets para producir parte o toda la interfaz de usuario. Como casi todos los browsers incluyen soporte para estas tecnologías (usualmente por medio de plug-ins), se pueden implementar con casi la

misma facilidad, aplicaciones basadas en Flash o Java. Estas tecnologías permiten más control sobre la interfaz, aunque pueden traer nuevas complicaciones por la incompatibilidad entre implementaciones de Flash o Java.

En este análisis de muestran las distintas pantallas que intervienen en la aplicación migrada, y se realiza un diagnóstico sobre su presentación y contenido.

b) Construcción del metamodelo

Se describe, utilizando el UML, la aplicación web centrada en el navegador (browser), que muestra la información de sitios a través de la gestión de protocolos de comunicación, la interacción de páginas subdivididas en frames para facilitar los enlaces y, la gestión de acceso a la base de datos a través de su interfaz.

Los navegadores Web utilizan protocolos de comunicación tales como el http, https y ftp para la gestión de información, a la vez que autentican servicios y gestionan cookies.

Los links permiten la navegación entre páginas. Los frames y otros componentes facilitan la organización y la interacción. Las páginas Web pueden ser simples páginas HTML o páginas del tipo estáticas o dinámicas. Mientras que el contenido de una página Web estática es fijo, el contenido de una página dinámica es determinado en tiempo de ejecución por el Server y puede depender de la información provista por el usuario a través de campos de entrada.

Un frame es un área rectangular en la Página donde la navegación puede tener su lugar de forma independiente. Además los diferentes frames en los que una página está descompuesta pueden interactuar entre sí, de modo que un link en una página cargada en un frame puede forzar la carga de otra página en un frame diferente.

En el caso de estudio desarrollado en el capítulo 5, se observa en detalle la implementación de este metamodelo.

4.3.5. Revisión de la arquitectura física y del software de base

a) Arquitectura física

La arquitectura física del sistema migrado, basado en la tecnología web, se implementa generalmente en un modelo multicapa donde los usuarios remotos, a través de su terminal y del navegador se conectan a Internet a través de distintos tipos de conexión: RDSI, ADSL, Cable, Satélite, Redes inalámbricas, como se observa en la figura 5.

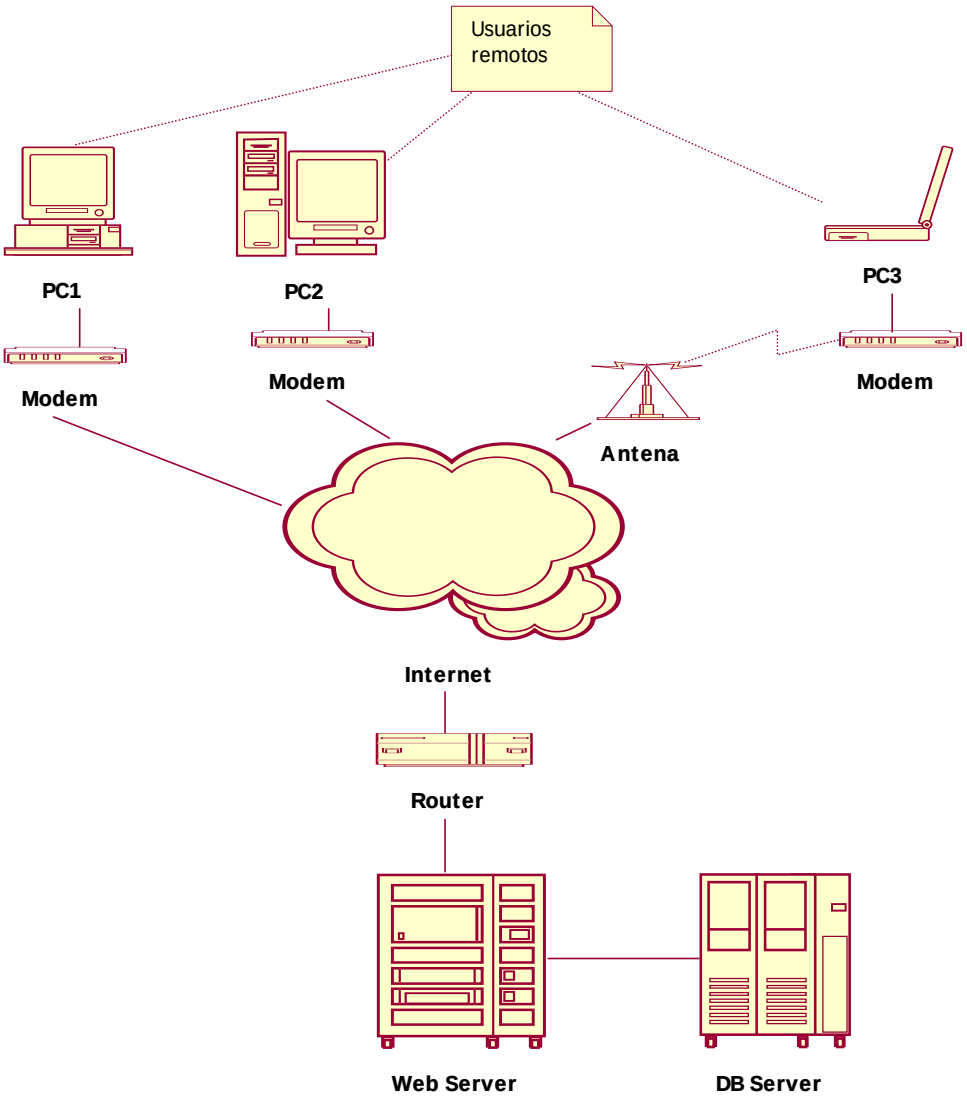


Figura 5 – Gráfico de componentes distribuidos de la aplicación Web

Internet funciona con la estrategia "Cliente/Servidor", lo que significa que en la Red hay ordenadores Servidores que dan una información concreta en el momento que se solicite, y por otro lado están los ordenadores que piden dicha información, los llamados Clientes.

Se ha establecido que en Internet, toda la información ha de ser transmitida mediante el Protocolo TCP/IP, y cada ordenador que se conecta a Internet se identifica por medio de una dirección IP. Sin embargo, se necesita nombrar de alguna manera los ordenadores de Internet, para poder elegir a cuál pedir información, lo que se logra por medio de los Nombres de Dominio. No todos los ordenadores conectados a Internet tienen un nombre de dominio, sólo suelen tenerlo los ordenadores que reciben numerosas solicitudes de información, o sea, los servidores, por lo que, los ordenadores cliente, los que consultan por Internet, no necesitan un nombre de dominio, puesto que ningún usuario de la Red va a pedirles información.

Los nodos de la red dedicados a encaminar los paquetes hacia su destino, eligiendo el enlace más adecuado en cada momento, reciben el nombre de enrutadores o ROUTERS, y al resto de los nodos se los denomina Host.

b) Arquitectura del Software de Base y diagrama de componentes

A continuación se presentan los distintos componentes que permiten realizar la visualización y transferencia de la información en un sistema distribuido en la Web, de acuerdo a la arquitectura física detallada en el punto anterior

Componentes:

- Sistema Operativo con interfaz gráfica.
- Conexión remota a servidores (local o de Internet).
- Browser de navegación.
- Componentes de la aplicación:
  - Páginas de hipertexto HTML.
  - Páginas para tratamiento de información (ASP, PHP, CLASS).
  - Códigos Script del lado del cliente (VBScript, JavaScript, JSP, etc.).
  - Plantillas de diseño (CSS).
- Componentes de comportamientos (OCX, DLL).
  - Manejo de imágenes (JPG, GIF, BMP, PNG).
  - Animaciones (AVI, SWF).
  - Sonido (WAV, MP3, etc.).
- Transferencia de información (XML, XSL, XLT, etc.).
  - Archivos de configuración y ocultos (CONFIG, WEBINFO).
  - Archivos inmodificables (PDF).
  - Facilidades de conexión a webcam.
  - Ensambles directos a email.

El diagrama de componentes consiste en la descripción de la vista física de las tres capas de la aplicación Web: cliente, servidor web y servidor de base de datos, donde es posible observar la realización de los componentes de la aplicación GUI en la aplicación Web.

El cliente accede al sistema de manera remota mediante un SO con interfaz gráfica a través de Internet. En esta capa, el componente principal es el Browser de navegación el cual despliega páginas HTML encargadas de la interfaz con el usuario.

Un servidor es una computadora que entrega a otras computadoras (los clientes), una información que ellos requieren bajo un lenguaje común, denominado protocolo. Por lo tanto al ver una página Web es porque el servidor les entrega una página HTML vía protocolo HTTP (HyperText Transport Protocol) o protocolo para la transmisión de hipertexto, a través de una conexión TCP/IP por el puerto 80.

Por lo tanto en el Servidor Web es donde se almacena la información estática accedida y/o las aplicaciones que la generan.

Los componentes del servidor son programas que sirven para realizar acciones más o menos complejas en las páginas ASP. Estas acciones pueden ser por ejemplo, el envío de correo electrónico, realizar upload de ficheros al servidor, conectar con una base de datos, etc. El lenguaje con el que se escribe (VBScript o Jscript) permite las funcionalidades básicas de cualquier lenguaje: trabajo con variables, tipos, estructuras de control y un juego de funciones (que en el caso de VBScript es bastante limitado).

En este componente se encuentran las políticas de acceso y concurrencia de clientes remotos al uso de la aplicación.

El servidor de base de datos es el Motor de Base Datos que contiene el servicio principal para la gestión de datos. Los componentes asociados son las Tablas donde se halla la organización de la información, las Vistas donde se encuentran las consultas más comunes, y los Procedimientos Almacenados donde están todos los Script para la gestión de datos.

Para acceder a una BD desde páginas web, se necesita un servidor web y una base de datos compatible con ODBC. Como servidor web podría utilizarse el Personal Web Server, que acompaña a los sistemas operativos Windows 9x, o Internet Information Services para NT o W2000, o cualquier otro servidor capaz de procesar páginas ASP.

4.4 Enfoque de Testing de Migración

El enfoque de Testing de migración que se presenta se basa en la corrección funcional o testing de caja negra, que implica que el sistema migrado implemente correctamente su especificación, apoyado en los casos de uso/casos de prueba diseñados para el sistema anterior. Se propone la confección de los casos de prueba del nuevo sistema, a partir de la reutilización de los casos de prueba del sistema anterior y el análisis de las actualizaciones correspondientes. Mediante la identificación temprana de los casos de uso, se puede comenzar con la planificación de las actividades de Testing y el armado de los casos de prueba.

4.4.1. Modelo de pruebas

Según Jacobson (Jacobson et al, 2003), el modelo de pruebas describe principalmente cómo se prueban los componentes ejecutables, y puede describir también otros aspectos del sistema, por ejemplo si la interfaz del usuario es utilizable y consistente y si el manual del usuario cumple su contenido.

Según puede observarse en la figura 6, el modelo de pruebas está compuesto fuertemente por el sistema de pruebas, el cual se integra por los casos de prueba, los procedimientos de prueba y los componentes de prueba, tal como se reflejan en la relación de agregación.

Un caso de prueba es un conjunto de entradas de prueba, condiciones de ejecución y resultados esperados, desarrollados para un objetivo concreto, tal como probar un camino especificado en un caso de uso o verificar que se cumple un requisito específico.

Un procedimiento de prueba es una especificación de cómo llevar a cabo la preparación, ejecución y evaluación de los resultados de un caso de prueba particular. Los procedimientos de prueba también pueden derivarse de los casos de uso. Los defectos hallados se analizan para localizar el problema. Tras la detección de estos problemas, se priorizan y se corrigen por orden de importancia (o de gravedad del defecto que se produjo).

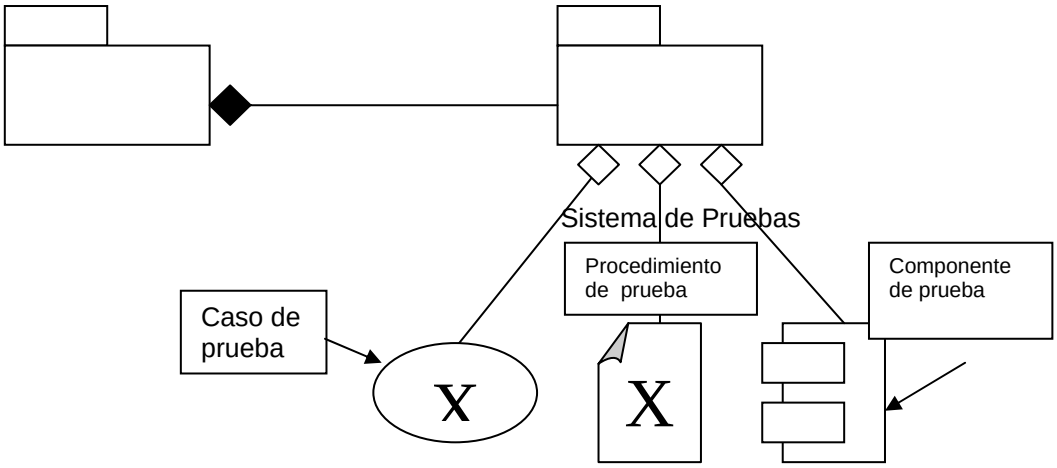


Figura 6 – Modelo de Pruebas

Un Caso de Prueba puede derivarse de -y por lo tanto seguir la traza de- un Caso de Uso del Modelo de Casos de Uso, en el caso de prueba de caja negra, como puede verse en la figura 7.

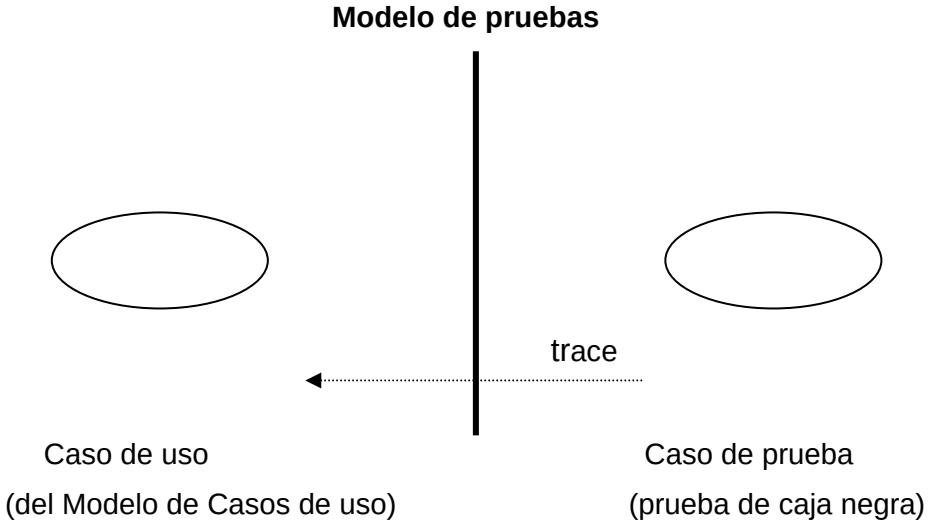


Figura 7– Trazabilidad de los casos de prueba

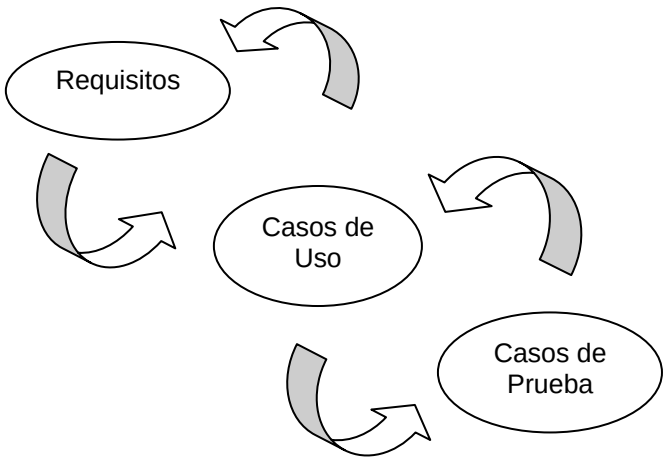


Figura 8 - Relación entre requisitos, casos de uso y casos de prueba

Según Evans (2002), los casos de uso deben probarse contra los requisitos. Así, este proceso constituye la validación del sistema, mientras que son los propios casos de uso los que deben utilizarse como autoridad de prueba para probar el código; es decir, para realizar la verificación del sistema. En la figura 8 es posible observar la relación entre requisitos, casos de uso y casos de prueba.

A los fines de modelar la aplicación que obtiene como resultados los casos de prueba, se utilizan las especificaciones funcionales representadas gráficamente por los casos de usos que darán lugar a las distintas condiciones de prueba. A continuación de la figura 9, se detallan las características de los componentes de la prueba funcional.

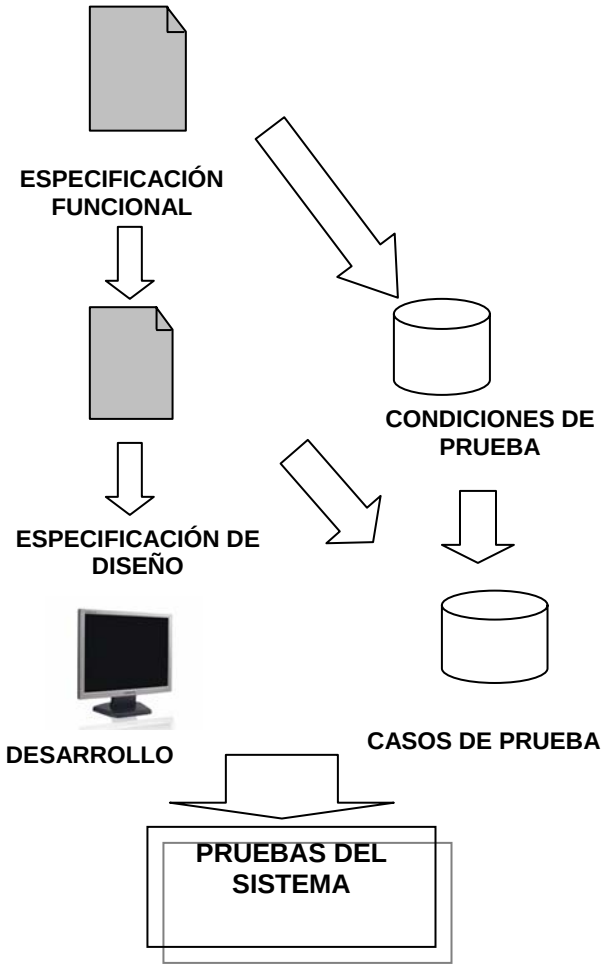


Figura 9 – Configuración de las pruebas del sistema

4.4.2. Características de las condiciones de prueba

Definir cuáles serán las condiciones a probar en un sistema es una etapa fundamental del proceso. La calidad del conjunto de condiciones elegido incidirá directamente en la calidad del producto obtenido.

Cada condición de prueba debe definir clara y brevemente una situación del sistema que se desea probar. En general, no debería llevar más de una oración redactarla. Si se precisan más, posiblemente pueda ser subdividida en distintas “subcondiciones”. Las condiciones son descripciones generales, no tienen el detalle de qué datos se utilizarán para probar; lo que se especificará en los casos de prueba, al igual que los resultados esperados de su ejecución.

4.4.3. Identificación de las condiciones de prueba

Al utilizarse la técnica de Casos de Uso para definir el comportamiento del sistema, los diferentes cursos de acción normales y alternativos de dichos casos dan origen a las condiciones de pruebas funcionales. Otra posible fuente de estas condiciones, puede ser la lista de requerimientos del sistema.

Para obtener las condiciones de prueba no funcionales, es necesario consultar una mayor variedad de fuentes. La primera fuente a utilizar deben ser los requerimientos no funcionales del sistema. Allí deben establecerse todos los requisitos que debe cumplir el sistema en lo referido a tiempos de respuesta esperados, volúmenes de información a manejar, configuraciones de software y hardware sobre las que debe correr, recuperación ante fallas, seguridad y otros aspectos dependiendo del tipo de sistema.

Otras fuentes de requerimientos no funcionales son los estándares de desarrollo del desarrollador y del cliente, estándares de pantallas, informes, formatos u otros. Si se dispone de la especificación de diseño del sistema, allí se pueden obtener condiciones especiales para validar la integración de los diferentes módulos, las implementaciones parciales, si el sistema será implementado en etapas, y otras.

Una vez obtenidas todas las condiciones de prueba, puede determinarse la necesidad de priorizar unas por sobre otras, ya que no se alcanzará a probar todas o a que algunos módulos son más complejos o críticos.

4.5. Casos de prueba

Una vez que se han definido las condiciones de prueba y se cuenta con la especificación de diseño del producto a construir, se comienzan a definir los casos necesarios para probar cada condición.

Un caso de prueba debe incluir qué se desea probar mediante su ejecución, esto es: la/s condición/es, la alternativa y la variación específica de dicha condición que se ejecutará, los datos de entrada que se introducirán al sistema, los pasos a seguir para ejecutarlo y las respuestas esperadas y obtenidas de la prueba del sistema.

Al ejecutarse el caso de prueba, debe registrarse el resultado del mismo. Si la ejecución del caso origina defectos, debe indicarse cuál fue el paso que los originó, para facilitar posteriormente las pruebas de regresión del defecto una vez que éste ha sido corregido.

La utilización de Condiciones y Casos de Prueba tiene muchas ventajas, entre las cuales pueden destacarse las siguientes:

a) Mayor cobertura: Derivar las condiciones y casos de prueba de las especificaciones de análisis y diseño de sistema, permite disminuir la posibilidad de que queden aspectos funcionales o no funcionales del producto sin probar.

b) Indicador de avance: Registrar el resultado de cada prueba realizada y conocer la cantidad de casos a probar, permite saber el grado de avance las pruebas realizadas.

c) Regresión más fácil: Registrar qué casos ocasiona cada defecto, permite reproducir el defecto más fácilmente y determinar qué otros casos deben ejecutarse durante la regresión.

d) Revisión de Especificaciones: Implica una revisión adicional de las especificaciones de análisis y diseño antes de comenzar el desarrollo del software.

4.5.1. Control de avance

Al registrar los casos de prueba, se pueden obtener mediciones del avance real de la prueba en el proyecto, lo que permite comparar la cantidad total de casos planificados para el sistema, con las cantidades reales de casos ejecutados y ejecutados sin error. Este indicador del avance o cobertura de la prueba, es uno de los más claros y objetivos que se pueden obtener.

4.5.2. Mantenimiento de condiciones y casos de prueba

Los casos de prueba deben poder ser reutilizados a lo largo de las diferentes pruebas realizadas al sistema durante su ciclo de vida, en sucesivas tareas de mantenimiento. Por eso, es necesario que ante cada cambio que sufra el sistema en su desarrollo o mantenimiento, se revisen los casos de prueba, para ver cuáles de ellos necesitan actualización y cuáles se seleccionarán para hacer pruebas de regresión luego de las modificaciones. Se debe tener en cuenta: qué casos se vuelven obsoletos, para qué casos cambia el resultado esperado y si es necesario incorporar nuevos casos.

4.6. Derivación de casos de prueba

A continuación se describe el proceso a seguir para derivar un caso de prueba a partir de un caso de uso. Obsérvese que, a pesar de que los casos de prueba desempeñan un rol pequeño en el desarrollo de planes de prueba, juegan un papel decisivo en el desarrollo de casos de prueba funcional. El desarrollo de casos de prueba es directamente conducido por los casos de uso, pero siempre teniendo en cuenta los requerimientos funcionales y las particularidades o reglas de negocio para que el enfoque de los mismos se traduzca luego en pruebas eficaces.

Para cada caso de uso se crea un conjunto de casos de prueba. Cada caso de prueba representa un escenario del caso de uso. Si el flujo de eventos del caso de uso está numerado hay que marcar cada paso del caso de uso con una letra: A, si se trata de un paso en el que se necesita la retroalimentación de un actor para realizarlo; S cuando es el sistema el que realiza la acción y una E cuando se trata de una condición de excepción, que no es parte del curso normal, como se esquematiza en la tabla 8.

Tabla 8 - Caso de Uso “Registrar Inscripción en Materia”

Paso	Acción	Curso Básico	Curso Alternativo
1	A	El alumno selecciona la opción “Inscripción”.	
2	S	El sistema permite visualizar la opción “Materias”.	
3	A	El alumno selecciona la opción “Materias”.	
4	S	El sistema valida que la fecha del día actual en el cual desea inscribirse el alumno se encuentre dentro de los períodos de inscripción a materias publicados. Y la fecha es válida.	3.1 La fecha en la cual el alumno desea inscribirse no es una fecha válida. 3.1.1 El sistema informa de dicha situación al alumno. 3.1.2 Se cancela el caso de uso.

5	A	El alumno selecciona la/s materias a inscribirse	
Etc.			

La descripción del caso de uso debe estar organizada en pasos mediante la numeración de éstos en el margen. Una vez que los pasos estén numerados se puede crear el árbol dibujando los caminos a lo largo del caso de uso. El siguiente paso consiste en dibujar un diagrama de grafos de los caminos del caso de uso, tal como se observa en la figura 10.

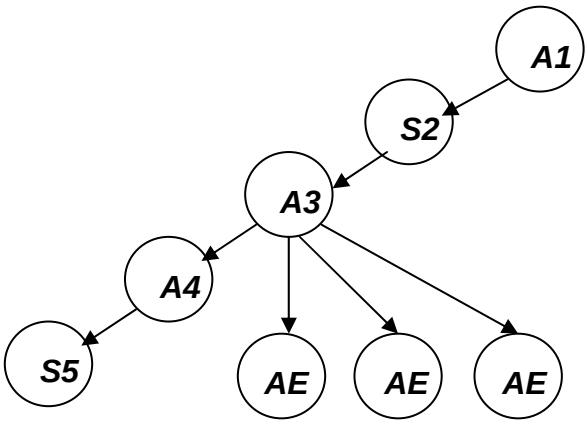


Figura 10 – Grafos de los caminos de un caso de uso

Es conveniente crear al menos un caso de prueba para cada camino posible. Los casos de prueba positivos son los que llevan al éxito, es decir que se cumple el objetivo del caso de uso. Los casos de prueba negativos son aquéllos que llevan a un resultado de fracaso.

Para crear casos de prueba a partir de casos de uso se utiliza el procedimiento inverso al que se utilizó para unir escenarios en un caso de uso. Se reemplazan las entidades abstractas del caso de uso con instancias específicas. Este reemplazo se realiza en las precondiciones, resultados esperados, flujo de eventos y excepciones. Las precondiciones corresponden a la sección de inicio del caso de prueba y los resultados esperados a la sección de búsqueda de defectos.

A continuación, y a título de ejemplo, se presenta la realización de dos casos de prueba para un Caso de Uso de un sistema académico denominado: “Registrar Inscripción a Materia”.

Para el primer caso de prueba, el alumno se encuentra dentro del período de inscripción, no adeuda abonos y tiene materias para cursar en el período de la consulta, y además se tienen en cuenta las siguientes precondiciones:

- Que el alumno se haya identificado correctamente al sistema de autogestión.
- Que el alumno se encuentre activo para una carrera seleccionada.

Para el segundo caso de prueba, el alumno no se encuentra al día con los abonos correspondientes a la carrera, y se tienen en cuenta las siguientes precondiciones:

- Que el alumno se haya identificado correctamente al sistema de autogestión
- Que el alumno se encuentre activo para una carrera seleccionada.

Se observa que las precondiciones coinciden en ambos casos de prueba; pero en las condiciones previas del segundo caso de prueba, se produce una condición que afecta el resultado esperado (adeuda abono).

El modelo de Planilla de Casos y Condiciones de Prueba presentado se muestra a modo de ejemplo, no necesariamente es el modelo de planilla que será utilizado, el cual puede variar según convenga en el tipo de proyecto.

Cada paso en un caso de prueba debe tener un estado que indique si se llegó al resultado esperado (OK) o no (NO OK). El estado general del caso de prueba está determinado por la suma de los estados de los pasos del mismo.

En caso de no coincidir el resultado obtenido con el esperado (Estado NO OK), se debe evaluar el alcance del problema para realizar la registración del tipo y dar prioridad al error producido para su derivación y posterior corrección. El error puede afectar solamente a un paso particular o a todo el caso de prueba, con lo cual se debería ejecutar la prueba en su totalidad y no desde el paso que falló.

Tabla 9 – Caso de prueba en el que el alumno esté activo y no adeude abono

Registrar Inscripción a materia. El alumno está dentro del período de inscripción, no adeuda abonos y tiene materias para cursar en el semestre de la consulta						
Precondi-ciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-05-2006				
	Resultado esperado	Que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado Esperado**	**Re-sultado obte-nido**	**Códi-go de error**	**Obser-vacio-nes**
1	El alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El sistema permite visualizar la opción “Materias”.	OK		
2	El alumno selecciona la opción “Materias”.	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en el cual desea inscribirse el alumno se encuentre dentro de los períodos de inscripción a materias publicados. Y la fecha es válida. El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el Alumno se encuentra al día. El Sistema despliega una lista con las distintas materias que el alumno puede cursar ese semestre.	OK		
3	El alumno selecciona la/s materia/s a inscribirse.	Seleccionar materia: "Auditoria y Evaluación de Sistemas"	El sistema solicita al alumno que seleccione el curso en el cual realizará la materia.	NO OK		

4	El alumno selecciona el curso en el que desea inscribirse.	Seleccionar: Curso	El sistema solicita al alumno si desea confirmar la Inscripción.	OK		
5	El alumno confirma la inscripción.	Presionar "Aceptar"	El sistema registra la inscripción y envía al mail institucional del alumno los comprobantes de inscripción ccorrespondientes.	OK		

La repetición de la ejecución de un caso de prueba debe registrarse en el mismo como un nuevo ciclo, incrementándose en 1 el ciclo en la planilla anterior. El administrador de pruebas debe realizar el control estadístico de los ciclos de prueba por cada caso ejecutado. Ejemplos de casos de prueba con las consideraciones anteriores pueden visualizarse en las tablas 9 y 10.

Tabla 10– Caso de prueba en que el alumno esté activo y adeude abono

Registrar Inscripción a materia. El alumno no se encuentra al día con los abonos correspondientes a la carrera						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-05-2006				
	Resultado Esperado	Que el Alumno realice la inscripción de una o varias materias				
	Resultado Obtenido	OK				
	Código de Error					
Id Paso	**Descripción**	**<Valor>**	**Resultado Esperado**	**Resultado Obtenido**	**Código de error**	**Observaciones**
1	El Alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El Sistema permite visualizar la opción “Materias”.	OK		
2	El alumno selecciona la opción “Materias”.	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en el cual desea inscribirse el alumno se encuentre dentro de los períodos de inscripción a materias publicados. Y la fecha es válida. El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno debe pagar los abonos correspondientes a la carrera. El sistema informa de dicha situación al alumno.	OK		

4.7. Consideraciones especiales para las pruebas

Hay numerosos aspectos adicionales que deben ser considerados en la planificación, definición, ejecución y evaluación de pruebas de sistemas. Entre otros, reviste especial importancia la definición del entorno de pruebas, la especificación de rangos de validez en las variables de entrada, la clasificación de los errores y la definición de las prioridades en el tratamiento de estos errores.

4.7.1. El Entorno y los datos para las pruebas

Las pruebas se deben realizar en un entorno aislado, asegurando que los datos sean extraídos de la realidad o que sean semejantes a la realidad y que, en la medida de lo posible, cubran la totalidad de casos previstos para realizar las pruebas.

Tal como se mencionó, estos datos deben provenir del sistema base, es decir, deberían ser obtenidos a través de la migración de los datos existentes, y no ser generados en forma manual para las pruebas. Esto último puede introducir errores "humanos" en los datos que perjudiquen la consistencia de las pruebas. En caso de no poder contar con datos "reales" se deberá tener en cuenta este factor de riesgo al evaluar los resultados de las pruebas.

Es conveniente generar algún tipo de documentación (puede ser un checklist) donde se especifiquen los pasos necesarios para la creación de un entorno adecuado en el que realizar el desarrollo y las pruebas. Dicha documentación también servirá de base para la posterior implementación del sistema.

Una aplicación de los conceptos vertidos en este capítulo se desarrolla en el capítulo 5 al hacer la aplicación de casos de prueba al caso de estudio de autogestión.

4.7.2. Rangos de prueba

A los fines de proveer una cobertura de prueba mayor, es necesario analizar las entradas al sistema y las circunstancias en las que las entradas van a producir un efecto adverso en el sistema.

Para entender dónde el sistema no tiene control sobre su información, se debe observar nuevamente los pasos en los que el actor provee de información al sistema. Es necesario ejecutar todos los pasos para determinar si la lógica está validando correctamente las condiciones límite. También se deben probar condiciones cuyo resultado esperado no está considerado explícitamente en el modelo de casos de uso.

Las verificaciones del rango de prueba es otra área en la que se puede validar los puntos de decisión. Las verificaciones de rango consisten en darle al sistema datos potencialmente dañinos en ciertos puntos de decisión para verificar la respuesta del mismo. Por ejemplo, "¿se puede brindar un crédito al 0% de interés?" "¿Qué pasa con tasas "de interés negativas? Las reglas de negocio juegan un papel importante a la hora de verificar que la información devuelta es la correcta.

Para muchas validaciones sobre una condición específica de un caso de prueba, se puede tomar la misma como "de múltiples valores" y repetirla agregando a cada una el sufijo a, b, c, etc. para cada variación del valor de la condición. El resto de las condiciones y precondiciones son iguales, con lo cual estaríamos evitando así generar "n" casos de prueba los cuales diferirían solamente en el valor de dicha condición.

4.7.3. Clasificación de errores y tipificación de sus prioridades

La clasificación de los errores que puede darse durante la ejecución de los casos de prueba es presentada en la tabla 11.

Tabla 11 – Tipos de error en la ejecución de las pruebas

Tipos de error		
Invalidante	Es todo acontecimiento que impide avanzar con la prueba	No cumple con la funcionalidad
Grave	Es todo tipo de defecto con el cuál no se dará la aceptación pero no impide seguir con la prueba	No cumple con la funcionalidad
Medio	Es todo tipo de defecto con el cuál se puede dar la aceptación y el mismo no impide seguir con la prueba	Cumple con la funcionalidad pero no de la forma especificada
Leve	Defecto no crítico o de presentación que no se vincula en forma directa con requerimientos del usuario	Cumple con la funcionalidad
Mejora / Sugerencia	No son errores. Son agregados y/o modificaciones sugeridas que contribuyan mejorar u optimizar el cumplimiento del requerimiento que se está evaluando	Cumple con la funcionalidad
Consulta	Formalidad para solicitar alguna aclaración sobre alguna duda respeto al resultado obtenido en la prueba	Cumple con la funcionalidad

Asimismo, es necesario realizar la tipificación de la prioridad que se le debe dar al error encontrado, teniendo en cuenta que los errores invalidantes tendrán máxima prioridad ya que los mismos hacen imposible avanzar con las pruebas El resto puede variar su prioridad según la celeridad con la que se necesite solucionar el inconveniente. Dicha prioridad la determinará el testeador, pero la misma estará sujeta a confirmación o modificación por parte de los supervisores o líderes de proyecto. La tipificación a utilizar es la que se presenta en la tabla 12.

Tabla 12 – Prioridad de corrección de errores

PRIORIDAD		
1	Inmediata	Deberá ser corregida de inmediato
2	Urgente	Deberá ser corregida tan pronto como sea posible
3	Media	Deberá corregirse rápido
4	Baja	Deberá corregirse pero no afecta el funcionamiento

Una vez corregido el error se procederá a realizar un nuevo ciclo de prueba que comenzará desde la búsqueda o generación de casos (valores para cada condición) o desde la ejecución misma, de acuerdo a si el valor de prueba es reutilizable o no.

En caso de que la corrección del error implique algún tipo de modificación en los flujos del caso de uso, habría que revisar nuevamente los mismos y de ser necesario también modificarlos, lo que determina realizar nuevamente todas las pruebas. Sin embargo, si el impacto es sólo sobre la condición que falló se continuará la prueba desde dicha condición.

4.7.4. Definición de ciclo y versión

En este punto, resulta importante definir a qué se hace referencia cuando se habla de *versión* y *ciclo*. Tales conceptos pueden diferenciarse considerando quién es la persona que genera la solicitud de cambio del programa:

- Si el programa se modifica a raíz de un requerimiento del usuario, el programa tiene una nueva versión.
- Si el programa se modifica a raíz de un defecto detectado por un testeador, el programa además de modificar su versión, generará un nuevo ciclo de testeo.

En algunos casos, un nuevo ciclo puede generar un testeo completo, debido a que el defecto encontrado altera o cambia condiciones/casos de prueba o necesita una nueva corrida completa. En otros casos, el programa puede ser probado solamente en una de sus opciones o funciones.

CAPÍTULO 5. MIGRACION DE UN CASO DE ESTUDIO

5.1 Introducción

A los fines de comprobar el desempeño de la metodología propuesta para el testeo de aplicaciones distribuidas a entornos Web, se la aplicó a un Caso de Estudio, y con este fin se seleccionó el *Sistema de Autogestión de Alumnos* del *Instituto Universitario Aeronáutico (IUA)* que tiene su sede en la Ciudad de Córdoba. Este Instituto tiene dos Facultades, que son la de Ingeniería y la de Ciencias de la Administración. El sistema forma parte de los recursos y herramientas administrativas con que cuenta la universidad, destinado a la gestión de alumnos y distribuido en las distintas unidades académicas de ambas Facultades.

Se consideró que el Sistema seleccionado es apropiado para esta evaluación por presentar las siguientes características:

a) Se trata de un sistema al que podría asignársele una complejidad y dimensión “media”, que cubre toda la gestión de alumnos de un instituto universitario.

b) Fue originalmente desarrollado, implementado y utilizado bajo un entorno cliente-servidor.

c) Varios años después el sistema fue migrado a la Web.

d) Se dispone de información fragmentada e incompleta sobre el análisis y diseño del sistema original y de su migración a la Web.

e) Se dispone de testimonios de miembros de los equipos que participaron del diseño del sistema original y de su posterior reconversión.

f) El sistema original fue utilizado intensamente y lo mismo ocurre con su versión actual reconvertida a la Web, por lo que se dispone de testimonios de los usuarios de ambos sistemas.

Por todo lo expuesto, al sistema adoptado como Caso de Estudio se lo podría tipificar como un “Sistema heredado” que completó su ciclo de reconversión a una nueva plataforma y tecnología.

5.2 Aplicación de la metodología para el análisis del sistema a migrar

La descripción del sistema sirve como punto de partida para comprender los requerimientos del sistema.

5.2.1 Reconstrucción de la especificación de requerimientos

a) Estudio preliminar

a.1 Servicios prestados por el sistema

Realiza el seguimiento y administración de cada alumno, desde que ingresa como aspirante, mientras es alumno regular, hasta que arriba a la condición de egresado, con la obtención de su título. La gestión académica del instituto en su conjunto y de las unidades académica en particular, permite mediante el sistema, el registro de todas las actividades académicas como apoyo de las acciones operativas y de toma de decisiones, para producir datos académicos, de uso interno y con otros organismos.

a.2 Relación con otros sistemas

Biblioteca, Librería, Aranceles.

a.3 Alcance del SI

El sistema cubre los procedimientos relativos a la gestión de los alumnos de la Facultad de Educación a Distancia:

- Matriculación en carrera.
- Inscripción en materias
- Inscripción en exámenes parciales y finales.
- Recepción, devolución y registro de actividades parciales obligatorias.
- Emisión de constancias de alumno regular, examen parcial/final.
- Mantenimiento de carrera:
  - Suspensión y baja.
  - Alta y reinscripción.
  - Cambio de carrera y modalidad de carrera.
- Consultas en general:
  - Notas parciales/finales.
  - Estado de actividades obligatorias.
  - Fechas de exámenes finales.
  - Horarios de tutorías.
- Correlatividades.
  - Habilitación para rendir exámenes.
  - Materias a cursar.
  - Consulta de situación financiera

Quedan fuera del alcance de este proyecto los procedimientos implicados en la gestión de equivalencias, biblioteca, librería y tesorería.

b) Características generales y reglas de gestión

b.1 Perfiles

El sistema puede administrar un conjunto de perfiles que son representativos de los distintos usuarios que acceden al mismo. Existen perfiles ya predeterminados como el del alumno y el del docente, y la posibilidad de definir uno específico para un usuario puntual.

b.2 Auditoria

La auditoria es una función incorporada al sistema que permite obtener en forma automática un registro de la actividad que realizó cada usuario en el sistema de autogestión.

b.3 Programación

La herramienta usada para su programación fue Power Buillder

b.4 Base de datos

Microsoft SQL Server

b.5 Listado de procesos

- El sistema permite al alumno hacer *consultas e inscripciones* tanto en materias como en exámenes en forma remota, en terminales distribuidas en la universidad.
- La *inscripción en materias* tiene como requisito previo la *inscripción en carrera*, trámite que el alumno realiza personalmente en la oficina de alumnos, al completar una ficha de inscripción y abonar la matrícula.

- La solicitud de equivalencias y certificados debe hacerse en forma manual a través de la oficina de alumnos.
- El alumno al inscribirse en una materia selecciona el curso (de acuerdo al horario, modalidad y profesor). Esta información es suministrada al sistema por los departamentos académicos, de los cuales existe uno por cada carrera de la universidad.
- El alumno realiza el pago de cuotas en una oficina destinada para ello, puede hacerlo también en las entidades bancarias indicadas para tal fin, situación que es controlada por el sistema para la inscripción en exámenes y materias, a través de la consulta a un sistema externo denominado *Aranceles*.
- El alumno puede realizar consultas sobre su situación académica y financiera a través del sistema de autogestión. Si el alumno cambia de carrera o curso, o modifica sus datos personales debe hacerlo a través de la oficina de alumnos.
- Las notas de parciales pueden ser cargadas por los docentes, no así las notas finales de las materias.
- El plan de estudios establece el régimen de cursado y correlatividades.

Con la recopilación de los modelos de procedimientos confeccionados por el personal que intervino en el desarrollo del sistema anterior y las entrevistas realizadas, se logró establecer las reglas de gestión vigentes al momento de desarrollo del sistema de autogestión anterior.

b.6 Reglas de gestión

Estas reglas aseguran que la actividad de la organización se lleva a cabo de acuerdo a restricciones impuestas desde el entorno (leyes o normas) o desde dentro de la propia organización. En nuestro caso de estudio, en el momento del desarrollo del sistema anterior, no existía un reglamento del alumno que contemplara estas normas, pero, la vigencia del plan de estudios de cada carrera, aprobado por el Ministerio de Educación, imponía de por sí, cuanto menos, el marco normativo del cursado de las materias.

El resto de las reglas surgió de la actividad propia de la institución y de la necesidad de sistematizar y minimizar el régimen de excepciones, tanto en las inscripciones como en el pago de los aranceles, y se sintetizan a continuación:

- Un alumno puede inscribirse en más de una carrera y en más de una materia de acuerdo al régimen de correlatividades.
- El alumno deberá regularizar la materia dentro de los plazos establecidos en cada curso.
- Cada curso tiene una fecha de inicio y de finalización que deberá estar disponible en el sistema un plazo no menor a 15 días antes de la inscripción.
- Para inscribirse en materias o en exámenes el alumno deberá tener abonadas las cuotas de pago con una tolerancia de una cuota impaga a la fecha de inscripción.
- Toda materia consta del régimen de regularización y aprobación con examen final.
- El alumno puede eliminar por autogestión su inscripción en una o más materias antes de comenzar el cursado de la/s misma/s.
- Para rendir examen final el alumno deberá tener regularizada la materia, y podrá inscribirse para rendir 48 horas antes de la fecha de examen, completando el formulario de inscripción.

- Por cada materia puede haber más de un curso y cada curso tiene un docente responsable asignado y uno o ningún ayudante de cátedra.
- No se efectúa carga ni control de asistencia.
- La cantidad de aplazos en los tres primeros años de la carrera no debe superar los diez.
- La cantidad de aplazos por materia no debe superar los seis.
- Un alumno pierde la regularidad de la materia al tercer aplazo.

5.2.2 Descripción de la dimensión funcional

a) Elaboración del diagrama de contexto (DC)

En el caso de estudio del sistema de autogestión, este diagrama sirve para establecer las entidades que suministran y obtienen información del sistema a través de sus interfaces y cuál es el tipo de información que circula entre el sistema y las mismas.

Como se observa en la figura 11, los agentes o entidades externas que interactúan con el sistema en una primera aproximación (burbuja nivel 0), son: alumnos, docentes y Departamentos Académicos. Existen otras entidades como los aranceles que permiten el cursado y son materia de consulta permanente, y, la interfaz Impresora mediante la cual el alumno obtiene su comprobante de inscripción.

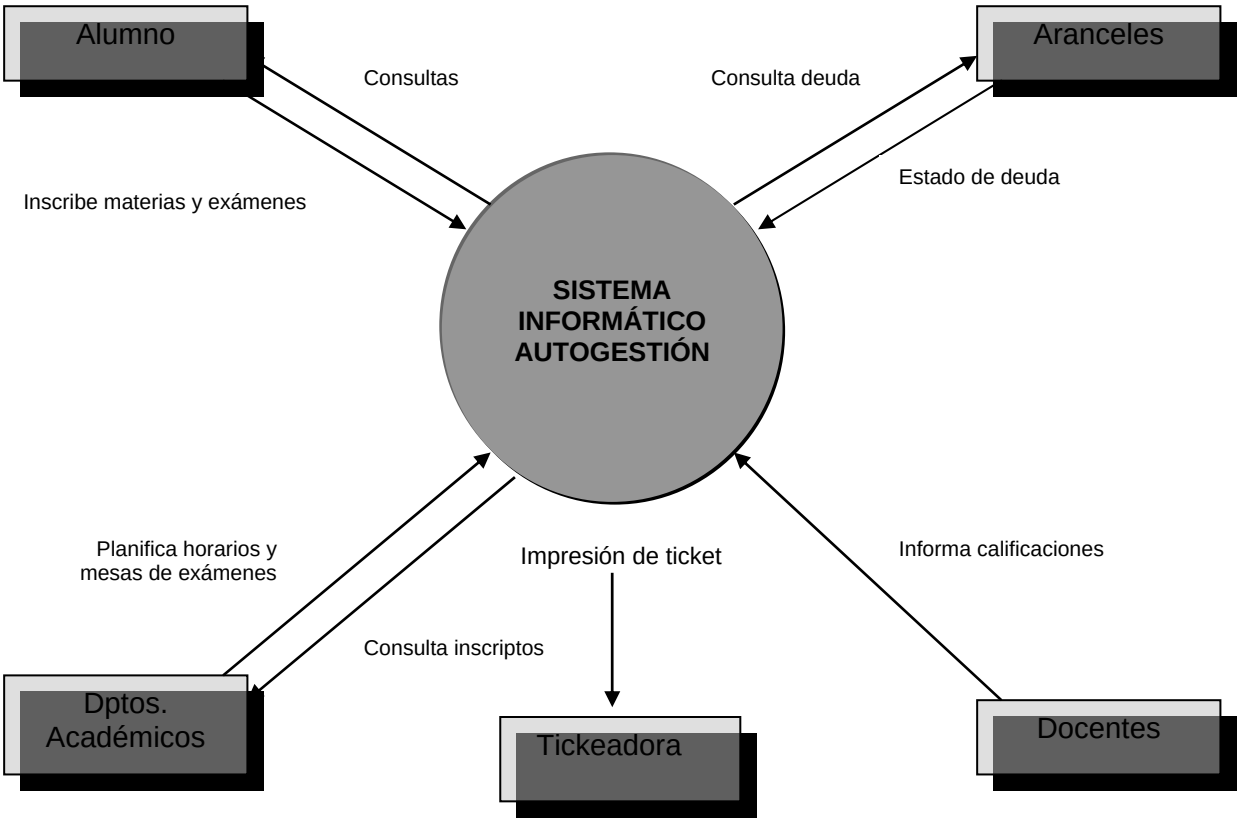


Figura 11 – Diagrama de contexto del Sistema Informático de Autogestión

b) Análisis de comportamiento del sistema

A los fines de continuar con el uso de la metodología estructurada, se realiza un análisis del comportamiento de los subsistemas “Identificación de usuario” y “Menú de opciones”. Para cumplir con este objetivo se utiliza la técnica de escenarios para cubrir las posibles interacciones del sistema con los usuarios, utilizándose para ello una descripción del flujo de eventos que figura a continuación.

b.1 Lista de eventos externos

1. Iniciar sesión Sistema Autogestión
2. Identificación del alumno
3. Mostrar menú de opciones
4. Selección de opciones
5. Consultar
6. Actualizar
7. Finalizar
8. Cancelar

b.2 Escenarios

b.2.1 Escenario 1

1. Usuario alumno visualiza el menú de inicio de sesión.
2. Ingresa DNI.
3. El sistema valida el DNI.
4. Si el usuario es correcto solicita la contraseña, caso contrario volver a punto 1.
5. Si la contraseña es incorrecta volver a punto 3.
6. El sistema muestra el Menú de Opciones.
7. El usuario puede presionar la tecla ESC para abandonar menú y volver a la función anterior.

b.2.2 Escenario 2

1. El usuario alumno selecciona una opción
2. El sistema valida la opción
3. Si la opción es inválida el sistema muestra mensaje de error
4. El usuario puede presionar la tecla ESC para abandonar menú y volver a la función anterior
5. El usuario selecciona la opción de consulta
6. El usuario selecciona la opción de inscripción
7. El usuario selecciona la opción de modificación de contraseña
8. El usuario selecciona la opción de fin para finalizar opciones

b.2.3 Escenario 3

1. El usuario alumno selecciona la opción de consulta.
2. El sistema muestra las distintas opciones de consulta.

3. El usuario selecciona la opción deseada.
4. El sistema consulta las distintas tablas de acuerdo a la opción consultada
5. Si el usuario selecciona Actividades Obligatorias muestra la calificación de las mismas o sino han sido presentadas.
6. El usuario selecciona Exámenes Finales muestra las fechas de exámenes finales de las materias en las que está inscripto.
7. El usuario selecciona la opción de Inscripción en materias muestra las materias en las que está inscripto.
8. El usuario selecciona la opción de Equivalencias muestra las materias que se le han considerado por equivalencia.
9. El usuario selecciona la opción de Situación Arancelaria se muestra el estado de pagos y las cuotas adeudadas.
10. El usuario selecciona la opción de fin para finalizar la consulta.

b.2.4 Escenario 4

1. El usuario alumno selecciona la opción de actualizar
2. El sistema muestra las distintas opciones de actualizar:
3. El usuario selecciona la opción deseada
4. El sistema consulta las distintas tablas de acuerdo a la opción consultada
5. Si el usuario selecciona la opción Inscripción en materias el sistema muestra las materias en las que puede inscribirse de acuerdo al régimen de correlatividades y el alumno seleccionará la/s materia/s a inscribirse. El sistema emitirá el ticket de inscripción
6. Si el usuario selecciona la opción Inscripción en Exámenes Finales, el sistema muestra las fechas de exámenes finales de las materias en las que puede inscribirse según condiciones de regularidad y el alumno seleccionará las mesas de examen. El sistema emitirá el ticket de inscripción
7. El usuario selecciona la opción de Eliminar la Inscripción en materias muestra las materias en las que está inscripto para que seleccione la que desea eliminar.
8. El usuario selecciona la opción de fin para finalizar la actualización.

b.2.5 Escenario 5

1. El usuario alumno selecciona la opción de Modificar PIIN.
2. El sistema solicita que el usuario ingrese código postal y fecha de nacimiento para verificar los datos personales.
3. Si los datos ingresados son correctos el sistema solicita al usuario la contraseña actual.
4. El sistema verifica la contraseña y si ésta es correcta, solicita la nueva contraseña y luego su reingreso para confirmarla.
5. Si el reingreso es correcto el sistema almacena la nueva contraseña.
6. El usuario puede cancelar la operación si no desea modificar su contraseña.

b.2.6 Escenario 6

1. El usuario selecciona la opción FIN para abandonar el sistema.

b.3 Requisitos no funcionales

- El sistema debe configurarse en cada PC dentro del ámbito del Instituto Universitario Aeronáutico destinada para tal fin, junto con la impresora para la impresión de los comprobantes de inscripción.
- El tiempo de espera máximo para cada transacción no debe superar los 6 segundos.
- Debe solamente habilitarse el teclado numérico.
- Cualquier modificación en el sistema debe reconfigurarse el mismo en la PC

c) Construcción del diagrama de flujo de datos - DFD

En la figura 12, que se muestra a continuación, puede observarse cómo cada proceso (representado por una burbuja) puede refinarse en otros DFD's de menor nivel para mostrar un mayor detalle en el flujo de datos y procesos.

Ward y Mellor (1985) ampliaron la notación básica del modelado de flujo de datos para adaptarlos a las demandas de tiempo y, en especial, sobre la información de control y su procesamiento, para cuya representación se utiliza la misma notación pero con trazos discontinuos.

En el caso de estudio pueden observarse las burbujas de mayor nivel con las funcionalidades principales de acuerdo a lo detallado en el punto anterior, y, las burbujas de menor nivel que extienden la funcionalidad de las anteriores. En forma de intermediarias entre ambas figuran las burbujas de control que verifican la correcta transformación de los flujos de datos.

En la figura 12 se presentan también los almacenes de datos que interactúan con las burbujas.

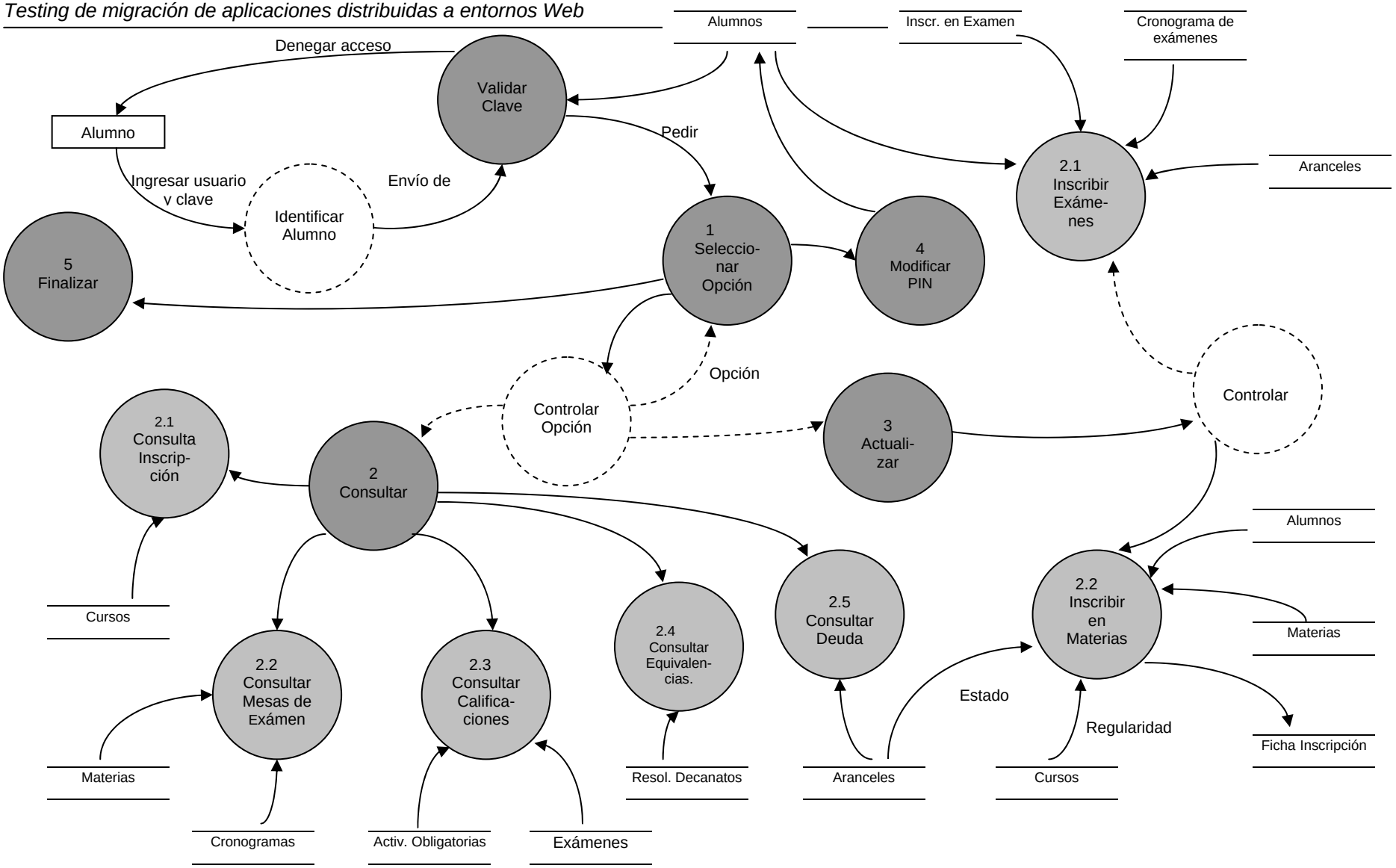


Figura 12– Diagrama de Flujo de Datos del Sistema Informático de Autogestión

d) Realización del modelo de casos de uso de la aplicación anterior

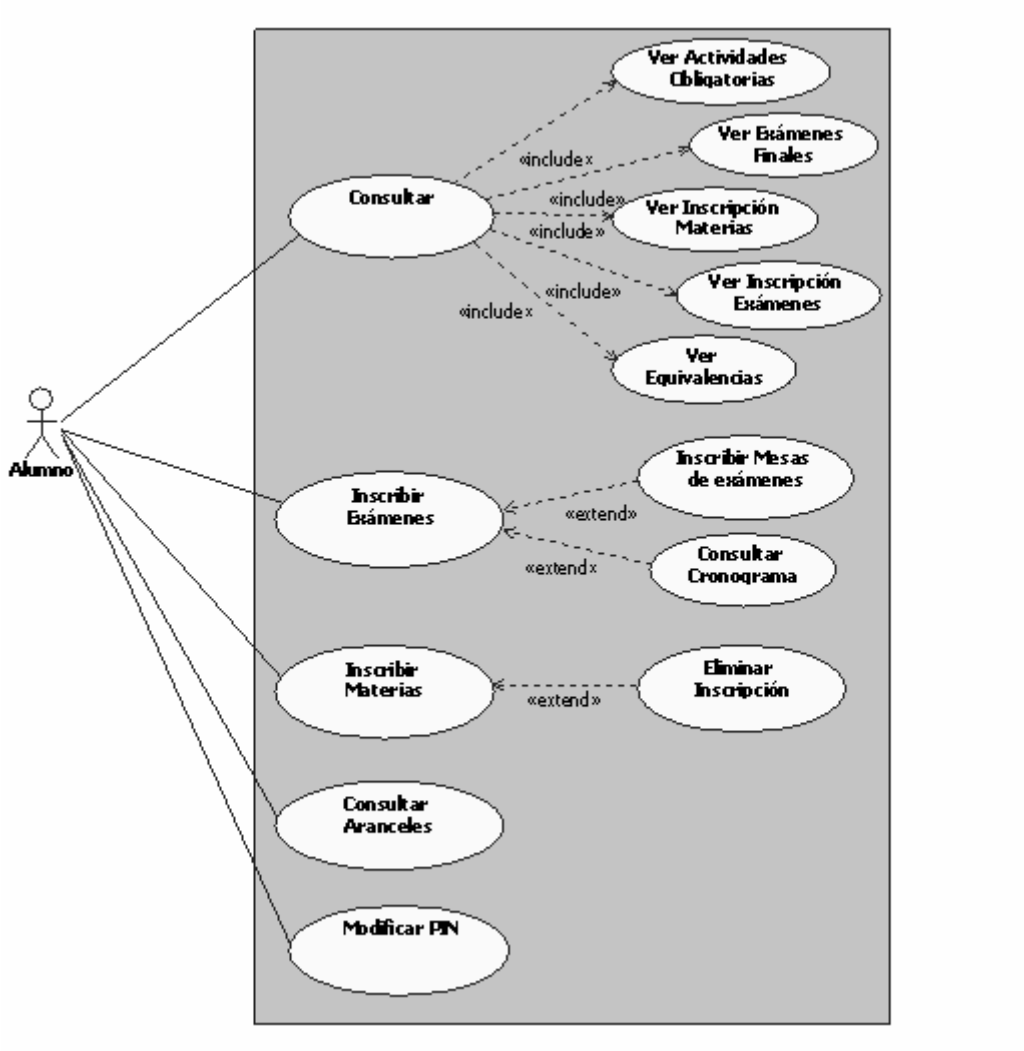


Figura 13– Modelo de casos de usos aplicación distribuida

5.2.3 Descripción de la dimensión estática del sistema anterior

a) Reconstrucción del modelo conceptual del sistema Autogestión

En la figura 14 se muestra un esquema conceptual de datos perteneciente al sistema de autogestión. Este modelo se obtuvo de observar los diagramas de entidad relación (DER) existentes utilizando una visión orientada a objetos, ya que se rescataron las principales entidades (u objetos) que se muestran en la figura, teniendo en cuenta además los distintos estados por los cuales atravesaban los mismos, al invocarse las diferentes funciones (o métodos) del sistema, por ejemplo el alumno pasa de estado activo a inscripto una vez que completó su ficha de inscripción (en exámenes o materias).

Básicamente, al migrarse al sistema en Web, la base de datos relacional no cambió su estructura de tablas, aunque sí se incorporaron distintos servicios Web que se detallan en las páginas siguientes.

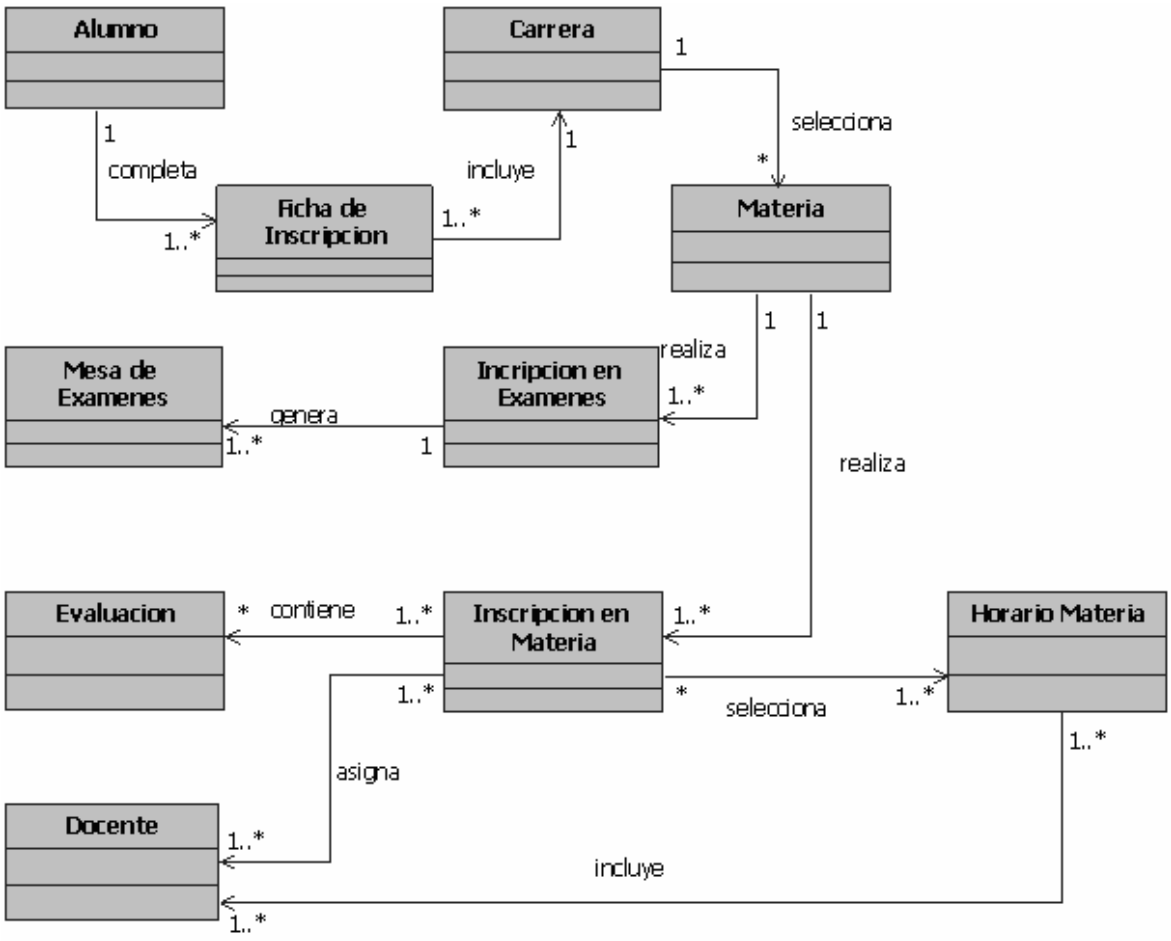


Figura 14– Modelo conceptual del sistema de autogestión

5.2.4 Descripción de la interfaz del usuario

a) Análisis del Modelo de Interfaz del Usuario

En el caso de este sistema de autogestión, la interfaz fue desarrollada con pantallas o ventanas con menús de opciones estrictamente jerárquicos, los que proporcionan al usuario una lista de las selecciones disponibles. El usuario no necesita conocer el sistema, pero sí necesita saber qué tarea debe ser realizada.

El espacio de diseño de la interfaz es de dos dimensiones. En el caso de este sistema, sólo se permite la utilización del teclado numérico y teclas de movimiento del cursor, así como también las teclas ENTER y ESC, además de FIN. El color añade una nueva dimensión a la facilidad de uso de la pantalla, para atraer la atención del usuario al facilitar la separación de componentes de la pantalla y acentuar las diferencias.

La crítica a realizar a esta interfaz se presenta en el menú principal, en el que no existe la suficiente separación jerárquica entre las opciones de consulta y las de actualización. Por ejemplo, Eliminar inscripción en Materia, debería estar dentro del grupo de actualización y, las consultas dentro de la misma opción de Consulta, por ejemplo: cronograma de actividades y consulta de situación arancelaria figuran como una opción nueva dentro del Menú de Opciones, y no, dentro de las consultas.

A continuación, en las siguientes figuras, presentamos las distintas pantallas del Menú de Autogestión del alumno, desde el ingreso al sistema, menú de opciones, entre las que pueden seleccionarse consultas, Inscripciones en exámenes y materias, Listado

de mesas de examen, Cronograma de actividades, consulta de situación arancelaria y Modificación de PIN.

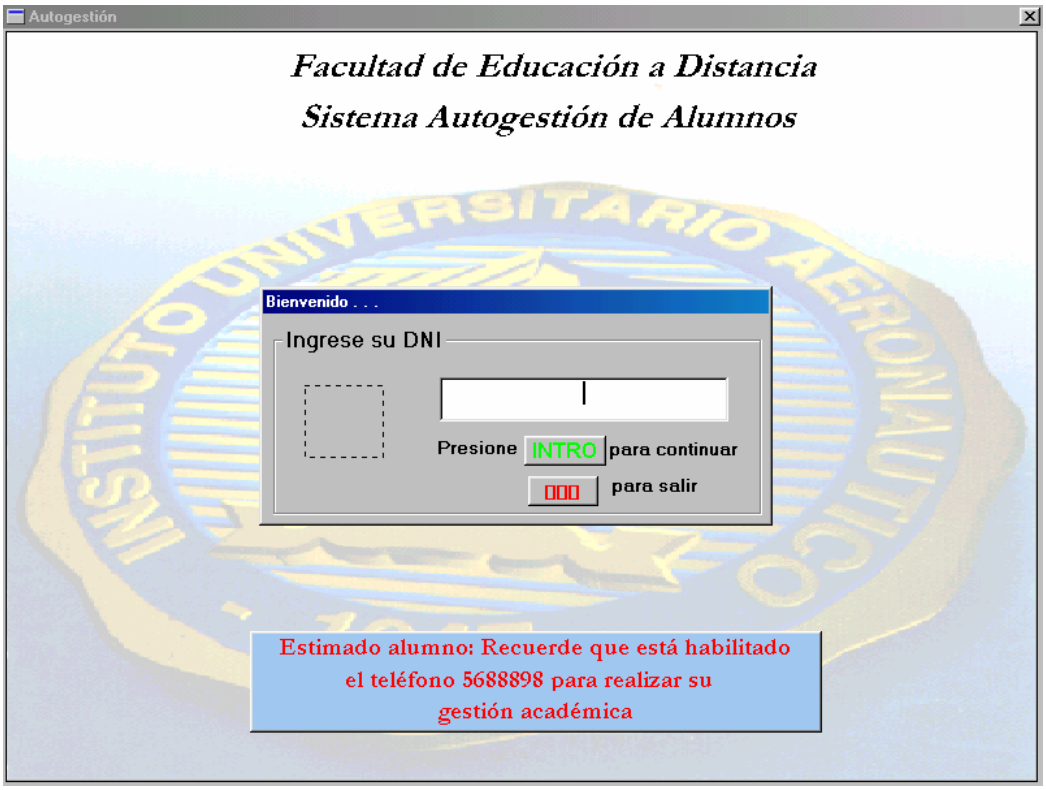


Figura 15 – Menú de Ingreso al Sistema de Autogestión

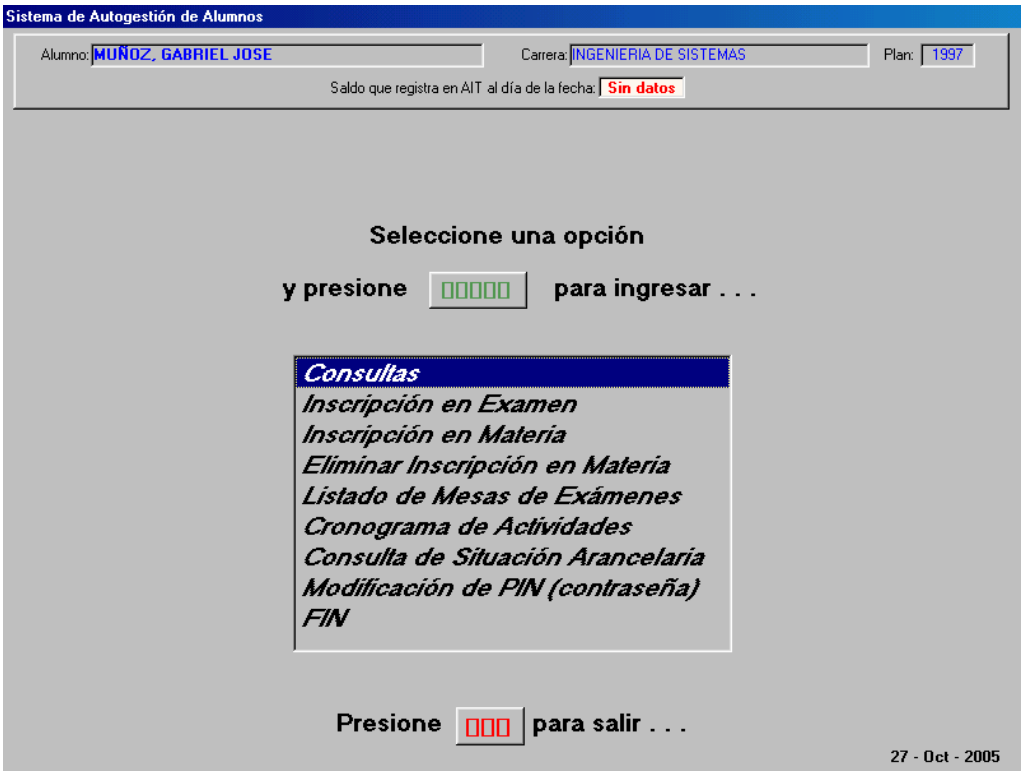


Figura 16 – Menú de Opciones del Sistema de Autogestión

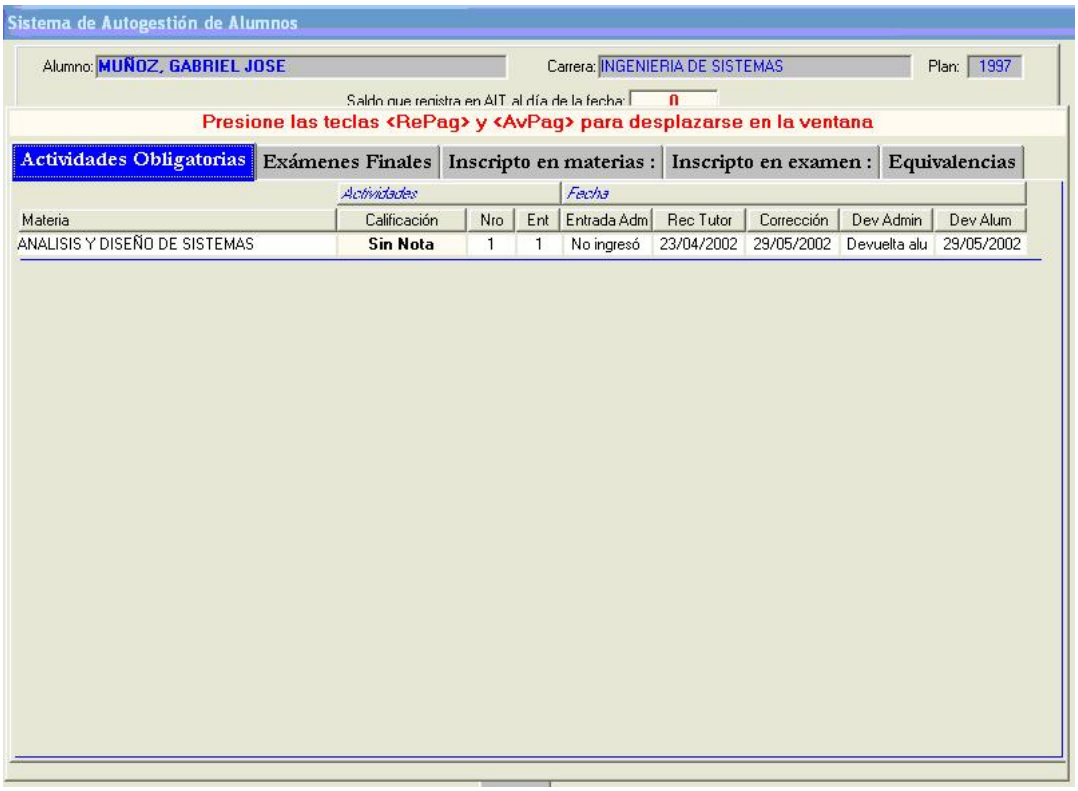


Figura 17 – Menú de Consultas del Sistema de Autogestión

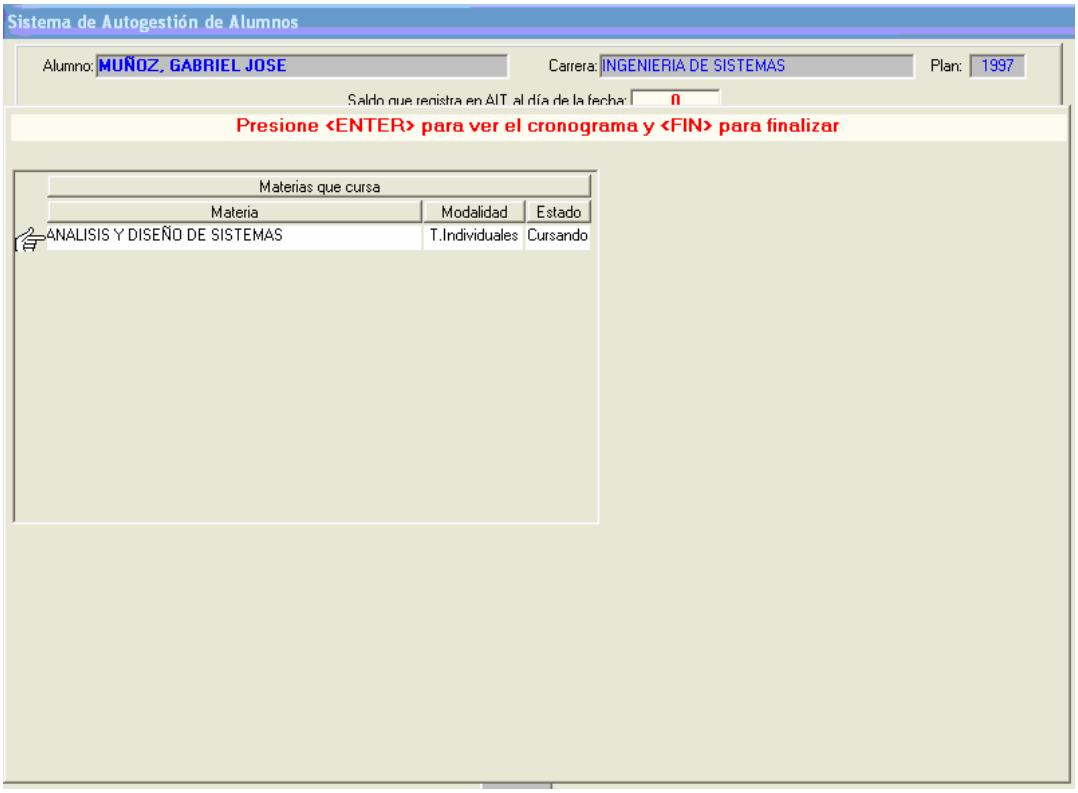


Figura 18 – Consulta de Materias Inscripto

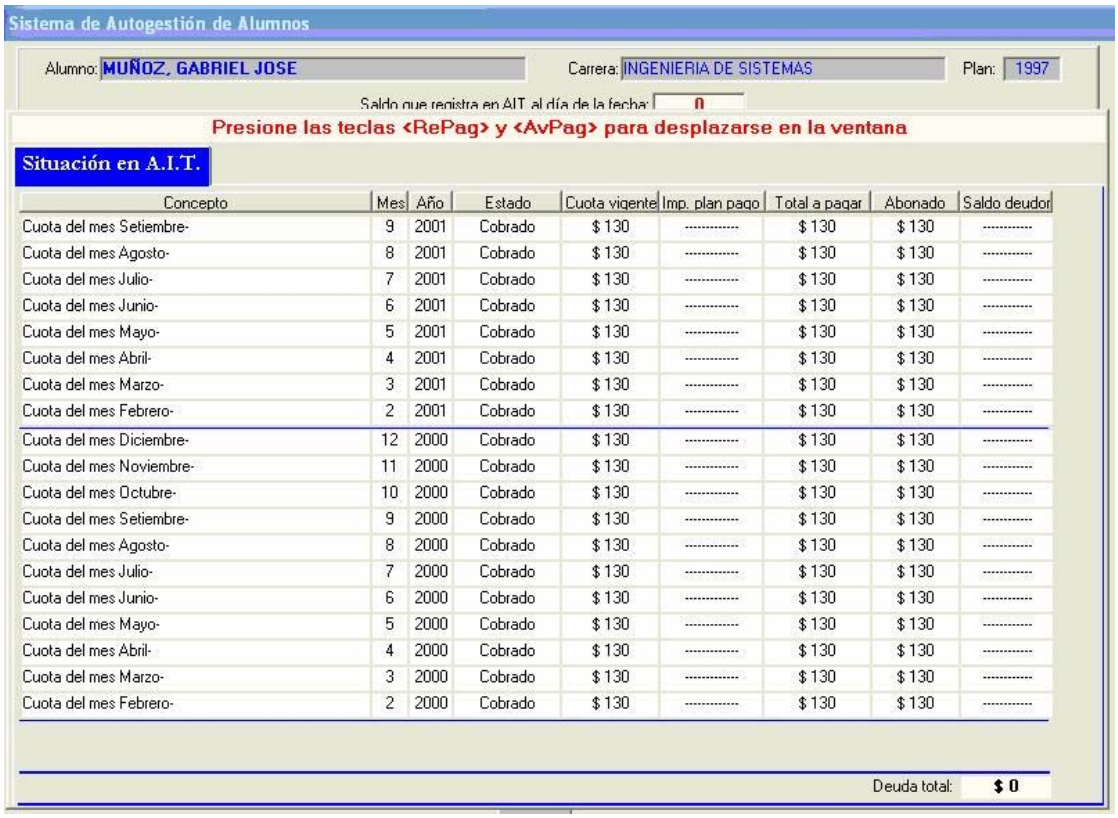


Figura 19 – Consulta de Situación Arancelaria

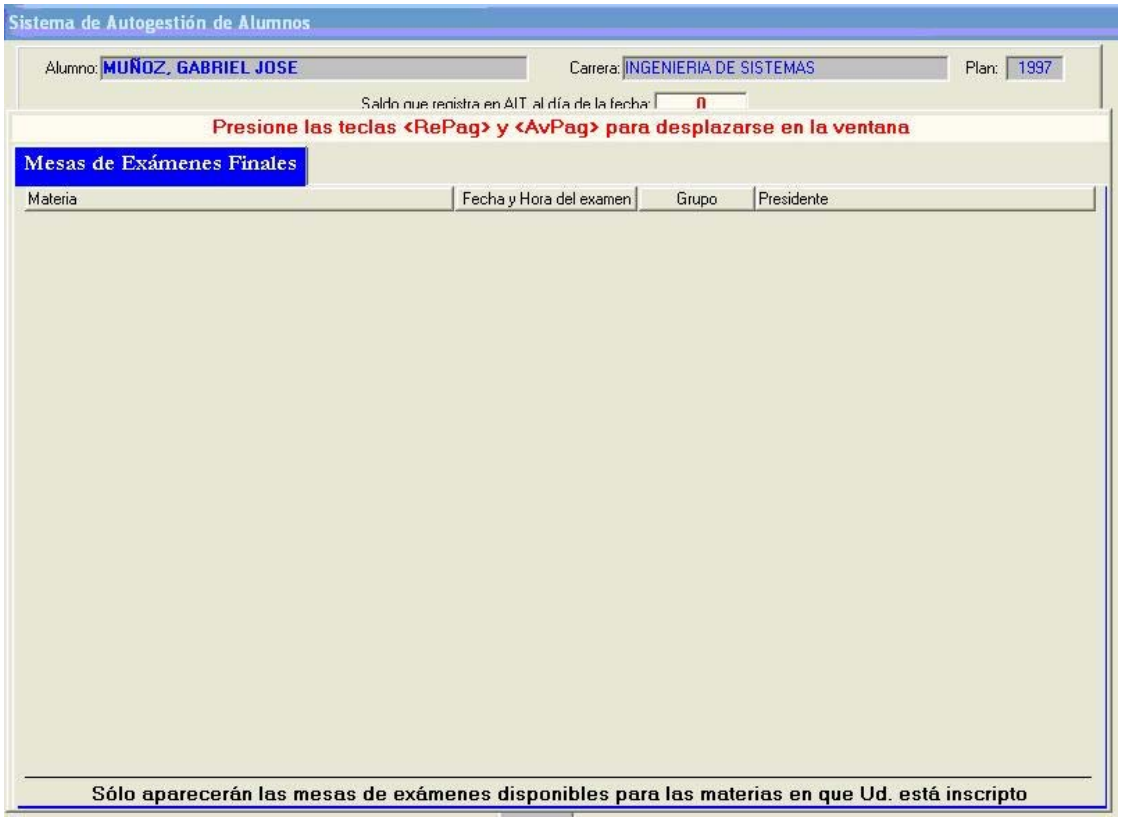


Figura 20– Listado de mesas de exámenes finales

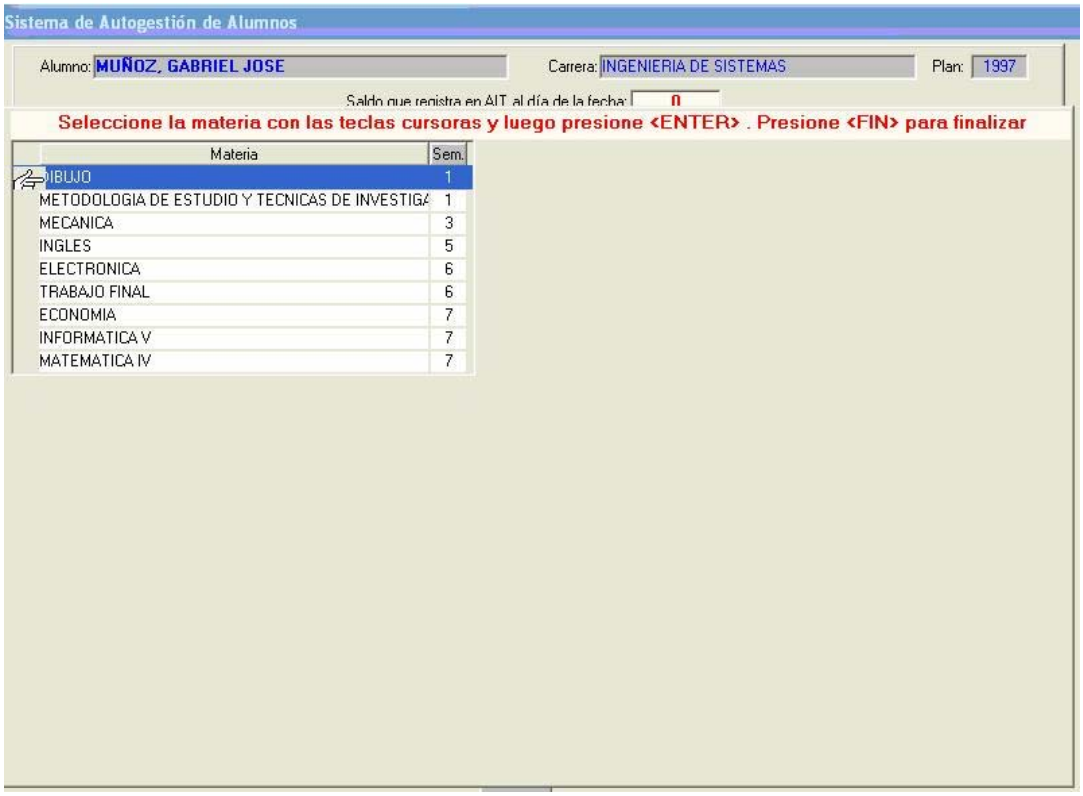


Figura 21 – Inscripción en Materias

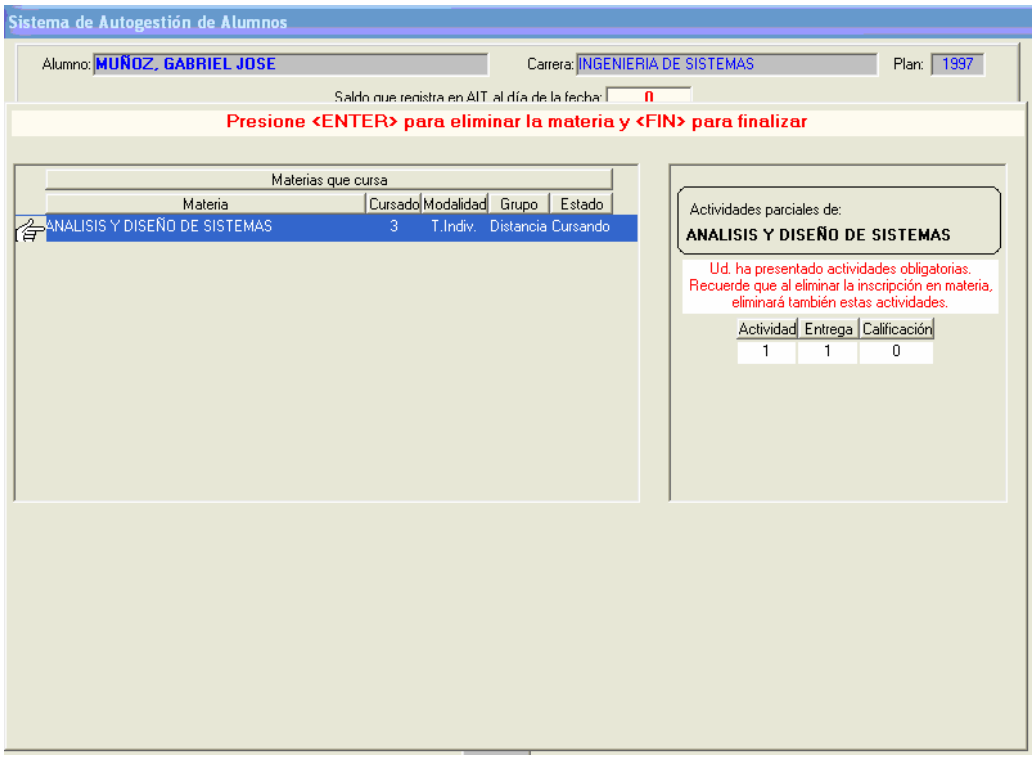


Figura 22 – Eliminar Inscripción en Materias

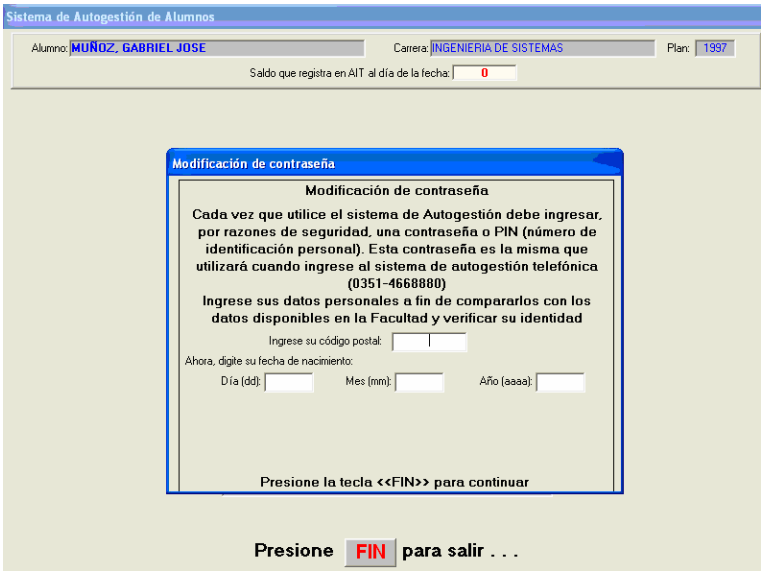


Figura 23 – Modificación de contraseña

b) Construcción del metamodelo del sistema

A los fines de realizar un estudio comparativo entre ambas aplicaciones, se propone la construcción de un meta-modelo empleando el Lenguaje UML.

La entidad central es el Módulo Principal, el cual despliega la información al usuario mediante la carga de forms.

Para acceder a él, previamente se toman los recaudos necesarios de seguridad de acceso, a través de un pequeño form denominado Login. Este Login se autocargará hasta un máximo de tres veces cuando ocurran intentos fallidos de acceso y en caso contrario desplegará el Modulo Principal.

Mientras que el contenido del FormRegistro, que se utiliza para el ingreso de datos, es estático y fijo, el contenido del FormConsulta es dinámico determinado por el puesto de trabajo y puede depender de la información provista por el usuario a través de campos de entrada.

Para modelar estas dos alternativas existen dos subclases derivadas de la clase ModuloPrincipal: FormConsulta y FormRegistro. Cuando el contenido de un form dinámico depende del valor de un conjunto de variables de entrada, estos estarán contenidos en el atributo “use” de las clase Componentes. Por otra parte un form estático sólo estará compuesto de campos que serán completados por el usuario para actualizar la Base de Datos.

La interfaz AccesoDatos existente entre los forms y la BaseDatos, es lo que permite el uso del SGBD. El acceso se produce mediante la clase Componentes, la que provee los drivers nativos de conexión. Las validaciones necesarias las realiza AccesoDatos y es la que permitirá interactuar con la BaseDatos.

En la figura 24 puede observarse gráficamente la implementación de este metamodelo de la aplicación GUI.

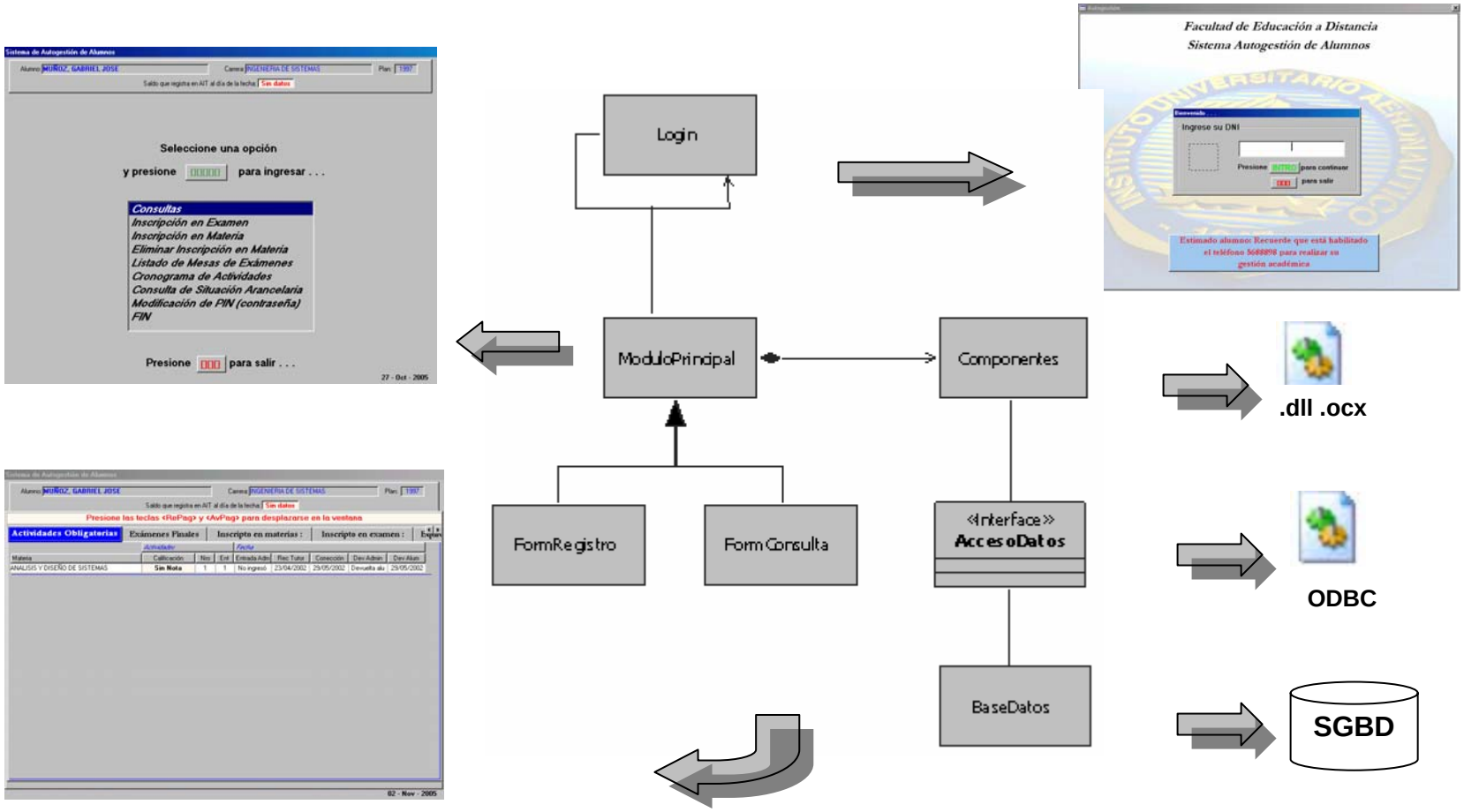


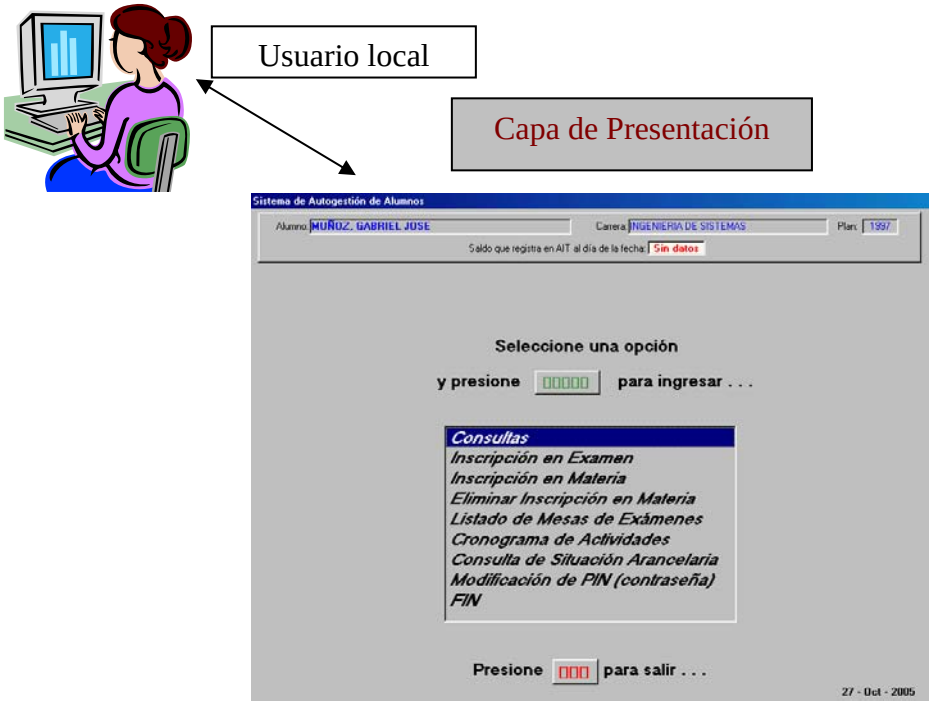
Figura 24 – Metamodelo de la GUI Application

5.2.5 Descripción de la arquitectura física y del software de base

a) Arquitectura física

La arquitectura de esta aplicación se basa en la tecnología cliente/servidor, la cual hace referencia a la conexión de ordenadores por medio de una red a los fines de descentralizar el procesamiento y utilizar fuentes de datos centralizadas. La arquitectura utilizada en este caso de estudio fue de dos capas, la cual se orientaba a la conexión de PC's clientes (alumnos), con servidores conectados a una red, en nuestro caso servidor de aplicaciones y de datos.

En la figura 25 se muestra cómo se implementó en el sistema anterior de autogestión el modelo de dos capas.



Menú principal de sistema de Autogestión

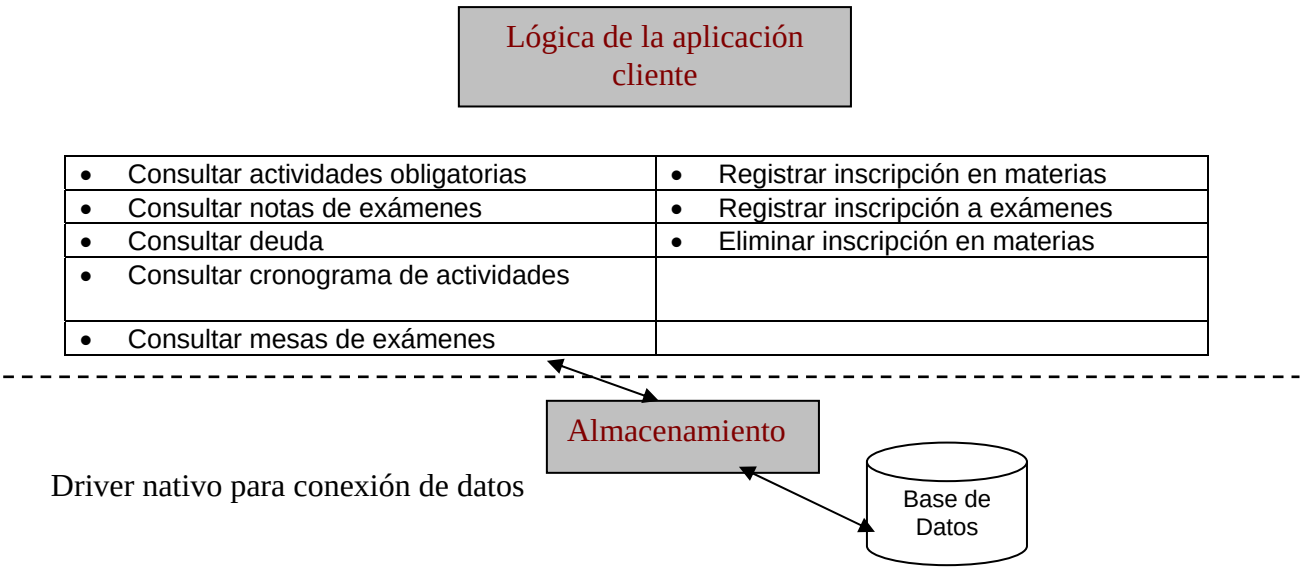


Figura 25– Modelo de dos capas de la aplicación distribuida

Como puede observarse en la figura anterior, la arquitectura de dos capas presenta el inconveniente de que la lógica de presentación y la lógica de negocio se hallan en la máquina cliente (en nuestro caso del alumno), y la lógica de datos en el servidor de base de datos, realizándose el acceso a través de drivers nativos para conexión a datos.

Es por ello que en estos casos debido al grado de procesamiento de la máquina cliente, a éste se lo denomina en nuestro caso Fat Client o cliente pesado, debido a la fuerte dependencia existente entre la aplicación, del lado del cliente y la Base de Datos. En consecuencia, cualquier modificación que se realice sobre ella, hace necesaria la modificación de todas las aplicaciones asociadas, lo que fue crítico en este sistema porque al incorporarse nuevas funcionalidades, la carga de mantenimiento fue creciente.

b) Arquitectura del software de base. Diagrama de componentes

A continuación, se detallan los componentes de este modelo de dos capas:

b.1 Cliente Windows

- El componente EXE UIAutogestion representa al programa que se ejecuta en la máquina cliente, es decir, que éste se encuentra instalado en la PC cliente y es el front-end de la aplicación.
- En el componente ActiveX LogicaUIAutogestion se encuentra la organización de los programas para la gestión.
- En tanto, en la capa componente ActiveX ComponentesLogicos se realiza la gestión de transacciones contra la base de datos. Ejemplo: cadenas de conexión y permisos.
- El componente ActiveX Entidades personaliza la gestión de transacciones por cada operación, es decir, separa por tareas las operaciones. Ejemplo: inscripción en materias, consulta de calificaciones.
- Por último, el componente ADO AccesoDatos utiliza drivers de conexión entre la aplicación y la base de datos. Ejemplo: drivers nativos del lenguaje de programación.

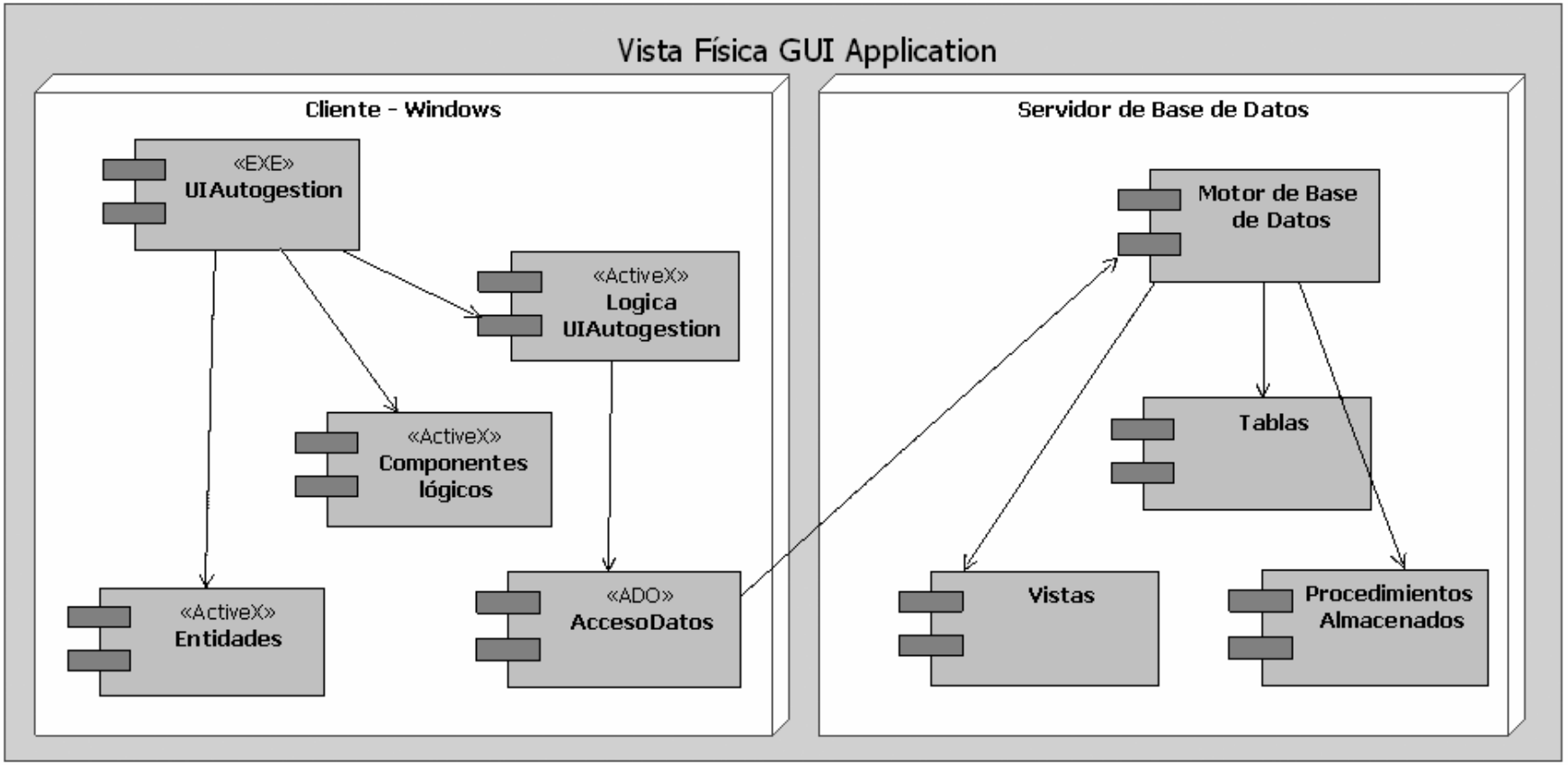
b.2 Servidor de base de datos

Del lado del servidor de base de datos se encuentran:

- El MotorBaseDatos es el servicio otorgado por SGBD y que, en este caso, es SQL Server. En el componente Tablas se encuentran almacenados los datos. En el componente Vistas se encuentran sentencias de consultas habituales a la BaseDatos y en el componente ProcedimientosAlmacenados se encuentran los script transaccionales principales para gestionar los datos.

La concurrencia es dividida en dos, por un lado cada ActiveX Entidades debe ser lo suficientemente corto y preciso en el envío y recepción de solicitudes de BaseDatos de manera que permita liberar rápidamente los registros en cuestión. Por otro lado, los componentes Vistas y ProcedimientosAlmacenados deben marcar el comienzo de una transacción mediante un Begin Transaction de manera que aseguren la consistencia de la información. Si la operación es exitosa se termina con un Commit Transaction actualizando, recién en este momento, los datos físicos de la Base, en caso contrario, se aplica un Rollback, la operación se deshace y vuelve al punto de partida. Por último el componente MotorBaseDatos manejará un determinado número de conexiones y solicitudes al mismo tiempo administrado por DBA (Data Base Administrador).

La figura 26 ilustra la distribución del procesamiento entre los distintos componentes o nodos distribuidos entre las estaciones de cliente (bajo Windows), y el servidor central.



Los componentes físicos se implementan agrupándolos en subsistemas organizados en capas y jerarquías.

Figura 26 – Diagrama de componentes de la Aplicación GUI

5.2.6 Análisis de las limitaciones del modelo anterior implementado

a) Adaptado a una sola facultad: sólo fue diseñado para la modalidad de educación a distancia

b) Operatividad: cualquier cambio efectuado en el sistema implicaba la reconfiguración del mismo en cada PC. Las terminales asignadas eran escasas en número por lo que los alumnos debían realizar largas esperas con el objetivo de consultar o inscribirse.

c) Mantenimiento: el mantenimiento de las PC en funcionamiento y la impresora asignada era permanente y exigía un control continuo por personal de soporte técnico.

d) Servicios: los servicios prestados a los alumnos se limitaban a la gestión académica prioritaria, es decir consultas básicas e inscripciones, razón por la cual el resto de los servicios debía realizarse en forma personal en el departamento de alumnos o en las distintas dependencias, según el trámite a realizar por el alumno.

e) Interfaz: la interfaz es primitiva, el espacio de diseño de la interfaz es de dos dimensiones, con menús estrictamente jerárquicos, con opciones limitadas que sólo pueden seleccionarse a través del teclado numérico.

5.3 Aplicación de la metodología para el análisis del sistema migrado

El acceso al sistema actual de autogestión se encuentra inserto dentro del sitio Web del Instituto Universitario Aeronáutico (www.iua.edu.ar) como una forma de integrar la vida académica, investigativa y administrativa de la Universidad, enfocando la gestión de alumnos fuera de sus fronteras físicas, modalidad que permite la red de redes Internet.

La descripción del sistema sirve como punto de partida para comprender los requisitos del mismo.

5.3.1 Adecuación de la especificación de requerimientos

a) Estudio preliminar

a.1 Servicios prestados por el sistema

- Realiza el seguimiento y administración de los alumnos, desde que ingresan al curso de ingreso o nivelación, en ambas facultades del Instituto Universitario Aeronáutico, a diferencia del sistema anterior que estaba habilitado **sólo para los alumnos de la Facultad de Educación a Distancia.**
- Brinda al alumno nuevos servicios, a diferencia del sistema anterior que sólo lo habilitaba para consultas sobre su actividad académica e inscripción en materias y exámenes.
- El alumno, a través de su clave, puede hacer uso de muchos otros servicios que se detallan a continuación y a los que se accede a través del sitio Web dentro del subsistema de gestión de alumnos, al cual ingresa a través de su clave.

a.2 Relación con otros sistemas

- Biblioteca
- Librería
- Aranceles

a.3 Alcance del SI

El sistema comprende los procedimientos relativos a la gestión de los alumnos:

- Uso de correo electrónico a través de cuenta asignada por la Universidad.
- Uso de aulas virtuales.
- Control de datos personales.
- Matriculación en curso de admisión.
- Matriculación en carrera.
- Inscripción en materias y obtención de comprobantes direccionados por el sistema al correo electrónico del alumno.
- Reinscripción anual a través de formulario.
- Inscripción en exámenes parciales y finales con obtención de comprobantes.
- Inscripción en materias con obtención de comprobantes
- Envío de actividades parciales obligatorias.
- Emisión de constancias de alumno regular, examen parcial/final.
- Mantenimiento de carrera:
  - Suspensión y baja.
  - Alta y reinscripción.
  - Cambio de carrera y modalidad de carrera.
  - Consultas en general:
  - Notas parciales/finales.
  - De Planes de estudio y materias.
  - Del Reglamento del alumno y resoluciones decanales y rectorales.
  - Estado de actividades obligatorias.
  - Material de estudio.
  - Fechas de exámenes finales.
  - Horarios de tutorías.
  - Correlatividades.
  - Habilitación para rendir exámenes.
  - Materias a cursar.
  - Consulta de situación financiera.
  - Consulta de actividades recreativas y culturales.
- Comunicación con los departamentos académicos, de alumnos y docentes.

a.4 Características tecnológicas

a.4.1 Programación

La herramienta usada para la programación es PHP, ASP

a.4.2 Base de Datos

Microsoft SQL Server

b) Reutilización de requisitos en el proceso de migración a la Web

Tal como se plantea en el capítulo anterior, la reutilización de requisitos es un enfoque importante en el proceso de migración, ya que no sólo se aprovecha el conocimiento del sistema anterior, sino que además permite identificar los requisitos nuevos y aquellos sujetos a cambios en el nuevo sistema.

Como los requisitos representan el conocimiento de un dominio particular, y éste se refiere a un área funcional diferenciable dentro de un contexto dado, en el caso de estudio, ese contexto es la Universidad, y el dominio es el Subsistema de Autogestión de Alumnos, sobre el cual se aplica este enfoque.

Debido a la necesidad del conocimiento de este dominio para aplicar el enfoque de migración al nuevo sistema, se propone dividir el dominio en dos subdominios, ambos basados en los requisitos de ambos sistemas (anterior y actual). La comparación de estos subdominios se realiza mediante analogía basada en escenarios y casos de uso.

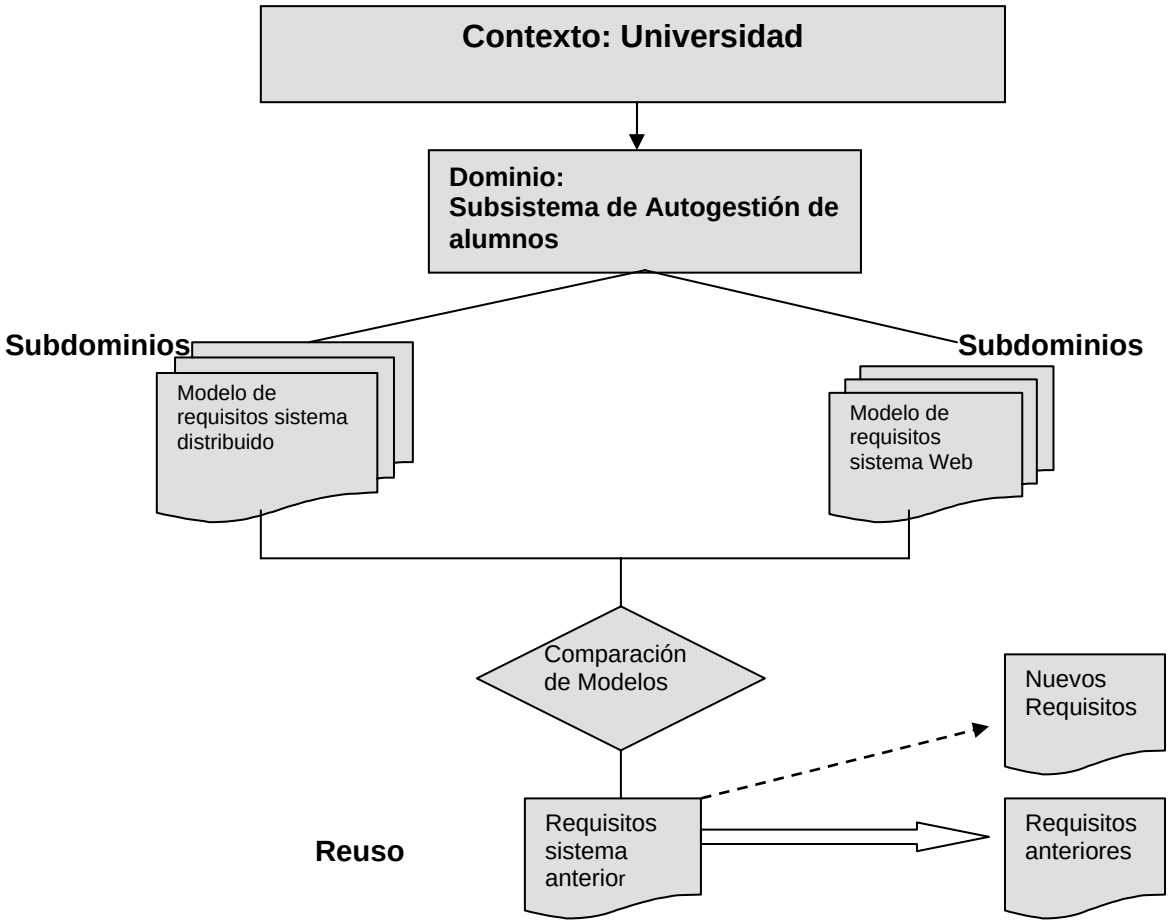


Figura 27 – Modelo de requisitos basado en dominios

Con el objetivo de abordar nuestro caso de estudio basado esencialmente en la utilización de modelos aplicando UML, se observa que este enfoque se acerca a la Metodología UWE, enunciada por Koch. Del estudio realizado es posible rescatar las siguientes características:

- UWE es una propuesta basada en el proceso unificado y UML, pero adaptados a la Web.
- En requisitos, separa las fases de captura, definición y validación.

Hace además una clasificación y un tratamiento especial dependiendo del carácter de cada requisito. En el Anexo II se presenta un estudio comparativo del tratamiento de requisitos para la Web.

c) Previsiones para superar las limitaciones comprobadas en el sistema original

Las "nuevas reglas de gestión" surgen del reglamento del alumno que figura en la página Web de la institución. A continuación se sintetizan las más relevantes:

- Para obtener la condición de alumno de la carrera o curso, deberá estar inscripto o reinscripto en la carrera o curso que se dicte en la facultad correspondiente y estar habilitado, condición que se mantiene mientras no se registre un atraso mayor a una cuota del régimen arancelario vigente.
- Los alumnos deberán concretar anualmente la reinscripción en la facultad que corresponda, abonando la matrícula respectiva. Es requisito para realizar el trámite de reinscripción en la carrera mantener la condición de "habilitado".

Para mantener la condición de alumno, se deberá:

- Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del año académico en la modalidad presencial o del año académico personal, en la modalidad a distancia.
- No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera.
- Efectuar la reinscripción anual.
- Es obligatorio inscribirse en la/s asignatura/s a cursar. Los requisitos para dicha inscripción son:
    - Estar inscripto o reinscripto en la carrera, según el caso.
    - Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes.
    - Estar en condición de habilitado.
- El alumno accederá a la condición de regular en una asignatura, aprobando las actividades obligatorias previstas en la misma que establezca cada facultad.
- La condición de regular en la asignatura habilita el acceso al examen final de la misma.
- En el caso de que se produzca la pérdida de la condición de alumno por cualquiera de las causas mencionadas anteriormente, se invalidará para el causante cualquier registro de calificaciones de actividades obligatorias pertenecientes a las asignaturas en las que no haya alcanzado la regularidad correspondiente.
- El alumno perderá la condición de regular en la asignatura, cualquiera fuere la modalidad de cursado, cuando se cumpla alguna de las siguientes condiciones:
    - Haberse vencido el plazo desde que adquirió su condición de alumno, establecido por cada facultad.
    - Haber sido aplazado tres (3) veces en el examen final de la misma asignatura, aún cuando quedasen turnos pendientes.

- En caso de pérdida de la condición de regular, el alumno deberá recursar la asignatura y alcanzar la regularidad según lo establecido.
- Es requisito indispensable para presentarse a un examen de una asignatura ser alumno regular de la misma, estar habilitado e inscribirse en término.
- Los requisitos para inscribirse en un examen final son:
  - Tener adquirida la condición de alumno regular en la asignatura.
  - Tener aprobadas las asignaturas correlativas correspondientes de conformidad con su plan de estudio.
  - Estar habilitado.
- Se considera que un alumno egresa, con un nivel de pregrado o grado, según corresponda, cuando ha aprobado satisfactoriamente todos los requisitos del plan de estudios correspondiente.

Del análisis de las reglas de gestión del nuevo sistema puede determinarse la existencia de estados bien definidos en la condición del alumno, como se observa en la figura 28.

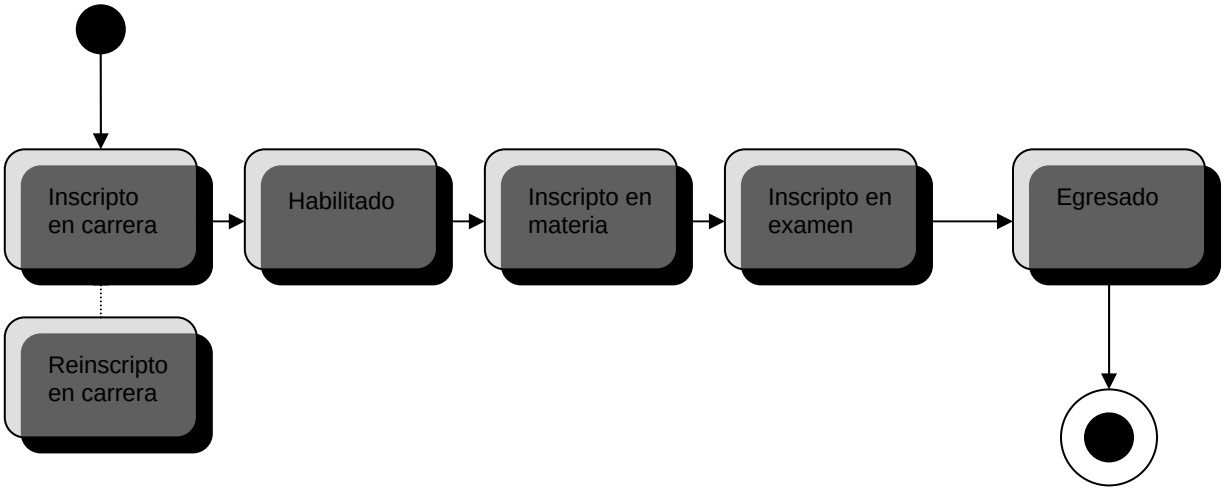


Figura 28 - Diagrama de estados de la condición del alumno en el Sistema de Autogestión

5.3.2 Adecuación de la descripción funcional

a) Revisión del modelo funcional

El esquema funcional del sistema anterior se amplía con el agregado de funciones correspondientes a la interacción de un alumno con el portal de la universidad.

El mismo le permite, no sólo realizar las funciones comunes referentes a su condición de alumno sino también realizar trámites varios, descargar software, realizar encuestas en línea, obtener sus comprobantes de inscripción y de pago en forma virtual, además de ser partícipe, a través de las Noticias, de cursos, novedades y becas que ofrece el instituto.

Por otro lado, a través de la página e identificándose como alumno puede acceder a las aulas virtuales de las materias en las cuales está inscripto y acceder al Plan de Estudios de las carreras que se cursan.

Las opciones de consulta le permiten a su vez poder conocer el estado de su currícula, la correlatividad de las materias y exámenes y los trámites que ha realizado.

b) Realización del modelo de casos de uso de la aplicación Web

En la página siguiente se presenta el diagrama de caso de uso general de la aplicación migrada a la Web. Puede observarse en la figura 29 la adición de nuevas funcionalidades en la aplicación Web; asimismo, que existen funciones principales que han sido obtenidas en el nuevo sistema a partir de la aplicación tradicional, las que se demarcan con una tonalidad más intensa.

5.3.3 Adecuación de la descripción de la dimensión estática

a) Revisión del modelo conceptual

El modelo conceptual construido de la aplicación anterior se revé y se modifica para que se reflejen las nuevas entidades que lo componen en función de la redefinición de las funcionalidades y del agregado de nuevas entidades que responden a las mismas, con el objeto de adaptar el mismo a las nuevas reglas de negocio relevadas y a las exigencias de la nueva tecnología a implementar.

Este nuevo modelo conceptual puede observarse en la figura 30.

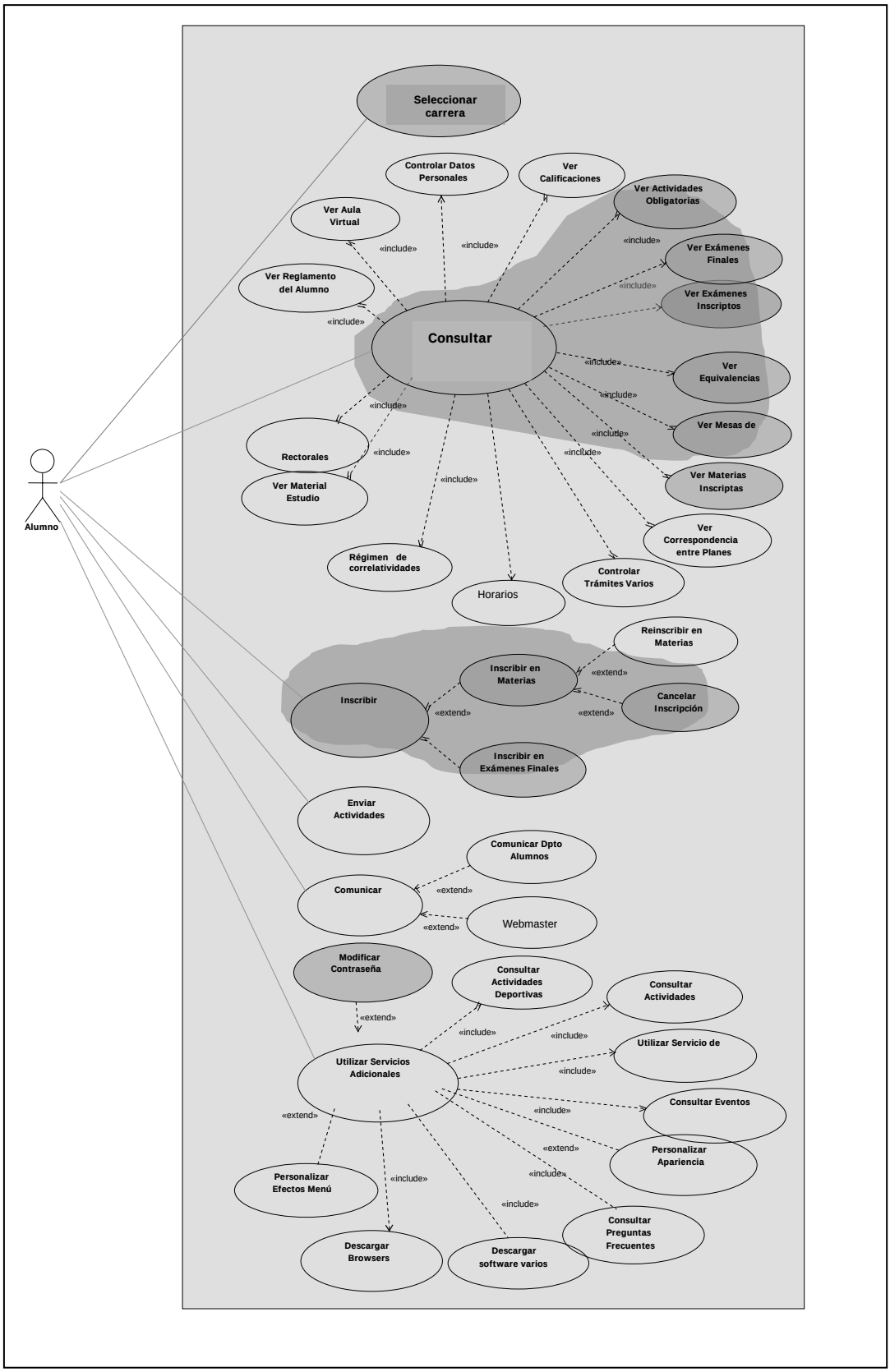


Figura 29– Diagrama de casos de uso de la aplicación Web

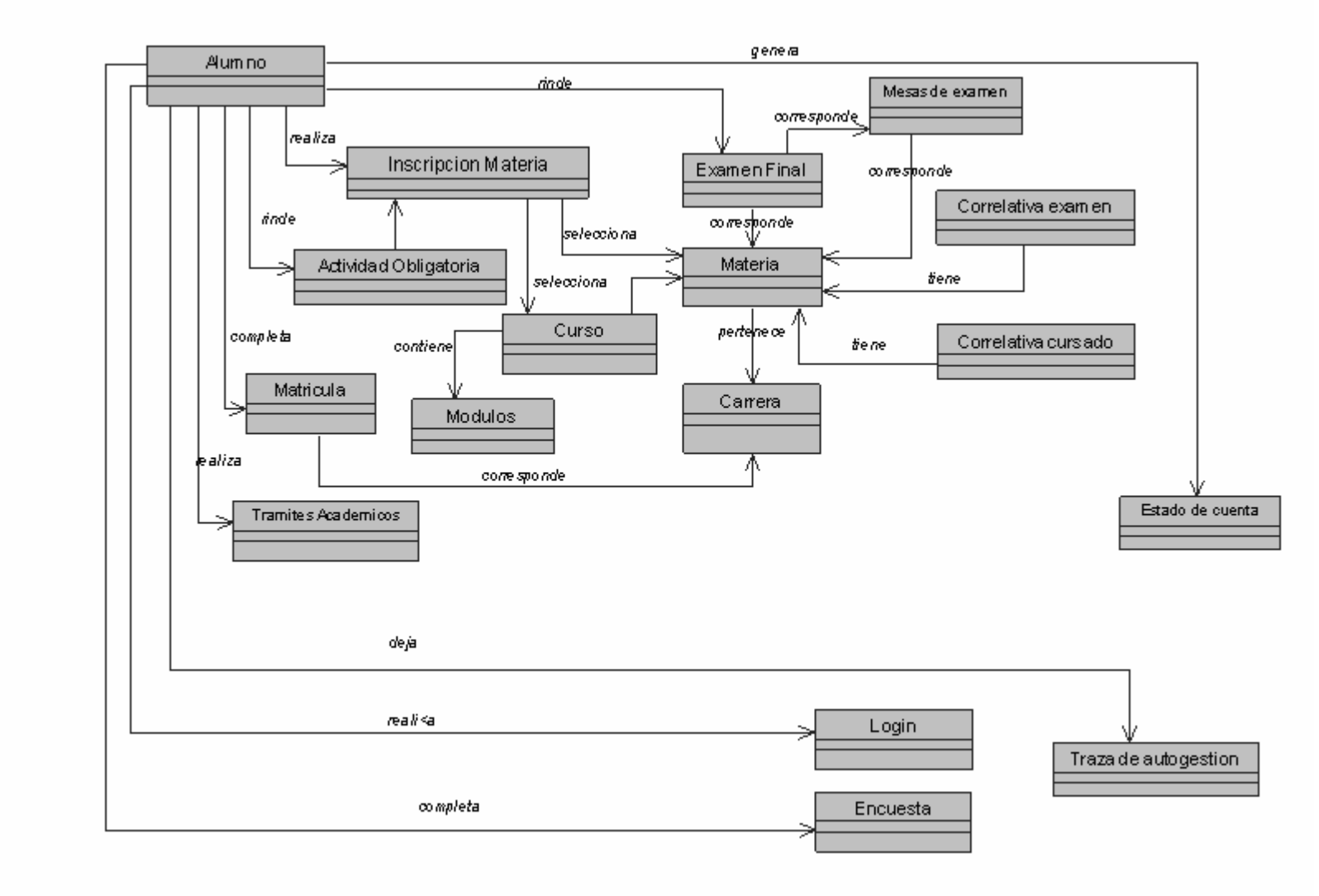


Figura 30 – Modelo Conceptual aplicación Web

5.3.4 Descripción de la interfaz de usuario en la aplicación web

a) Análisis del modelo de interfaz de usuario de la aplicación Web

A continuación se muestran las pantallas diseñadas para la interfaz con el usuario en la aplicación Web. Como podemos observar en las figuras 31 a 39, las opciones se encuentran divididas de acuerdo a la funcionalidad prevista. Así se presentan las opciones de consulta de alumnos (datos personales, calificaciones, trámites, etc.) e institucional (horarios de materias, exámenes, correlatividades, etc.). En la opción Inscripción puede observarse en Materia, Examen o preinscripción.

A través del menú el alumno puede Enviar actividades obligatorias, Cancelar su inscripción en materias, disponer de Servicios como publicar avisos, configurar y descargar programas, etc. Puede seleccionar Comunicarse para hacerlo con los departamentos o el Webmaster, es decir toda la funcionalidad prevista y organizada.

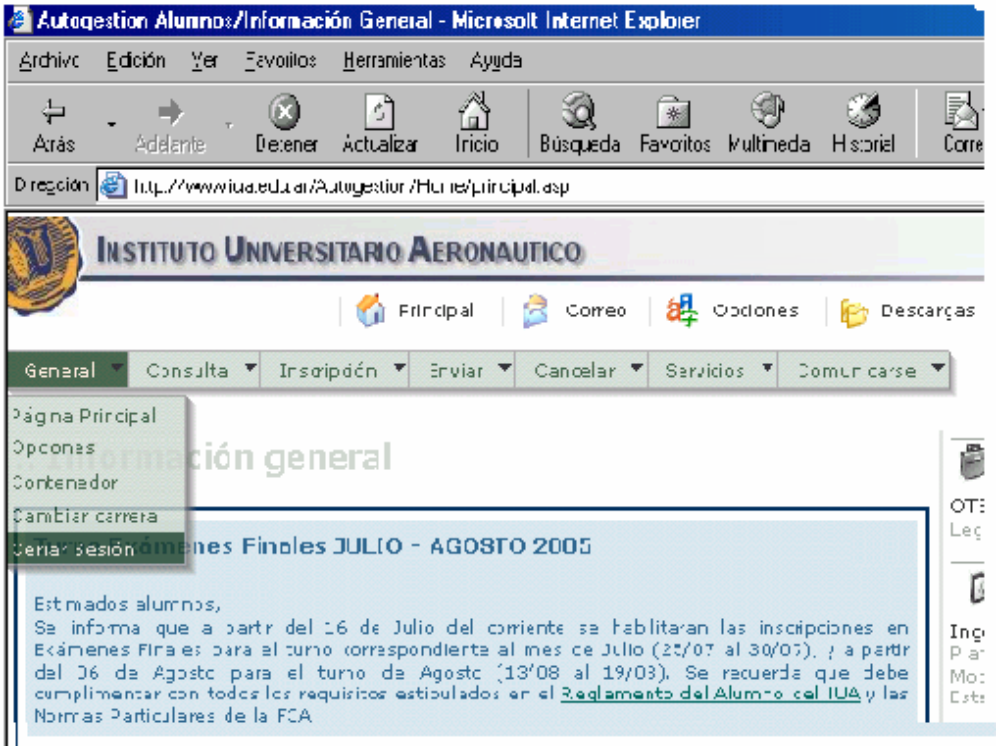


Figura 31 – Pantalla principal del sistema de autogestión

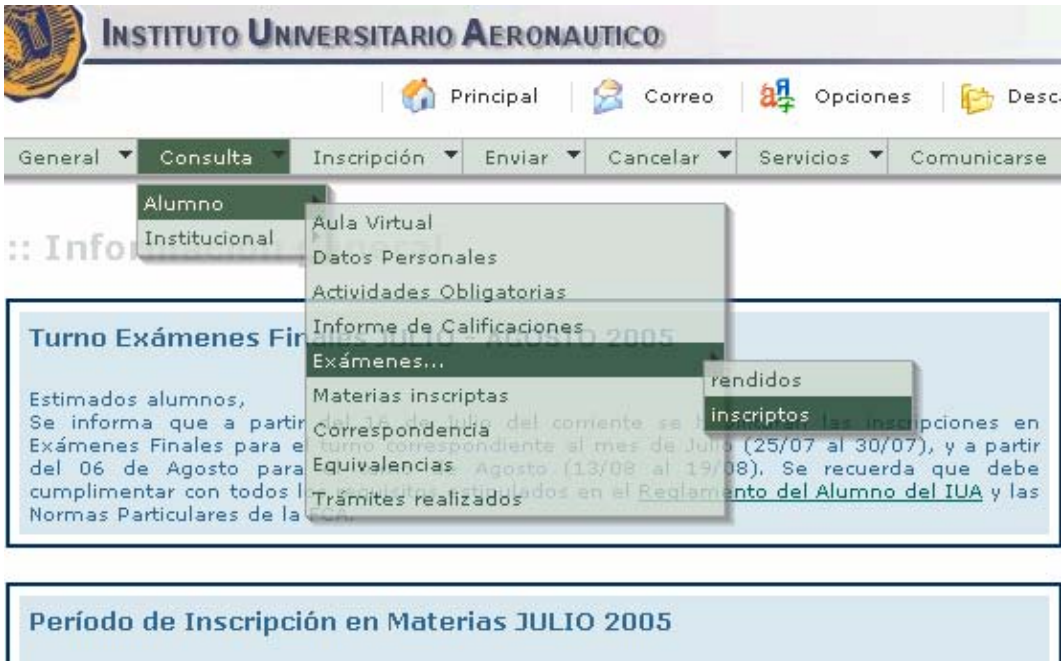


Figura 32 – Opciones de la Consulta de alumnos

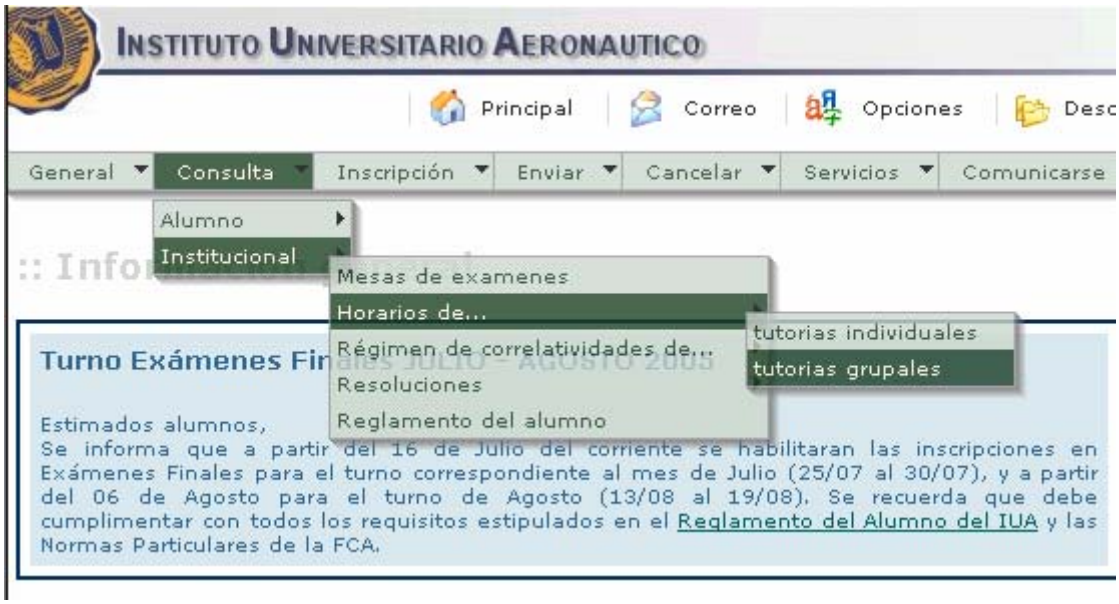


Figura 33 – Opciones de la Consulta institucional

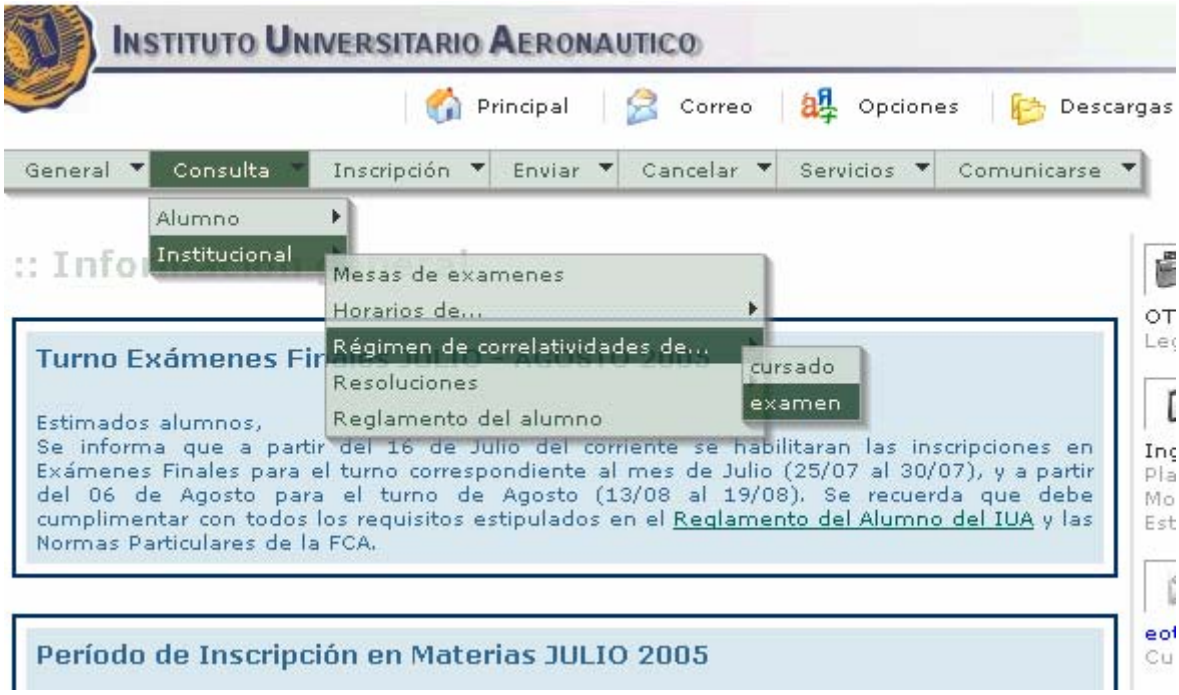


Figura 34 – Opciones de la Consulta institucional

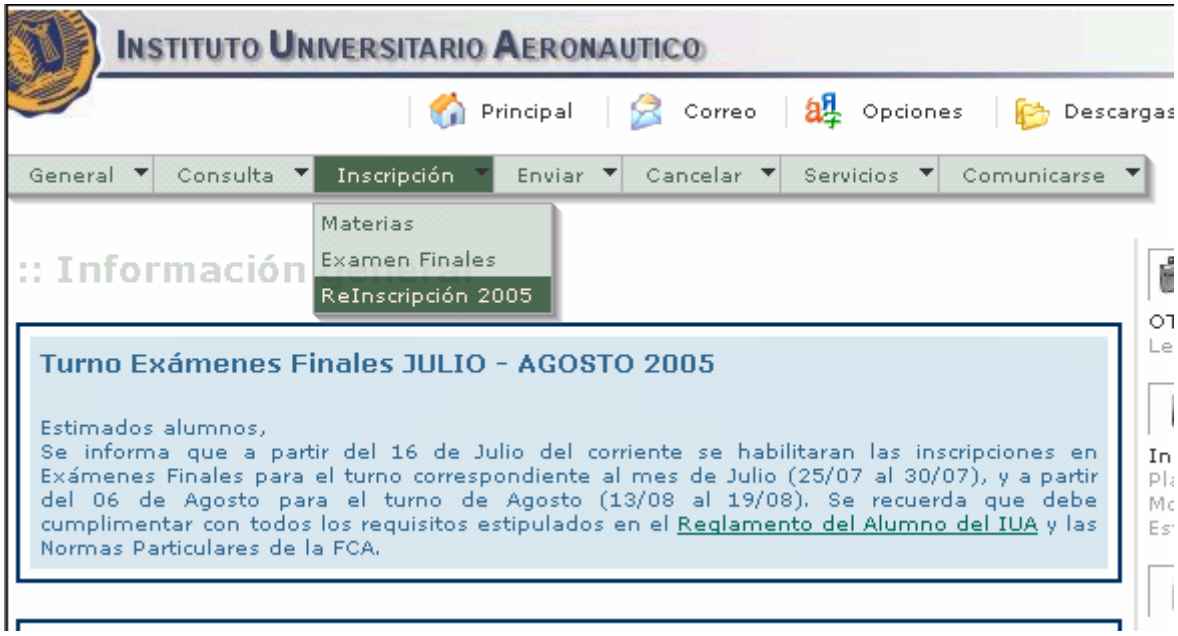


Figura 35 – Opciones de Inscripción

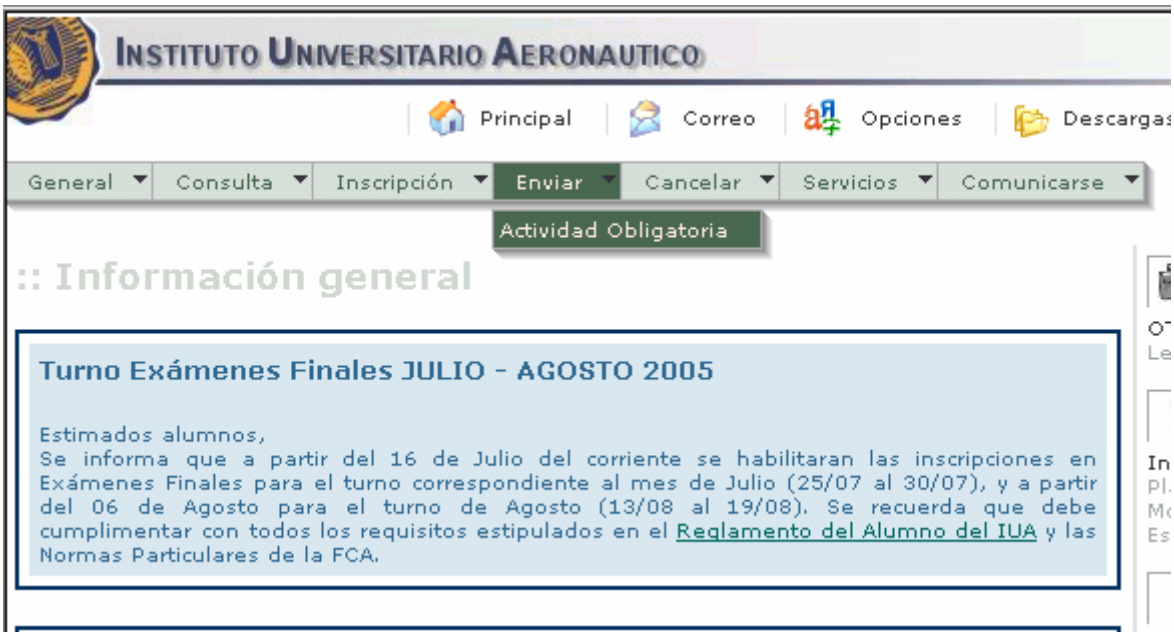


Figura 36 – Envío de actividades obligatorias

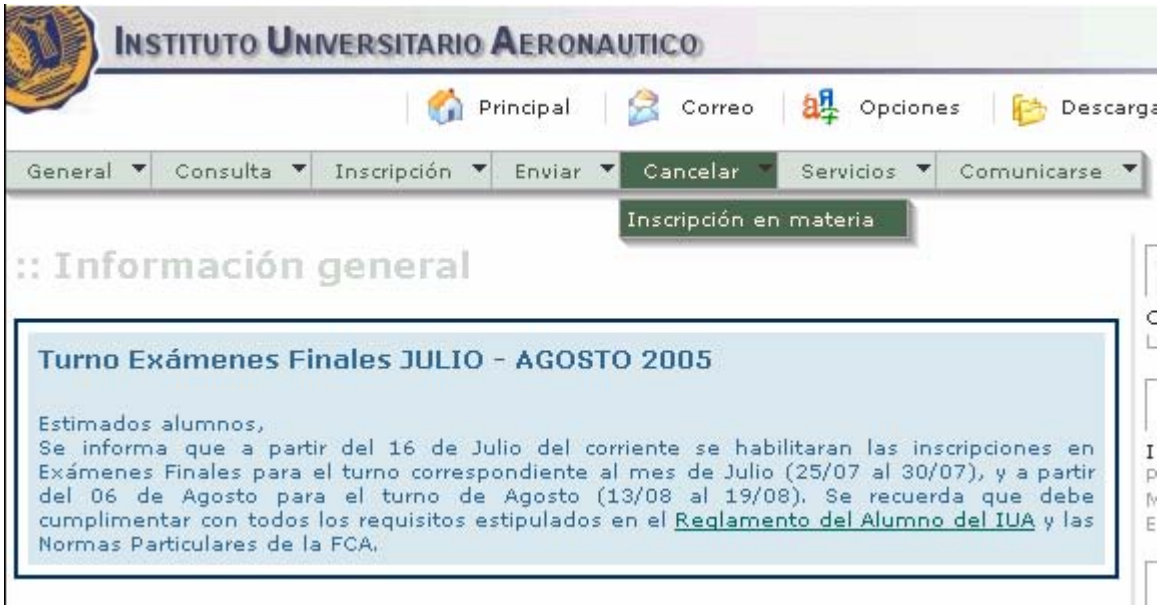


Figura 37– Cancelar Inscripción en materia

Figura 38 – Servicios ofrecidos

Figura 39– Comunicación Institucional

Si bien se presentan las pantallas principales del sistema, su diseño arquitectónico y los contenidos funcionales hacen que la navegación sea sencilla orientada básicamente a las necesidades de los alumnos. Su estructura es en red, lo que permite seleccionar cualquier opción sin necesidad de seguir una estructura jerárquica establecida.

En el diseño se respetaron los colores institucionales y las páginas siguen los estándares establecidos por la W3C (Consorcio World Wide Web) que es un consorcio internacional donde las organizaciones miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web..

En la interacción se brindan opciones de cambio de la configuración del browser en la máquina local y, la posibilidad de efectuar descarga del software necesario para operar en el sitio web.

b) Construcción del metamodelo de la aplicación web

Para describir la aplicación Web genérica se utiliza un meta-modelo, que se presenta en la figura 40, en el cual la entidad central es el *Browser*. Los navegadores Web utilizan protocolos de comunicación tales como el *http*, *https* y *ftp* que interactúan con *WebServer* mediante servicios del tipo Apache o IIS (Internet Information Server), para la gestión de información, como es nuestro caso; a la vez que autentican servicios y gestionan cookies. En el modelo a desarrollar, este servicio tomará contacto con un ApplicationServer que contará con una interfaz de *AccesoDatos* para gestionar la información con la *BaseDatos.*

Los *frames* y otros *Componentes* facilitan la organización y la interacción. La navegación de una página a otra es modelada por la asociación a sí misma de la clase *Paginas*. El acceso a las páginas se realiza mediante autenticación a través de la clase *Login*, que tendrá la misión de permitir la navegación en el sitio.

Mientras que el contenido de una página Web estática es fijo, el contenido de una página dinámica es determinado en tiempo de ejecución por el Server y puede depender de la información provista por el usuario a través de campos de entrada. Para modelar estas dos alternativas existen elementos contenidos en la clase *Componentes*. Esta clase contiene Scripts, XML para la transferencia de datos, componentes ActiveX, Applets, Flash Movies, JavaBeans, etc. El contenido de una página dinámica depende del valor de un conjunto de variables de entrada provistas por el usuario.

La organización en frames es representada por la asociación split into, cuyo destino es un conjunto de entidades frames. La subdivisión en frames puede ser recursiva y cada frame tiene una asociación unaria con la página Web inicialmente cargada dentro del frame (ausente en el caso de subdivisión recursiva dentro de los frames). Cuando un link en una página Web fuerza la carga de otra página dentro de un frame diferente, el frame de destino se convierte en el miembro de datos de la clase opcional de asociación Load Page Into Frame.

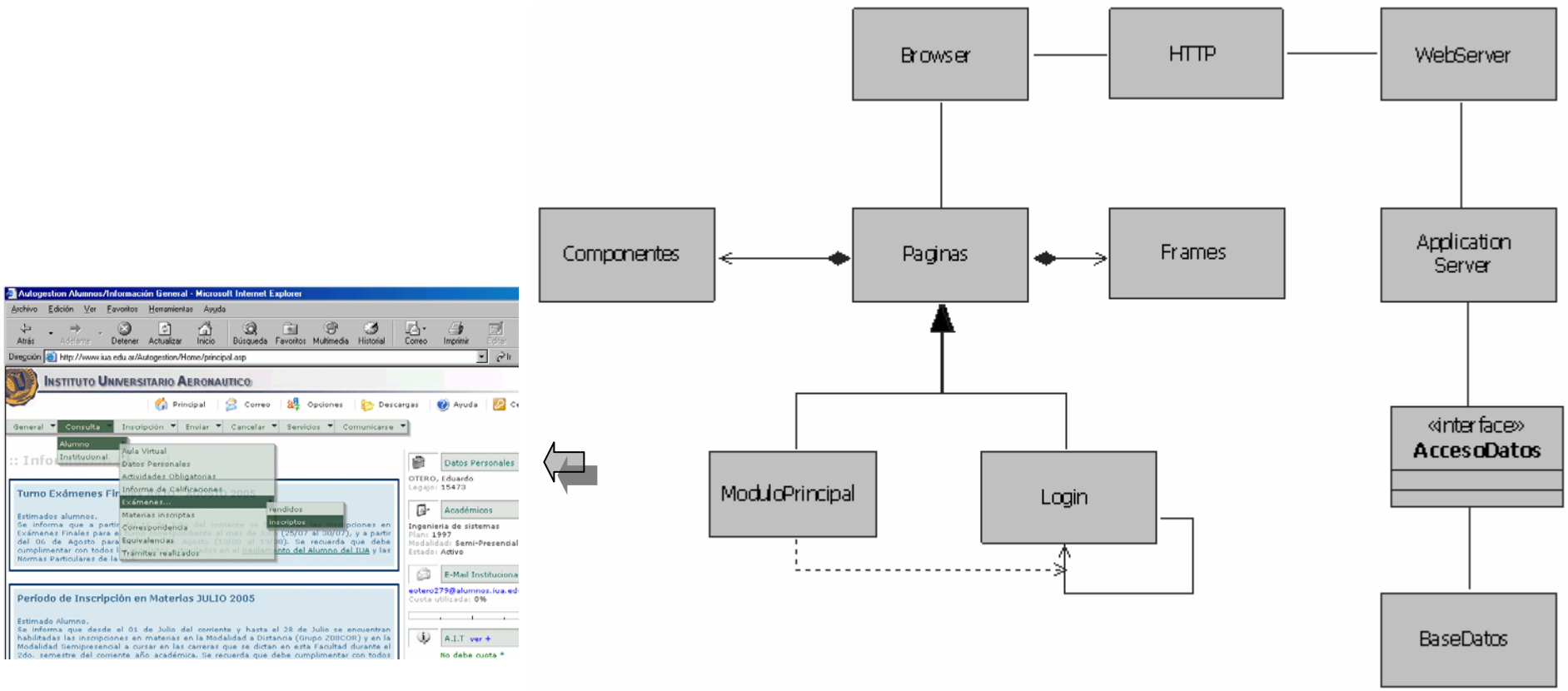


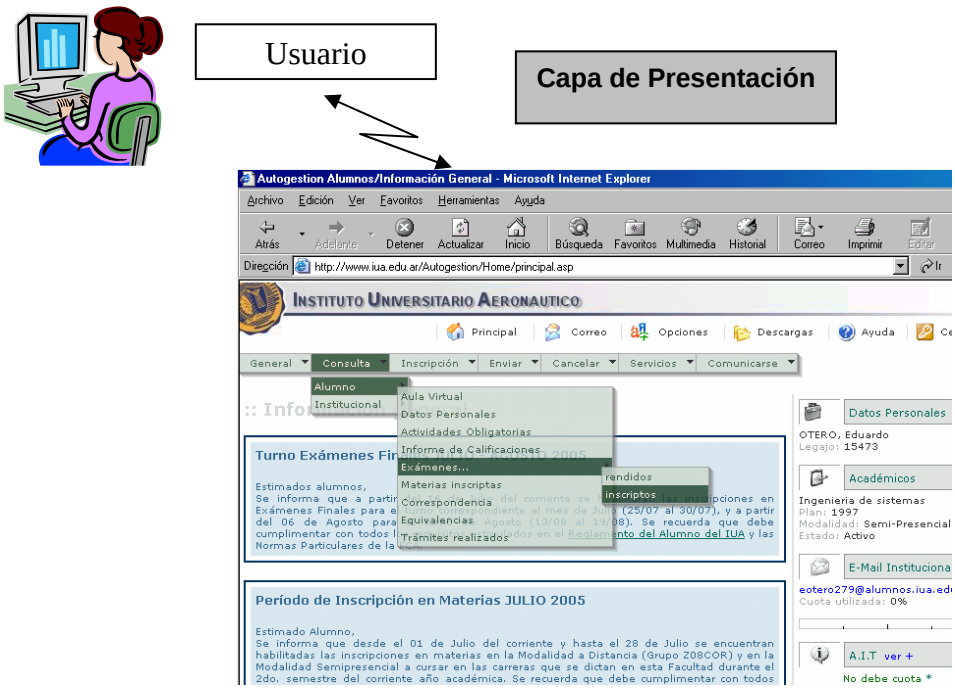
Figura 40 – Metamodelo de la aplicación web

5.3.5 Revisión de la arquitectura física y del software de base

a) Arquitectura física

La disciplina de diseño de interfaces experimentó un gran impulso con el desarrollo de aplicaciones Web para uso masivo por grupos de usuarios de ámbito universal y bajo fuertes restricciones de velocidad debido al ancho de banda existente. En esta arquitectura a la cual se migró, una máquina cliente realiza peticiones a una máquina servidora y ésta a su vez a otros servidores para satisfacer la petición original. Como se observa en la figura 41, el nivel lógico es independiente de la capa física y de la presentación (browser), pudiendo ambos configurarse en máquinas servidoras independientes. Esta arquitectura fue mejorada a su vez con una arquitectura multicapa, en donde cada nivel físico se responsabiliza de una función del sistema.

En el caso de estudio presentado, el cliente (alumno) es un cliente ligero, ya que cualquier alumno desde su PC con un navegador Web puede realizar el acceso, a través de la intermediación de un servidor Web, un servidor de lógica de negocio y un servidor de base de datos.



Menú principal de sistema de Autogestión mediante browser

Consultas:	**Inscripciones**:
• Aula virtual	• Materias
• Actualización de datos personales	• Exámenes
• Actividades obligatorias	• Reinscripciones
• Informe de calificaciones	
• Exámenes rendidos e inscriptos	Envío de actividades obligatorias
• Materias inscriptas	
• Correspondencia entre planes de estudio	Cancelación de inscripciones
• Equivalencias	Servicio de clasificados
• Trámites realizados	Servicio de email
• Mesa de exámenes	Servicio de noticias institucionales

• Horarios de tutorías	Avisos interactivos
• Régimen de correlatividades	
• Resoluciones Rectorales	
• Reglamento	

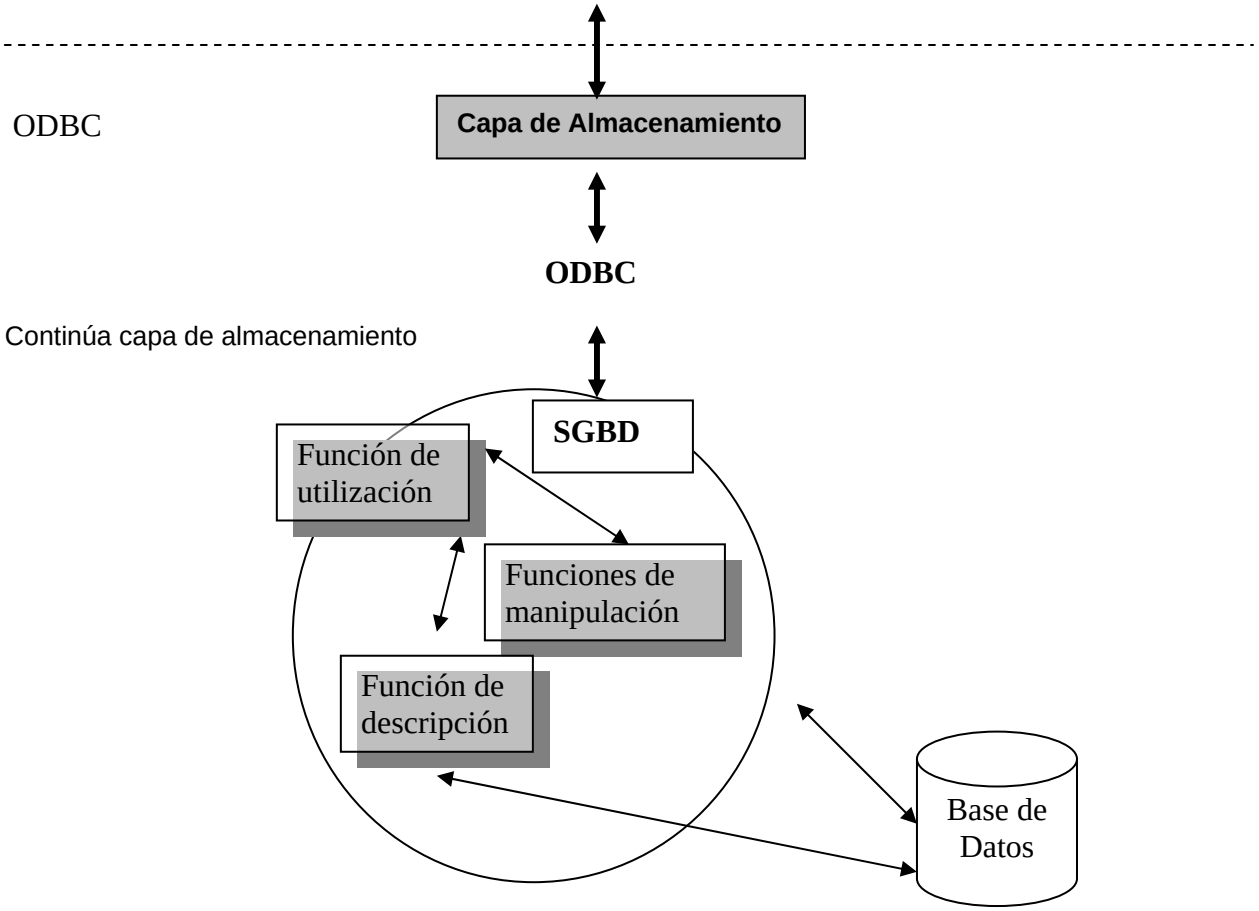


Figura 41– Modelo de tres capas de la aplicación Web

b) Diagrama de componentes de la aplicación web

En la figura 41 es posible observar cómo funciona la nueva aplicación luego de la reingeniería. En ella, a diferencia de la GUI Application, se muestran marcadas las tres capas del proceso, tal como se detallan a continuación.

b.1 Cliente

El cliente accede al sistema de manera remota mediante un SO con interfaz gráfica a través de Internet. En esta capa, el componente principal es el Browser de navegación el cual despliega páginas HTML encargadas de la interfaz con el usuario y que se representa mediante el componente HTML UIAutogestión. Estas páginas contienen componentes de animación FlashPlayer ComponentesDinamicos lo cual permite que el sitio no sean páginas frías y desagradables a la vista del usuario. El componente ASP LogicaUIAutoegestión contiene algunos componentes de JavaScript que se descargan y funcionan en la máquina del cliente.

b.2 Servidor Web

En esta capa, el componente principal es IIS (Internet Information Server) para el caso de la tecnología Microsoft. En ese componente se encuentran las políticas de acceso y concurrencia de clientes remotos al uso de la aplicación. El componente ASP LogicaUIAutogestion contiene la lógica de negocio y, en conjunto, con el componente

ActiveX Entidades que realiza la gestión entre las entidades del sistema utilizan los objetos COM+ para el manejo de datos. Luego se gestiona el acceso a la Base de Datos mediante el componente ADO AccesoDatos que toma la funcionalidad de conceder el permiso de acceso por medio de drivers ODBC.

b.3 Servidor de Base de Datos

En esta última capa, el componente principal es el Motor de Base Datos que contiene el servicio principal para la gestión de datos. Los componentes asociados son las Tablas donde se halla la organización de la información, las Vistas donde se encuentran las consultas más comunes, y los Procedimientos Almacenados donde están todos los Script para la gestión de datos. Todo este manejo lo realiza T-SQL (Transact SQL) propio del motor utilizado. En esta capa el SGBD también maneja la concurrencia, mediante permisos otorgados por el DBA a determinado número de operaciones por vez; de esta manera, se garantiza seguridad y consistencia en la información evitando que los servidores colapsen.

La figura 42 corresponde a la vista física de las tres capas de la aplicación Web. Es posible observar la reutilización de los componentes de la Aplicación GUI en la Aplicación Web. Dentro de estos nodos, se ejecutan procesos, servicios y/o componentes y sus relaciones de dependencia, como por ejemplo el Internet Explorer “muestra” la página HTML que corresponde a la presentación o Interfaz del Usuario de la aplicación.

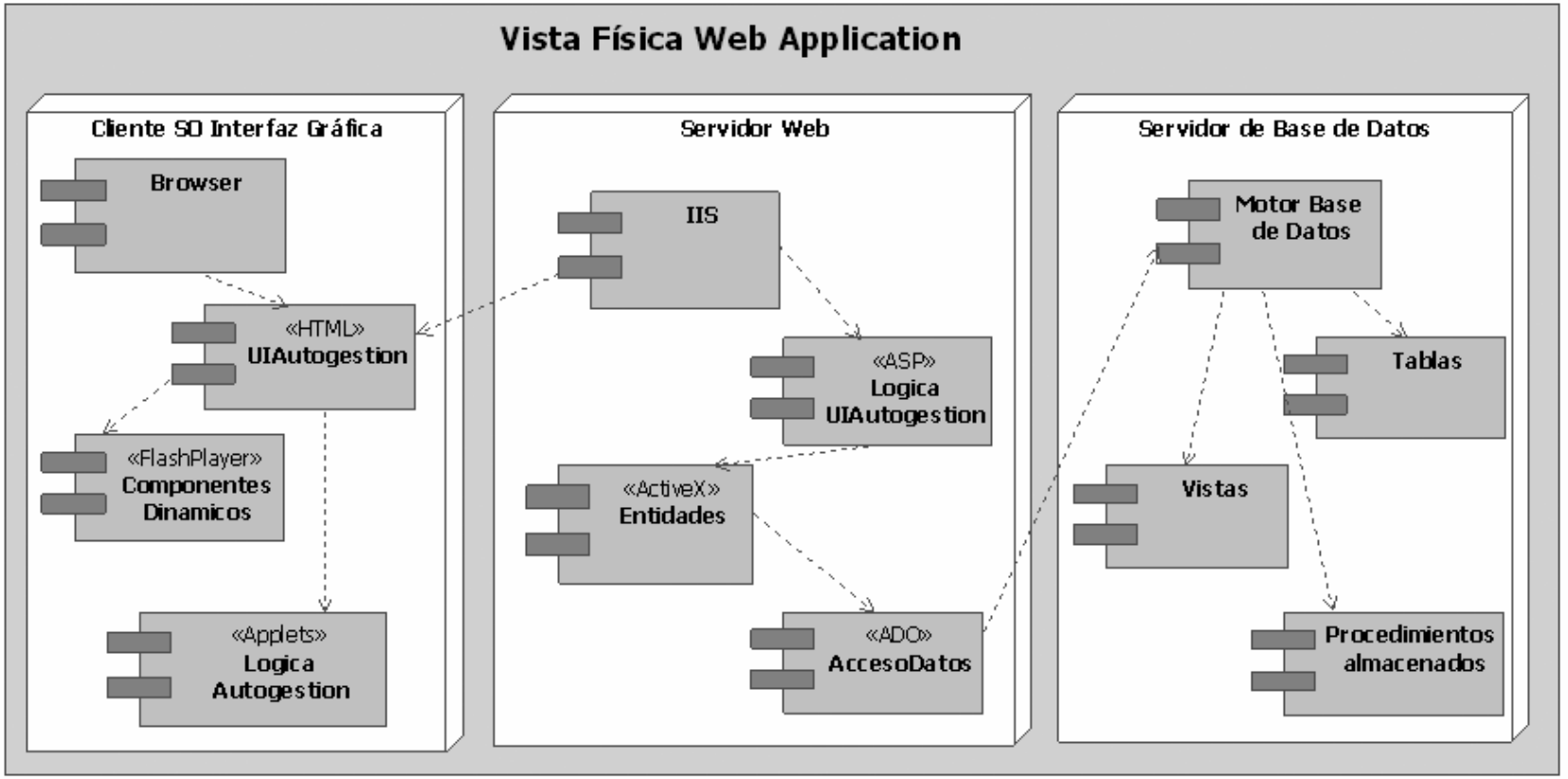


Figura 42 – Vista Física de la Web Application

CAPÍTULO 6. APLICACIÓN DE CASOS DE PRUEBA AL CASO DE ESTUDIO

6.1. Estudio comparativo de los casos de uso de ambos sistemas

Sobre la base de la metodología explicitada en el capítulo precedente, se confeccionaron los casos de uso principales de ambos sistemas. La carpeta Desarrollo de Casos de Prueba- que se adjunta, permite visualizar, de modo completo, las diferencias entre los mismos. En la carpeta citada, se presentan las planillas con:

- casos de uso principales del sistema anterior,
- casos de uso del sistema Web,
- casos de prueba funcionales para ambos sistemas, y
- cheklist de pruebas de interface.

Las diferencias mencionadas en las planillas desarrolladas para ambos sistemas provienen de tres fuentes perfectamente identificables:

a) Nueva funcionalidad incorporada al nuevo sistema
b) Modificación de la funcionalidad existente en el sistema anterior
c) Baja de alguna funcionalidad del sistema anterior

A continuación, se analizan los siguientes casos de uso:

1. Ingresar al Sistema de Autogestión
2. Seleccionar opciones del Menú Principal
3. Registrar Inscripción en Materia

6.1.1. Estudio comparativo del caso de uso "Ingresar al Sistema de Autogestión"

Se presenta un estudio comparativo de las plantillas del caso de uso "Ingresar al Sistema de Autogestión" en ambos sistemas. A los fines de su lectura, se resaltan con color diferente las principales diferencias encontradas.

- En las plantillas del primer caso de uso -"Ingresar al sistema de Autogestión"-, pueden observarse las modificaciones de funcionalidad por **cambio de tecnología** respecto al mismo caso de uso desarrollado para la aplicación Web: a) el sistema de autogestión debe estar activado (en el caso de la aplicación GUI), o b) el sistema de autogestión debe estar publicado (en el caso de la aplicación Web), lo cual se observa claramente en las precondiciones del caso de uso.

El **agregado de funcionalidad** se encuentra implícito en los siguientes casos:

- En la aplicación Web el alumno puede elegir la facultad en la cual cursa una carrera, mientras que en el sistema anterior sólo estaban habilitados para ingresar al sistema de autogestión los alumnos de la ex Facultad de Educación a Distancia, actual Facultad de Ciencias de la Administración.
- En las condiciones que debe tener un alumno para conservar su condición de tal, si el alumno fue dado de baja, puede igualmente acceder al resto de las otras opciones incorporadas en la aplicación Web, más allá de las consultas que le permitía solamente el sistema anterior.

A continuación se desarrollan ambas plantillas en las tablas 13 y 14:

Tabla 13 – Caso de Uso: "Ingresar al sistema de autogestión" en la aplicación GUI

Nombre del Use Case: Ingresar al Sistema de Autogestión.	**Nro. de Orden**: 1
Prioridad: ☐ Alta ☐ Media ☐ Baja	
Actor Principal: Alumno	**Actor Secundario:**
Tipo de Use Case: ☐ Concreto ☐ Abstracto	
Objetivo: Que el alumno ingrese al sistema de autogestión	
Precondiciones: El sistema de autogestión debe estar activado.	
Post-Condiciones: Que el alumno ingrese al Sistema de Autogestión pudiendo visualizar todas las opciones que tiene disponible como alumno de la facultad – El Caso de Uso se cancela si los datos de ingreso del usuario no son válidos (el usuario no es alumno de la facultad).	
Curso Normal	**Alternativo**
1. El Caso de Uso comienza con el ingreso de un Usuario a la pantalla principal del sistema de autogestión, y el sistema solicita la identificación del usuario que desea ingresar al Sistema.	
2. El alumno ingresa su identificación de usuario (su número de documento).	2.1. El usuario no es alumno de la facultad. 2.1.1. El sistema informa al usuario de dicha situación y vuelve a mostrar la pantalla de inicio. 2.1.2. Se cancela el caso de uso.
3. El sistema solicita la contraseña del alumno.	
4. El sistema valida la contraseña para el ingreso a las opciones del Sistema de Autogestión y si el usuario es o ha sido alumno de la facultad (usuario activo).	4.1 El usuario no es alumno de la facultad. 4.1.1 El Sistema informa al usuario de dicha situación y vuelve a desplegar la pantalla de inicio. 4.1.2 Se cancela el Caso de Uso.
6. El sistema permite visualizar al alumno una pantalla de selección de carrera (listado de carreras correspondientes a la Facultad de Educación a Distancia).	
6. El alumno selecciona la carrera a consultar.	
7. El sistema valida que el alumno ingresado esté activo para la carrera seleccionada.	
8. El alumno se encuentra activo para la carrera seleccionada. ***Observaciones***: *Condiciones: 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la facultad correspondiente.* *2- Abonar el arancel establecido.* El sistema permite visualizar un menú con todas las opciones disponibles de acceso para el alumno en	8.1 El alumno se encuentra dado de baja para la carrera seleccionada. 8.1.1 El sistema permite las siguientes opciones de menú: * Consultas * Cronograma de actividades * Consulta de situación arancelaria

dicha carrera. El sistema ofrece las siguientes opciones: * Consultas * Inscripción en examen * Inscripción en materias * Listado de mesas de examen * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin 9. Fin del Caso de Uso	* Modificación de PIN * Fin

Tabla 14 – Caso de Uso: "Ingresar al sistema de autogestión" en la aplicación Web

Nombre del Use Case: Ingresar al Sistema de Autogestión.	**Nro. de Orden**: 1
Prioridad: ☐ Alta ☐ Media ☐ Baja	
Actor Principal: alumno	**Actor Secundario:**
Tipo de Use Case: ☐ Concreto ☐ Abstracto	
Objetivo: Que el alumno ingrese al sistema de autogestión	
Precondiciones: El sitio Web del IUA (www.iua.edu.ar) debe estar publicado.	
Post-Condiciones: Que el alumno ingrese al Sistema de Autogestión pudiendo visualizar todas las opciones que tiene disponible como alumno de la universidad – El Caso de Uso se cancela si los datos de ingreso del usuario no son válidos (el Usuario no es alumno de la Universidad).	
Curso Normal	**Alternativo**
1. El Caso de Uso comienza con el ingreso de un usuario a la página principal del sitio Web del IUA (www.iua.edu.ar) y la selección de la facultad a la cual el usuario desea ingresar (Facultad de Ingeniería o Ciencias de la Administración).	
2. El sistema solicita la identificación del usuario que desea ingresar a la Universidad.	
3. El alumno ingresa su identificación de usuario (su número de documento) y contraseña.	
4. El sistema valida el usuario y contraseña para el ingreso al Sistema de Autogestión y el usuario es o ha sido alumno de la universidad (Usuario Activo).	4.1 El usuario no es alumno de la universidad. 4.1.1 El sistema informa al usuario de dicha situación. 4.1.2 El sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web. 4.1.3 Se cancela el Caso de Uso.
6. El sistema permite visualizar al alumno una pantalla de selección de carrera (listado de carreras	

correspondientes a la Facultad seleccionada).	
6. El alumno seleccionar la carrera a consultar.	
7. El sistema valida que el alumno ingresado esté Activo para la carrera seleccionada.	
8. El alumno se encuentra activo para la carrera seleccionada. ***Observaciones***: *Condiciones:* *1. Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente.* *2. Abonar el arancel establecido.* ***Para mantener la condición de alumno, se deberá:*** ***a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del año académico en la Modalidad Presencial o del año académico p****ersonal, en la Modalidad a Distancia.* *b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario* ***Aeronáutico en general y en cada facultad en particular.*** ***c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera.*** ***d. Efectuar la reinscripción anual.*** El sistema permite visualizar un menú con todas las opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Inscripción * Enviar * Cancelar * Servicios * Comunicarse	8.1 El alumno se encuentra dado de baja para la carrera seleccionada. 8.1.1 El sistema permite las siguientes opciones de menú: * General * Consultas * Servicios * Comunicarse
9. Fin del Caso de Uso.	

6.1.2. Estudio comparativo del caso de uso "Seleccionar opciones del Menú Principal"

En las tablas 15 y 16 correspondientes a este caso de estudio, se observa la incorporación de funcionalidad en la aplicación Web (Tabla 15), y la posibilidad de selección de la facultad, ya que en la Tabla 14, sólo el menú está disponible para alumnos de la Facultad de Educación a Distancia.

Tabla 15 – Caso de Uso: "Seleccionar opciones del Menú Principal" en la aplicación GUI

Nombre del Use Case: Seleccionar opciones del Menú Principal.	**Nro. de Orden**: 2
Prioridad: ☐ Alta ☐ Media ☐ Baja	
Actor Principal: **a**lumno	**Actor Secundario:**
Tipo de Use Case: ☐ Concreto ☐ Abstracto	
Objetivo: Que el alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión	
Precondiciones: Que el alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre activo.	
Post-Condiciones: El alumno ingresa a alguna de las opciones que ofrece el menú principal del Sistema de Autogestión.	
Curso Normal	**Alternativo**
1. El Caso de Uso comienza cuando el alumno ingresa a la página del Sistema de Autogestión, seleccionando una Carrera de la Facultad Distancia	
2. El Sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * Consultas * Inscripción en examen * Inscripción en materia * Eliminar inscripción en materia * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	
3. El alumno selecciona la opción "Consultas" del menú principal del Sistema de Autogestión.	3.1. El alumno selecciona otra opción del menú principal. 3.2. El alumno sale del menú principal
. El Sistema despliega las siguientes opciones: * Actividades obligatorias * Exámenes finales	

* Inscripto en materias * Inscripto en examen * Equivalencias Fin para abandonar Consultas	
6. El alumno selecciona alguna opción del menú de consultas	6.1 El alumno selecciona la opción FIN para volver al menú principal 6.1.1 El sistema vuelve a la pantalla principal del sistema, permitiendo el ingreso del alumno a otras secciones del Sistema. 6.1.2 Se cancela el Caso de Uso.
6. El sistema cierra la sesión del usuario logueado permitiendo abrir una nueva sesión.	
7. Fin del Caso de Uso.	

Tabla 16 – Caso de Uso: "Seleccionar opciones del Menú Principal" en la aplicación Web

Nombre del Use Case: Seleccionar opciones del Menú Principal.	**Nro. de Orden**: 2
Prioridad: ☐ Alta ☐ Media ☐ Baja	
Actor Principal: Alumno	**Actor Secundario:**
Tipo de Use Case: ☐ Concreto ☐ Abstracto	
Objetivo: Que el Alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión	
Precondiciones: Que el Alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre Activo.	
Post-Condiciones: El Alumno ingresa a alguna de las opciones que ofrece el menú principal del Sistema de Autogestión.	
Curso Normal	**Alternativo**
1. El Caso de Uso comienza cuando el Alumno ingresa a la página del Sistema de Autogestión, seleccionando una Carrera de una Facultad seleccionada	
2. El Sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * General * Consultas * Inscripción * Enviar * Cancelar * Servicios * Comunicarse	

3. El Alumno selecciona la opción “General” del menú principal del Sistema de Autogestión.	3.1 El Alumno selecciona la opción “Consulta” del menú principal del Sistema de Autogestión. 3.1.1 El Sistema despliega las siguientes opciones: * Acceso al Aula Virtual. * Datos Personales * Actividades Obligatorias * Informe de Calificaciones * Exámenes * Materias Inscripto * Correspondencia * Equivalencias * Trámites realizados * Materiales de estudio 3.1.2 Se cancela el Caso de Uso.
4. El Sistema despliega las siguientes opciones: * Volver a página principal * Opciones: Modificar contraseña Modificar pregunta secreta Publicar un aviso Comunicación por e-mail automática Cambiar apariencia del Explorer * Contenedor Descarga de utilitarios (adobe, Explorer, etc.) * Cambiar carrera * Cerrar sesión	
6. El Alumno selecciona la opción “Cerrar Sesión”.	6.1 El Alumno selecciona la opción “Volver a página principal” 6.1.1 El Sistema vuelve a la página principal del sitio, permitiendo el ingreso del alumno a otras secciones del Sistema. 6.1.2 Se cancela el Caso de Uso.
6. El Sistema cierra la sesión del usuario logueado permitiendo abrir una nueva sesión.	
7. Fin del Caso de Uso.	

En esta última tabla se observa la incorporación de funcionalidad en la aplicación Web, en algunos casos acompañada por el cambio de tecnología, como las opciones: Publicar un aviso, Cambiar apariencia del Explorer, Comunicación por e-mail automática y Descarga de utilitarios.

6.1.3. Estudio comparativo del caso de uso: “Registrar Inscripción en materia” en ambas aplicaciones

A continuación, se detallan las diferencias fundamentales remarcadas en los mismos:

- Las condiciones para inscribirse en materias exigen en el sistema nuevo que el alumno haya realizado previamente su inscripción o reinscripción según corresponda, como puede observarse en la Tabla 17.
- En el sistema anterior no se controlaba la fecha de cursado de las materias de acuerdo a la fecha de inscripción, sino que se ofrecía la inscripción a todas las materias en las que el alumno estuviese habilitado, independientemente de la existencia de cursos abiertos para las mismas, según puede observarse en Tabla 16.
- Los comprobantes de inscripción en materias se emitían por la tickeadora (ver Tabla 16); mientras que en la aplicación Web (ver Tabla 17) la diferencia

tecnológica permite enviarlos en forma automática por el correo institucional del alumno.

Tabla 17 – Caso de Uso: "Registrar Inscripción en Materia" en la aplicación GUI

Nombre del Use Case: Registrar Inscripción en Materia	**Nro. de Orden**: 3
Prioridad: ☐ Alta ☐ Media ☐ Baja	
Actor Principal: alumno	**Actor Secundario:**
Tipo de Use Case: ☐ Concreto ☐ Abstracto	
Objetivo: Que el alumno realice la inscripción de una o varias materias.	
Precondiciones: Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.	
Post-Condiciones: El alumno se inscribe a una o varias materias de una carrera. El Caso de Uso se cancela si la fecha de inscripción en Materias no es válida, si el alumno no tiene ninguna materia vigente para inscribirse en el próximo semestre o si el alumno no confirma la inscripción.	
Curso Normal	**Alternativo**
1. El Caso de Uso comienza cuando el alumno selecciona la opción "Inscripción en Materia". *Condiciones de Inscripción en materia:* a) Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes. b) No tener deuda de cuotas pendientes.	
2. El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día.	2.1 El alumno debe pagar los abonos correspondientes a la carrera. 2.1.1 El sistema informa de dicha situación al alumno. 2.1.2 Se cancela el Caso de Uso.
3. El sistema verifica que el alumno cumpla con las condiciones de cursado vigentes y el alumno la cumple.	3.1 El alumno no cumple con condiciones de cursado vigentes. 3.1.1 El sistema informa de dicha situación al alumno. 3.1.2 Se cancela el Caso de Uso.
4. El sistema despliega una lista con las distintas materias que el alumno puede cursar.	
6. El alumno selecciona la/s materias a inscribirse.	
6. El sistema solicita al alumno que seleccione el curso en el cual realizará len Materia.	
7. El alumno selecciona el curso en el cual desea inscribirse.	

8. El sistema solicita al alumno si desea confirmar la Inscripción, y el alumno confirma.	8.1 El alumno no confirma la inscripción la/s materias. 8.1.1. Se cancela el Caso de Uso.
9. El sistema registra la inscripción y emite el/los comprobantes de inscripción correspondientes.	
10. Fin del Caso de Uso.	

Tabla 18 – Caso de Uso: "Registrar Inscripción en Materia" en la aplicación Web

Nombre del Use Case: Registrar Inscripción en Materia	**Nro. de Orden**: 3
Prioridad: Alta Media Baja	
Actor Principal: alumno	**Actor Secundario:**
Tipo de Use Case: Concreto Abstracto	
Objetivo: Que el alumno realice la inscripción de una o varias materias.	
Precondiciones: Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre Activo para una carrera seleccionada.	
Post-Condiciones: El alumno se inscribe a una o varias materias de una carrera. El Caso de Uso se cancela si la fecha de inscripción en Materias no es válida, si el alumno no tiene ninguna materia vigente para inscribirse en el próximo semestre o si el alumno no confirma la inscripción.	
Curso Normal	**Alternativo**
1. El Caso de Uso comienza cuando el alumno selecciona la opción "Inscripción".	
2. El sistema permite visualizar la opción "Materias".	
3. El alumno selecciona la opción "Materias". ***Observaciones***: *Condiciones de Inscripción en Materia:* *a. Estar inscripto o reinscripto en la carrera, según el caso.* *b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes.* *c. Estar en condición de habilitado.*	
4. El sistema valida que la fecha del día actual en la cual desea inscribirse el alumno se encuentre dentro de los períodos de inscripción en Materias publicados. Y la fecha es válida.	4.1 La fecha en la cual el alumno desea inscribirse no es una fecha válida. 4.1.1 El Sistema informa de dicha situación al alumno. 4.1.2 Se cancela el caso de Uso.
6. El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día.	6.1 El alumno debe pagar los abonos correspondientes a la carrera. 6.1.1 El sistema informa de dicha situación al alumno.

	6.1.2 Se cancela el Caso de Uso.
6. El sistema verifica que el alumno cumpla con el régimen de correlatividades y condiciones de cursado vigentes y el alumno las cumples.	6.1 El alumno no cumple con el régimen de correlatividades y condiciones de cursado vigentes. 6.1.1 El sistema informa de dicha situación al alumno. 6.1.2 Se cancela el Caso de Uso.
7. El sistema despliega una lista con las distintas materias que el alumno puede cursar ese semestre.	
8. El alumno tiene materias para cursar en el semestre de la consulta.	8.1 El alumno no tiene ninguna materia disponible para cursar en el semestre en el cual se realiza la consulta. 8.1.1 El sistema notifica al alumno de dicha situación. 8.1.2 Se cancela el Caso de Uso.
9. El alumno selecciona la/s materias a inscribirse.	
10. El Sistema solicita al alumno que seleccione el curso en el cual realizará len materia.	
11. El alumno selecciona el curso en el cual desea inscribirse.	
12. El sistema solicita al alumno si desea confirmar la Inscripción, y el alumno confirma.	12.1 El alumno no confirma la inscripción la/s materias. 12.1 Se cancela el Caso de Uso.
13. El sistema registra la inscripción y envía al mail institucional del alumno los comprobantes de inscripción correspondientes.	
14. Fin del Caso de Uso.	

6.2. Estudio comparativo de los casos de prueba de ambos sistemas

La utilización de la técnica de Casos de Uso para definir el comportamiento del sistema, determina que los diferentes cursos de acción normales y alternativos de dichos casos originen las condiciones de pruebas funcionales. El desarrollo de casos de prueba es directamente conducido por los casos de uso; pero siempre teniendo en cuenta los requerimientos funcionales y las particularidades o reglas de negocio para que el enfoque de los mismos se traduzca luego en pruebas eficaces. Para cada caso de uso se crea un conjunto de casos de prueba. Cada caso de prueba representa un escenario del caso de uso.

Cada condición de prueba debe definir clara y brevemente una situación del sistema que se desea probar. Las condiciones son descripciones generales, no tienen el detalle de qué datos se utilizarán para probar; eso se especificará en los pasos de prueba, al igual que los resultados esperados de su ejecución.

Se desarrollaron los casos de prueba para los casos de uso detallados en el punto anterior, tomándose para cada uno de ellos la prueba con datos válidos e inválidos, tal como puede observarse en detalle a continuación.

6.2.1. Estudio comparativo de los casos de prueba: "Ingresar al Sistema de Autogestión" en ambas aplicaciones

Para el caso de uso: **Ingresar al Sistema de Autogestión** se presentan los siguientes casos de prueba analizados:

- Ingresar al sistema con datos válidos. Alumno activo.
- Ingresar al Sistema con Datos Válidos. El alumno se encuentra dado de baja para una carrera seleccionada.
- Ingresar al sistema con datos inválidos. El actor no es alumno de la Universidad.

Los casos de prueba referentes al mismo caso de uso aunque se plantean de la misma forma, difieren en las condiciones de prueba, los resultados esperados y, las respuestas del sistema; pues, como se vio en el caso de uso descripto en el punto anterior la diferencia tecnológica y el agregado de funcionalidad modifican y aumentan las condiciones de prueba, alterando los resultados esperados. Por ejemplo, en el caso de que el usuario olvidara su contraseña el sistema anterior le permitía recuperarla, para lo cual los únicos datos solicitados eran código postal y fecha de nacimiento.

En la aplicación Web, el grado de seguridad es mayor, y, se solicita además conocer la pregunta secreta que consignó al registrarse por primera vez, etc., dejando abierta la posibilidad al sistema de enviar la contraseña por el correo institucional.

Puede observarse también que los datos de entrada en estos casos de prueba se mantienen idénticos en condiciones semejantes, pero al cambiar la funcionalidad en ambas aplicaciones, se mantienen las condiciones a probar aunque los resultados esperados sean diferentes, y, los errores graves que se identifican permanecen en ambas aplicaciones. En color, se resaltan las diferencias encontradas en los casos de prueba de ambas aplicaciones.

A continuación, en la Tabla 19, pueden observarse en la aplicación GUI las diferencias sobre las condiciones de inscripción y arancelamiento, así como las opciones disponibles de acceso al sistema con datos válidos e inválidos, para lo cual se plantean para el mismo caso de prueba, las condiciones 1, 2 y 3.

Tabla 19 – Caso de Prueba para el caso de uso: "Ingresar al sistema de autogestión" en la aplicación GUI

Cond. 1: Ingreso al sistema con datos válidos						
Ingresar al sistema con datos válidos. Alumno activo						
Precondiciones	*El Sistema de Autogestión debe estar activado*					
	Ciclo:	1				
	Fecha:	23-04-2007				
	Resultado esperado	Que el alumno ingrese al sistema de autogestión				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El sistema muestra al usuario la pantalla principal del sistema de autogestión.		El sistema muestra la pantalla de inicio solicitándole al usuario el ingreso de su DNI	OK		
2	El alumno ingresa su identificación de usuario (su número de documento).	Usuario: 28254712	El sistema permite el ingreso de dicho campo y valida el usuario para el ingreso al Sistema de Autogestión. El sistema muestra una pantalla de ingreso de contraseña	OK		

3	El alumno ingresa su contraseña de usuario	Contraseña: 1234	El sistema permite el ingreso de dicho campo y valida el usuario y contraseña para el ingreso al Sistema de Autogestión. El sistema muestra una pantalla de selección de carrera.(listado de carreras correspondientes a la facultad)	OK		
4	El alumno selecciona una carrera a consultar y sistema verifica si el alumno cumple con las siguientes condiciones: 1- Está inscripto en la carrera o curso que se dicte en la Facultad de Educación a Distancia 2- Está al día con el pago del régimen arancelario (no se registra un atraso mayor a una cuota).		El Sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El Sistema permite visualizar un menú con todas las opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * Consultas * Inscripción en examen * Inscripción en materias * Listado de mesas de examen * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	OK		**Condiciones:** 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la facultad correspondiente. 2- Abonar el arancel establecido.

Cond. 2: Ingresar al Sistema con Datos Válidos.						
El alumno se encuentra dado de baja para una carrera seleccionada						
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
Precondiciones	*El sistema de Autogestión de alumnos debe estar activo*					
	Ciclo:	1				
	Fecha:	23-04-2007				

	Resultado esperado	Que el alumno ingrese al sistema de autogestión				
	Resultado obtenido	Error grave				
	Código de error					
1	El sistema muestra al usuario la pantalla principal del sistema de autogestión.		El sistema muestra la pantalla de inicio solicitándole al usuario el ingreso de su DNI	OK		
2	El alumno ingresa su identificación de usuario (su número de documento)	Usuario: 2800000	El sistema solicita la contraseña del usuario	OK		
3	El alumno ingresa su contraseña.	Contraseña: 1234	El sistema permite el ingreso de dicho campo y valida el usuario y contraseña para el ingreso al Sistema de Autogestión. El sistema muestra una pantalla de selección de carrera.(listado de carreras correspondientes a la Facultad)	OK		
4a	El alumno selecciona una carrera a consultar y el alumno No está inscripto o en la carrera o curso que se dicta en la facultad correspondiente.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * Consultas * Cronograma de Actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	Error grave El sistema muestra además las siguientes opciones del menú: * Inscripción en materia * Inscripción en examen * Listado de mesas de examen * Eliminar inscripción en materia		**Condiciones:** 1- Estar inscripto en la carrera o curso que se dicte en la facultad correspondiente. 2- Abonar el arancel establecido.

4b	El alumno selecciona una carrera a consultar y el alumno debe abonar el arancel establecido.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * Consultas * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	Error grave El sistema muestra además las siguientes opciones del menú: * Inscripción en materia * Inscripción en examen * Listado de mesas de examen * Eliminar inscripción en materia		Observaciones: Condiciones: 1- Estar inscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido.

Cond. 3: Ingreso al sistema con datos inválidos		
Ingresar al sistema con datos inválidos. El actor no es alumno de la universidad.		
Precondiciones	*El Sistema de Autogestión de alumnos debe estar activo*	
	Ciclo:	1
	Fecha:	23-04-2007
	Resultado esperado	Validación de los datos ingresados, y emisión de los mensajes de error correspondientes.
	Resultado obtenido	OK
	Código de error	

Id Paso	Descripción	<Valor>	Resultado esperado	Resultado obtenido	Código de error	Observaciones
1	El sistema muestra al usuario la pantalla principal del sistema de autogestión.		El sistema muestra la pantalla de inicio solicitándole al usuario el ingreso de su DNI	OK		
2	El alumno ingresa su identificación de usuario (su número de documento) inválido.	Usuario: (<Valor> no existente)	El sistema valida el dato ingresado y envía un mensaje informando el error, y permite corregirlo.	OK		
3a	El alumno ingresa una contraseña válida.	Contraseña: 1234	El sistema valida los datos ingresados. El sistema le permite al usuario acceder a la Autogestión.	OK		
3b	El alumno satura los campos de login.	Usuario: saturación Contraseña: saturacion	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la autogestión.	OK		
3c	El alumno ingresa caracteres especiales en los campos del login y acepta la operación.	Usuario: '/=%"(Contraseña: '&%)	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la autogestión.	OK		
3d	*El alumno ingresa espacios en blanco en el login y acepta la operación.*	Usuario: (Espacios en Blanco) Contraseña: (Espacios en Blanco)	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la autogestión.	OK		

3e	*El alumno no ingresa ningún <Valor> en los campos del login y acepta la operación.*	*Usuario: (Vacio)* *Contraseña: (Vacio)*	*El sistema valida los datos ingresados.* *El sistema envía un mensaje informando el error, y permite corregirlo.* *El sistema no le permite al usuario acceder a la autogestión.*	*OK*		

Se observa en la comparación de los casos de prueba de ambos sistemas, que se completa con la Tabla 20 de la aplicación Web, no sólo el agregado de funcionalidad en las opciones de menú, sino, como se verá a continuación, en las condiciones que debe cumplir un alumno para poder ingresar a las distintas opciones, controles que fueron incorporados al modificarse el Reglamento General del alumno. Entre dichas condiciones, es posible citar, como más relevante: la condición anual de **reinscripción** y la condición de **habilitado** (tener al día el pago de aranceles), ambos nuevos estados necesarios y suficientes para que un alumno conserve su condición de tal. Además se realiza el control de la cantidad de aplazos, como puede verse en la Tabla 19, en las condiciones 1 2 y 3.

Tabla 20 – Casos de Prueba para el caso de uso: "Ingresar al sistema de autogestión" en la aplicación Web

Cond. 1: Ingreso al sistema con datos válidos						
Ingresar al sistema con datos válidos. Alumno activo						
Precondiciones	*El sitio Web del IUA (www.iua.edu.ar) debe estar publicado*					
	Ciclo:	1				
	Fecha:	23-04-2007				
	Resultado Esperado	Que el alumno ingrese al sistema de autogestión				
	Resultado Obtenido	OK				
	Código de Error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno solicita al Sistema ingresar a la página principal del sitio Web del IUA (www.iua.edu.ar).	Ingresar en la URL www.iua.edu.ar	El sistema muestra la página principal de la Universidad y muestra las distintas facultades a ingresar: * Facultad de Ingeniería * Ciencias de la Administración	OK		
2	El alumno selecciona la Facultad a ingresar	Seleccionar "Facultad de Ingeniería"	El Sistema solicita la identificación del usuario que desea ingresar a la Universidad.	OK		
3	El alumno ingresa su identificación de usuario (su número de documento) y contraseña.	Usuario: 28254712 Contraseña: 123	El sistema permite el ingreso de dichos campo y valida el usuario y contraseña para el ingreso al Sistema de Autogestión. El sistema muestra una pantalla de selección de carrera.(listado de	OK		

			carreras correspondientes a la Facultad seleccionada).			
4	El alumno selecciona una carrera a consultar y con las siguientes condiciones: 1.Está inscripto o reinscripto en la carrera o curso que se dicte en la facultad correspondiente. 2. Está al día con el pago del régimen arancelario (no se registra un atraso mayor a 1 cuota). 3. Tiene aprobadas 2 asignaturas que corresponden a la currícula de la carrera que cursa dentro del año académico en la modalidad presencial o del año académico Personal, en la Modalidad a Distancia. 4. Cumple con las disposiciones y normativas vigentes en el IUA en general y en cada facultad en particular. 6. Tiene menos de 10 aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El Sistema permite visualizar un menú con todas las opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Inscripción * Enviar * Cancelar * Servicios * Comunicarse			**Condiciones:** 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido. Para mantener la condición de alumno, se deberá: a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del Año Académico en la Modalidad Presencial o del Año Académico Personal, en la Modalidad a Distancia. b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario Aeronáutico en general y en cada Facultad en particular. c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera. d. Efectuar la reinscripción anual.

Cond. 2: Ingresar al sistema con datos válidos.						
El alumno se encuentra dado de baja para una carrera seleccionada						
Precondiciones	*El sitio Web del IUA (www.iua.edu.ar) debe estar publicado*					
	Ciclo:	1				
	Fecha:	23-04-2007				
	Resultado esperado	Que el alumno ingrese al sistema de autogestión				
	Resultado obtenido	Error grave				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado Esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno solicita al Sistema ingresar a la página principal del sitio Web del IUA (www.iua.edu.ar).	Ingresar en la URL www.iua.edu.ar	El sistema muestra la página principal de la Universidad y muestra las distintas facultades a ingresar: * Facultad de Ingeniería * Ciencias de la Administración	OK		
2	El alumno selecciona la Facultad a ingresar	Seleccionar "Facultad de Ingeniería"	El sistema solicita la identificación del usuario que desea ingresar a la Universidad.	OK		
3	El alumno ingresa su identificación de usuario (su número de documento) y contraseña.	Usuario: 28254712 Contraseña: 123	El sistema permite el ingreso de dichos campo y valida el usuario y contraseña para el ingreso al Sistema de Autogestión. Muestra una pantalla de selección de carrera.(listado de carreras correspondientes a la Facultad seleccionada).	OK		

4a	El alumno selecciona una Carrera a consultar y el alumno no está inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Servicios * Comunicarse	**Error grave** **El sistema muestra además las siguientes opciones del menú:** *** Inscripción** *** Enviar** *** Cancelar**		**Condiciones:** 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido. Para mantener la condición de alumno, se deberá: a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del Año Académico en la Modalidad Presencial o del Año Académico Personal, en la Modalidad a Distancia. b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario Aeronáutico en general y en cada Facultad en particular. c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera. d. Efectuar la reinscripción anual.

4b	El alumno selecciona una carrera a consultar y el alumno debe abonar el arancel establecido.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Servicios * Comunicarse	Error grave El sistema muestra además las siguientes opciones del menú: * Inscripción * Enviar * Cancelar		Observaciones: Condiciones: 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido. Para mantener la condición de alumno, se deberá: a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del Año Académico en la Modalidad Presencial o del Año Académico Personal, en la Modalidad a Distancia. b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario Aeronáutico en general y en cada Facultad en particular. c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera. d. Efectuar la reinscripción anual.

4c	El alumno selecciona una carrera a consultar y el alumno tiene aprobada solo una asignatura correspondiente a la currícula de la carrera que cursa dentro del año académico en la modalidad presencial o del año académico personal, en la modalidad a distancia.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Servicios * Comunicarse	Error grave El sistema muestra además las siguientes opciones del menú: * Inscripción * Enviar * Cancelar		Observaciones: Condiciones: 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido. Para mantener la condición de alumno, se deberá: a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del Año Académico en la Modalidad Presencial o del Año Académico Personal, en la Modalidad a Distancia. b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario Aeronáutico en general y en cada Facultad en particular. c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera. d. Efectuar la reinscripción anual.

4d	El alumno selecciona una carrera a consultar y el alumno no cumple con las disposiciones y normativas vigentes en el IUA en general y en cada Facultad en particular.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El Sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Servicios * Comunicarse	Error grave El sistema muestra además las siguientes opciones del menú: * Inscripción * Enviar * Cancelar		**Condiciones:** 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido. Para mantener la condición de alumno, se deberá: a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del Año Académico en la Modalidad Presencial o del Año Académico Personal, en la Modalidad a Distancia. b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario Aeronáutico en general y en cada Facultad en particular. c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera. d. Efectuar la reinscripción anual.

4e	El alumno selecciona una carrera a consultar y el alumno tiene once aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera.	Seleccionar: Ingeniería en Sistemas.	El sistema valida que el alumno ingresado esté activo para la carrera seleccionada. El alumno se encuentra dado de baja para la carrera seleccionada. El sistema permite visualizar un menú con las siguientes opciones disponibles de acceso para el alumno en dicha carrera. El sistema ofrece las siguientes opciones: * General * Consultas * Servicios * Comunicarse	Error grave El sistema muestra además las siguientes opciones del menú: * Inscripción * Enviar * Cancelar		**Condiciones:** 1- Estar inscripto o reinscripto en la carrera o curso que se dicte en la Facultad correspondiente. 2- Abonar el arancel establecido. Para mantener la condición de alumno, se deberá: a. Aprobar, como mínimo, dos asignaturas correspondientes a la currícula de la carrera que cursa dentro del Año Académico en la Modalidad Presencial o del Año Académico Personal, en la Modalidad a Distancia. b. Cumplir con las disposiciones y normativas vigentes en el Instituto Universitario Aeronáutico en general y en cada Facultad en particular. c. No tener más de diez aplazos en exámenes finales de las materias correspondientes a los tres primeros años de la carrera. d. Efectuar la reinscripción anual.

Cond. 3: Ingreso al Sistema con Datos Inválidos						
Ingresar al sistema con datos inválidos. El actor no es alumno de la universidad.						
Precondiciones	*El sitio Web del IUA (www.iua.edu.ar) debe estar publicado*					
	Ciclo:	1				
	Fecha:	23-04-2007				
	Resultado Esperado	Validación de los datos ingresados, y emisión de los mensajes de error correspondientes.				
	Resultado Obtenido	OK				
	Código de Error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado Obtenido**	**Código de error**	**Observaciones**
1	El alumno solicita al Sistema ingresar a la página principal del sitio Web del IUA (www.iua.edu.ar).	Ingresar en la URL www.iua.edu.ar	El sistema muestra la página principal de la universidad y muestra las distintas facultades a ingresar: * Facultad de Ingeniería * Ciencias de la Administración	OK		
2	El alumno selecciona la Facultad a ingresar	Seleccionar "Facultad de Ingeniería"	El sistema solicita la identificación del usuario que desea ingresar a la universidad.	OK		
3a	El alumno ingresa una identificación de usuario inválida y una contraseña válida.	Usuario: (<Valor> no existente) Contraseña: 123	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web.	OK		

3b	El alumno ingresa una identificación de usuario válida y una contraseña inválida.	Usuario: 28254712 Contraseña: (<Valor> no existente)	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web.	OK		
3c	El alumno satura los campos de login.	Usuario: saturación Contraseña: saturación	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web.	OK		
3d	El alumno ingresa caracteres especiales en los campos del login y Acepta la operación	Usuario: '/=%"(Contraseña: '&%)	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El Sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web	OK		

3e	El alumno ingresa espacios en blanco en el login y Acepta la operación.	Usuario: (Espacios en Blanco) Contraseña: (Espacios en Blanco)	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web.	OK		
3f	El alumno no ingresa ningún <Valor> en los campos del login y acepta la operación.	Usuario: (Vacio) Contraseña: (Vacio)	El sistema valida los datos ingresados. El sistema envía un mensaje informando el error, y permite corregirlo. El sistema no le permite al usuario acceder a la Autogestión pero le permite acceder a la información general del sitio Web.	OK		

6.2.2. Estudio comparativo de los casos de prueba: "Seleccionar opciones del Menú Principal" en ambas aplicaciones

Se detalla a continuación el agregado de funcionalidad en ambas versiones del caso de prueba "Seleccionar opciones del Menú Principal" analizado en la tabla 21, correspondiente a los casos de prueba del sistema anterior y en la Tabla 22 los casos de prueba del sistema Web.

Tabla 21– Casos de Prueba para el caso de uso: "Seleccionar opciones del Menú Principal" en la aplicación GUI

Cond.1: Seleccionar opción Consultas del Menú Principal y luego Cerrar.						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre Activo.*					
	Ciclo:	1				
	Fecha:	23-05-2006				
	Resultado esperado	Que el alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno ingresa al Sistema de Autogestión, seleccionando una carrera de la facultad	Seleccionar: Ingeniería en Sistemas.	El Sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * Consultas * Inscripción en examen * Inscripción en materia * Eliminar inscripción en materia * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	OK		

2	Alumno selecciona la opción “Consultas” del menú principal del Sistema de Autogestión.	Seleccionar la opción "Consultas"	El sistema despliega las siguientes opciones: * Consultas * Inscripción en examen * Inscripción en materia * Eliminar inscripción en materia * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	OK		
3	El alumno selecciona la opción “FIN".	Seleccionar la opción "FIN"	El Sistema retorna al menú principal del sistema de autogestión	OK		

Cand. 2: Seleccionar opción Inscripción en examen del menú principal		
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre activo.*	
	Ciclo:	1
	Fecha:	23-05-2006
	Resultado esperado	que el alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión.
	Resultado Obtenido	OK
	Código de Error	

Id Paso	Descripción	<Valor>	Resultado esperado	Resultado obtenido	Código de error	Observaciones
1	El alumno ingresa al Sistema de Autogestión, seleccionando una Carrera de la Facultad de Educ. a Distancia	Seleccionar: Ingeniería en Sistemas.	El Sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * Consultas * Inscripción en examen * Inscripción en materia * Eliminar inscripción en materia * Cronograma de actividades * Consulta de situación arancelaria * Modificación de PIN * Fin	OK		
2	El alumno selecciona la opción “Inscripción en examen” del menú principal del Sistema de Autogestión.	Seleccionar la opción "Inscripción en examen"	El sistema despliega el listado de materias en que el alumno está en condiciones de rendir examen	OK		

Tabla 22– Casos de Prueba para el caso de uso: "Seleccionar opciones del Menú Principal" en la aplicación WEB

Cond. 1: Seleccionar opción General del Menú Principal y luego Cerrar.						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre Activo.*					
	Ciclo:	1				
	Fecha:	23-05-2006				
	Resultado esperado	que el alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno ingresa a la página del Sistema de Autogestión, seleccionando una carrera de una facultad seleccionada	Seleccionar: Ingeniería en Sistemas.	El sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * General * Consultas * Inscripción * Enviar * Cancelar * Servicios * Comunicarse	OK		

2	alumno selecciona la opción “General” del menú principal del Sistema de Autogestión.	Seleccionar la opción "General"	El sistema despliega las siguientes opciones: * Volver a página principal * Opciones: Modificar contraseña Modificar pregunta secreta Publicar un aviso Comunicación por e-mail automática Cambiar apariencia del Explorer * Contenedor Descarga de utilitarios (adobe, Explorer, etc.) * Cambiar carrera * Cerrar sesión	OK		
3	El Alumno selecciona la opción “Cerrar Sesión".	Seleccionar la opción "Cerrar"	El Sistema cierra la sesión del usuario logueado permitiendo abrir una nueva sesión.	OK		

Cond. 2: Seleccionar opción general del menú principal y luego volver a página principal		
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre activo.*	
	Ciclo:	1
	Fecha:	23-05-2006
	Resultado esperado	que el alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión
	Resultado obtenido	OK
	Código de error	

Id Paso	Descripción	<Valor>	Resultado esperado	Resultado obtenido	Código de error	Observaciones
1	El alumno ingresa a la página del Sistema de Autogestión, seleccionando una carrera de una facultad seleccionada	Seleccionar: Ingeniería en Sistemas.	El sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * General * Consultas * Inscripción * Enviar * Cancelar * Servicios * Comunicarse	OK		
2	Alumno selecciona la opción “General” del menú principal del Sistema de Autogestión.	Seleccionar la opción "General"	El sistema despliega las siguientes opciones: * Volver a página principal * Opciones: Modificar contraseña Modificar pregunta secreta Publicar un aviso Comunicación por e-mail automática Cambiar apariencia del Explorer * Contenedor Descarga de utilitarios (adobe, Explorer, etc.) * Cambiar carrera * Cerrar sesión	OK		
3	El Alumno selecciona la opción “Cerrar Sesión".	Seleccionar la opción “Volver a página principal”	El sistema vuelve a la página principal del sitio, permitiendo el ingreso del alumno a otras secciones del Sistema.	OK		

Cond. 3: Seleccionar opción Consultar del Menú Principal						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y el mismo se encuentre Activo.*					
	Ciclo:	1				
	Fecha:	23-05-2006				
	Resultado esperado	que el alumno pueda ingresar a las opciones que se encuentran disponibles en el menú principal del sistema de autogestión				
	Resultado Obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno ingresa a la página del Sistema de Autogestión, seleccionando una carrera de una facultad seleccionada	Seleccionar: Ingeniería en Sistemas.	El sistema permite visualizar las siguientes opciones del menú principal del Sistema de Autogestión: * General * Consultas * Inscripción * Enviar * Cancelar * Servicios * Comunicarse	OK		

2	El alumno selecciona la opción “Consulta” del menú principal del Sistema de Autogestión.	Seleccionar la opción "Consultar"	El sistema despliega las siguientes opciones: * Acceso al Aula Virtual. * Datos Personales * Actividades Obligatorias * Informe de Calificaciones * Exámenes * Materias Inscripto * Correspondencia * Equivalencias * Trámites realizados *Materiales de estudio	OK		

Mientras que en el caso de prueba del sistema anterior, tal como se observa en la Tabla 21, las opciones son:

- Consultas
- Inscripción en examen
- Inscripción en materia
- Eliminar inscripción en materia
- Cronograma de actividades
- Consulta de situación arancelaria
- Modificación de PIN
- Fin

En la aplicación Web, como se presenta en la Tabla 22, las opciones del menú principal son:

- General
- Consultas
- Inscripción
- Enviar

- Cancelar
- Servicios
- Comunicarse

Como resultado del cambio de tecnología, se han agregado en el nuevo sistema algunas opciones tales como:

- Publicar un aviso
- Comunicación por e-mail automática
- Cambiar apariencia del Explorer
- Contenedor
- Descarga de utilitarios (adobe, Explorer, etc.)
- Acceso al Aula Virtual

6.2.3. Estudio comparativo de los casos de prueba: "Registrar Inscripción en materia"

En estos casos de prueba se presenta un agregado de funcionalidad ya que en el sistema anterior no se controlaba el período en el cual el alumno podía inscribirse, además de las condiciones para la inscripción de acuerdo con el régimen de correlatividades vigente y a las exigencias de inscripción anual y habilitación según el nuevo reglamento del alumno; por lo tanto en el nuevo sistema se controla que el alumno pueda inscribirse si lo hace dentro de las fechas habilitadas para tal fin.

En el sistema anterior se presentaban todas las materias en que el alumno podía inscribirse, aunque algunas de ellas no tuvieran curso asignado, como puede observarse en la Tabla 23. En el nuevo sistema, por el contrario, sólo se muestran las materias habilitadas para el cursado en el semestre de la consulta, según se ve en la Tabla 24.

Por otro lado, al existir un cambio en la tecnología, la inscripción y los comprobantes de inscripción pueden obtenerse directamente de Internet (Tabla 24). Otro agregado de funcionalidad consiste en el registro automático de todos los trámites realizados por el alumno (Tabla 24).

Tabla 23– Casos de Prueba para el caso de uso: "Registrar Inscripción en materia" en la aplicación GUI

Cond. 1: Registrar inscripción en materia con datos válidos						
Registrar Inscripción en Materia. El alumno no adeuda abonos y tiene materias para cursar en el semestre de la consulta						
Precondiciones	*Que el Alumno se haya logueado correctamente al sistema de autogestión y se encuentre Activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado Esperado	Que el Alumno realice la inscripción de una o varias materias				
	Resultado Obtenido	OK				
	Código de Error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**

1	El alumno selecciona la opción "Inscripción en Materia" y cumple con las siguientes condiciones: 1- Está inscripto en la carrera.	Seleccionar la opción "Inscripción en Materia".	El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día. El sistema despliega una lista con las distintas materias que el alumno puede cursar.	OK		
2	El alumno selecciona la/s materias a inscribirse.	Seleccionar materia: "Auditoria y Evaluación de Sistemas"	El sistema solicita al alumno que seleccione el curso en el cual realizará la materia.	OK		
3	El alumno selecciona el curso en el cual desea inscribirse.	Seleccionar: Curso	El sistema solicita al alumno si desea confirmar la Inscripción.	OK		
4	El alumno confirma la inscripción	Presionar "Aceptar"	El sistema registra la inscripción y emite el/los comprobantes de inscripción correspondientes.	OK		

Cond. 2: Registrar inscripción en materia. El alumno no se encuentra al día con los abonos orrespondientes a la carrera	
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*

	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obt.	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción "Inscripción en Materias" pero no se encuentra al día con los abonos correspondientes a la carrera. Además, el alumno cumple con las siguientes condiciones: 1- Está inscripto en la carrera. 2- Cumple con el régimen de correlatividades y condiciones de cursado vigentes.	Seleccionar la opción "Inscripción en Materias".	El Sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno debe pagar los abonos correspondientes a la carrera. El Sistema informa de dicha situación al alumno.		OK	**Condiciones de Inscripción en Materia:** a. Estar inscripto en la carrera, b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes.

Cond. 3: Registrar inscripción en materia. El alumno no tiene materias para cursar en el semestre de la consulta						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción “Inscripción en Materias”.	Seleccionar la opción “Inscripción en Materias”.	El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día. El sistema determina que el alumno no tiene ninguna materia disponible para cursar El sistema notifica al alumno de dicha situación.	OK		**Condiciones de Inscripción en Materia:** a. Estar inscripto o en la carrera. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes.

Cond. 4: Registrar inscripción en materia. El alumno puede inscribirse en materias pero no confirma la inscripción						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción “Inscripción en Materias”.	Seleccionar la opción “Inscripción en Materias”.	El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día. El sistema despliega una lista con las distintas materias que el alumno puede cursar independientemente del semestre.	OK		**Condiciones de inscripción en materia:** a. Estar inscripto en la carrera. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes.
2	El alumno selecciona la/s materias a inscribirse.	Seleccionar Materia: "Análisis Matemático"	El sistema solicita al alumno que seleccione el curso en el cual realizará la materia.	OK		

3	El alumno selecciona el curso en el cual desea inscribirse.	Seleccionar: curso	El sistema solicita al alumno si desea confirmar la Inscripción.	OK		
4	El alumno no confirma la inscripción	Presionar "Cancelar"	El sistema cierra la aplicación	OK		

Cond. 5: Registrar inscripción en materia. El alumno puede inscribirse en materias pero no hay curso asignado en el semestre de la consulta						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**

1	El alumno selecciona la opción “Inscripción en Materias”.	Seleccionar la opción “Inscripción en Materias”.	El sistema verifica que el Alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día. El sistema despliega una lista con las distintas materias que el alumno puede cursar independientemente del semestre.	OK		**Condiciones de inscripción en materia:** a. Estar inscripto en la carrera. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes.
2	El alumno selecciona la/s materias a inscribirse.	Seleccionar materia: "Análisis Matemático"	El sistema verifica que la materia no tiene curso asignado e informa de dicha situación al alumno	OK		
3	El alumno selecciona materia o abandona la aplicación	Presionar "Cancelar"	El sistema cierra la Aplicación	OK		

Tabla 24– Casos de Prueba para el caso de uso: "Registrar Inscripción en materia" en la aplicación Web

Cond. 1: Registrar inscripción en materia. El alumno está dentro del período de inscripción, no adeuda abonos y tiene materias para cursar en el semestre de la consulta						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción "Inscripción".	Seleccionar la opción "Inscripción".	El sistema permite visualizar la opción "Materias".	OK		
2	El alumno selecciona la opción "Materias" y Cumple con las siguientes condiciones: 1- Está inscripto en la carrera. 2- Cumple con el	Seleccionar la opción "Materias".	El sistema valida que la fecha del día actual en el cual desea inscribirse el alumno se encuentre dentro de los períodos de inscripción en materias publicados. Y la fecha es válida.El sistema verifica que el alumno tenga adía los abonos correspondientes a la carrera y el alumno se encuentra al día.	OK		***Condiciones de inscripción en materia:*** *a. Estar inscripto o reinscripto en la carrera, según el caso.* *b. Cumplimentar el régimen de correlatividades y*

	régimen de correlatividades y condiciones de cursado vigentes. 3-Está en condición de habilitado		El sistema despliega una lista con las distintas materias que el alumno puede cursar ese semestre.			*condiciones de cursado vigentes.* *c. Estar en condición de habilitado*
3	El alumno selecciona la/s materias a inscribirse.	Seleccionar materia: "Auditoria y Evaluación de Sistemas"	El sistema solicita al Alumno que seleccione el curso en el cual realizará len Materia.	OK		
4	El alumno selecciona el curso en el cual desea inscribirse.	Seleccionar: Curso	El sistema solicita al alumno si desea confirmar la Inscripción.	OK		
5	El alumno confirma la inscripción	Presionar "Aceptar"	El sistema registra la inscripción y envía al mail institucional del alumno los comprobantes de inscripción correspondientes.	OK		

Cond. 2: Registrar inscripción en materia. El alumno está fuera del período de inscripción						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre Activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El sistema permite visualizar la opción “Materias”.	OK		

2	El alumno selecciona la opción “Materias” pero está fuera del período de inscripción. Además, el alumno cumple con las siguientes condiciones:1- Está inscripto en la carrera.2- Cumple con el régimen de correlatividades y condiciones de cursado vigentes.3- Está en condición de habilitado	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en la cual desea inscribirse el Alumno se encuentre dentro de los períodos de inscripción en Materias publicados. La fecha en la cual el alumno desea inscribirse no es una fecha válida.El sistema informa de dicha situación al alumno.	OK		**Condiciones de inscripción en materia:** a. Estar inscripto o reinscripto en la carrera, según el caso. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes. c. Estar en condición de habilitado.

Cond.3: Registrar inscripción en materia. El alumno se encuentra al día con los abonos correspondientes a la carrera						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado**	**Código**	**Observaciones**

				obtenido	**de error**	
1	El alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El sistema permite visualizar la opción “Materias”.	OK		
2	El alumno selecciona la opción “Materias” pero No se encuentra al día con los abonos correspondientes a la carrera (No está habilitado). Además, el alumno cumple con las siguientes condiciones:1- Está inscripto en la carrera.2- Cumple con el régimen de correlatividades y condiciones de cursado vigentes.	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en el cual desea inscribirse el Alumno se encuentre dentro de los períodos de inscripción en Materias publicados. Y la fecha es válida.El sistema verifica que el Alumno tenga al día los abonos correspondientes a la carrera y el alumno debe pagar los abonos correspondientes a la carrera. El sistema informa de dicha situación al alumno.	OK		**Condiciones de inscripción en materia:** a. Estar inscripto o reinscripto en la carrera, según el caso. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes. c. Estar en condición de habilitado.

Cond. 4: Registrar inscripción en materia. El alumno no cumple con el régimen de correlatividades y condiciones de cursado vigentes			
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*		
	Ciclo:	1	
	Fecha:	29-04-2007	

	Resultado esperado	Que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El sistema permite visualizar la opción “Materias”.	OK		
2	El alumno selecciona la opción “Materias” pero no cumple con el régimen de correlatividades y condiciones de cursado vigentes. Además, el alumno cumple con las siguientes condiciones: 1- Está inscripto en la carrera. 2- Está en condición de habilitado.	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en el cual desea inscribirse el Alumno se encuentre dentro de los períodos de inscripción en Materias publicados. Y la fecha es válida.El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día.El sistema verifica que el alumno cumpla con el régimen de correlatividades y condiciones de cursado vigentes y el alumno no las cumple. El sistema informa de dicha situación al alumno.	OK		**Condiciones de Inscripción en Materia:** a. Estar inscripto o reinscripto en la carrera, según el caso. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes. c. Estar en condición de habilitado.

Cond. 5: Registrar inscripción en materia. El Alumno está dentro del período de inscripción pero no tiene materias para cursar en el semestre de la consulta						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado obtenido**	**Código de error**	**Observaciones**
1	El alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El sistema permite visualizar la opción “Materias”.	OK		

2	El alumno selecciona la opción “Materias”.	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en el cual desea inscribirse el Alumno se encuentre dentro de los períodos de inscripción en materias publicados. Y la fecha es válida.El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día.El sistema determina que el alumno no tiene ninguna materia disponible para cursar en el semestre en el cual se realiza la consulta.El sistema notifica al alumno de dicha situación.	OK		**Condiciones de inscripción en materia:** a. Estar inscripto o reinscripto en la carrera, según el caso. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes. c. Estar en condición de habilitado.

Cond. 6: Registrar inscripción en materia. El alumno está dentro del período de inscripción pero no confirma la inscripción						
Precondiciones	*Que el alumno se haya logueado correctamente al sistema de autogestión y se encuentre activo para una carrera seleccionada.*					
	Ciclo:	1				
	Fecha:	29-04-2007				
	Resultado esperado	que el alumno realice la inscripción de una o varias materias				
	Resultado obtenido	OK				
	Código de Error					
Id Paso	**Descripción**	**<Valor>**	**Resultado esperado**	**Resultado**	**Código**	**Observaciones**

				obtenido	**de error**	
1	El alumno selecciona la opción “Inscripción”.	Seleccionar la opción “Inscripción”.	El sistema permite visualizar la opción “Materias”.	OK		
2	El alumno selecciona la opción “Materias”.	Seleccionar la opción “Materias”.	El sistema valida que la fecha del día actual en el cual desea inscribirse el alumno se encuentre dentro de los períodos de inscripción en materias publicados. Y la fecha es válida.El sistema verifica que el alumno tenga al día los abonos correspondientes a la carrera y el alumno se encuentra al día.El sistema despliega una lista con las distintas materias que el alumno puede cursar ese semestre.	OK		**Condiciones de Inscripción en Materia:** a. Estar inscripto o reinscripto en la carrera, según el caso. b. Cumplimentar el régimen de correlatividades y condiciones de cursado vigentes. c. Estar en condición de habilitado.
3	El alumno selecciona la/s materias a inscribirse.	Seleccionar Materia: "Análisis Matemático"	El sistema solicita al alumno que seleccione el curso en el cual realizará la materia.	OK		
4	El alumno selecciona el curso en el cual desea inscribirse.	Seleccionar: Curso	El sistema solicita al alumno si desea confirmar la Inscripción.	OK		
5	*El alumno no confirma la inscripción*	*Presionar "Cancelar"*	*El sistema cierra la aplicación*	*OK*		

Puede observarse en la Tabla 24 cómo se incrementa el número de casos de prueba en la aplicación Web con respecto a la aplicación GUI (Tabla 23), debido al control de período de inscripción en materias y a las condiciones de habilitación para su cursado.

El caso de prueba “Registrar Inscripción en examen” tiene en el nuevo sistema las mismas condiciones diferenciadas: el alumno debe inscribirse en el período fijado para las fechas de examen con una antelación de hasta 48 hs. anteriores al día del examen y debe cumplir con las condiciones de regularidad, correlatividad y estar habilitado para rendir de acuerdo al reglamento vigente.

En todas las tablas observamos además el ítem Ciclo dentro de los casos de prueba, éste se incrementa aún cuando no existe condición de error en el ciclo de prueba anterior, debido a que si se encontraron errores en al menos un caso de prueba anterior, es necesario realizar un nuevo ciclo para soportar las pruebas de integración de sistema. Por ejemplo, al haber encontrado un error en el Caso de Prueba 2, se debe correr otro ciclo más probando **todos** los casos de prueba especificados, para verificar que TODA la funcionalidad para cada uno de los casos de pruebas se mantiene consistente.

La cantidad de ciclos y qué casos de prueba incluir en cada uno depende exclusivamente de lo que se haya definido en el Plan de Prueba, según puede verse en el Anexo3 y del tipo de pruebas a realizar.

En los casos en que no existe un plan de prueba (cuando la metodología no es muy formal) y cuando se decide (por un problema de tiempos) que se van a correr ciclos de prueba sólo a los casos de pruebas que tengan error, es **recomendable** poner en los casos de prueba que NO tienen condición de error en el ciclo anterior, el estado OK (como si se hubiera corrido un ciclo) ya que es un compromiso por parte del área de Testing especificar qué esos casos de pruebas siguen SIN error.

6.3. Estudio comparativo de la Interfaz de ambos sistemas

De acuerdo a lo desarrollado en el capítulo 4, se presentan en los puntos siguientes las interfaces de las principales funcionalidades de ambos sistemas y un checklist para analizar si las mismas se corresponden a los atributos determinados en el caso de uso correspondiente.

6.3.1. Estudio comparativo de la interfaz: "Ingresar al sistema de autogestión" en ambas aplicaciones

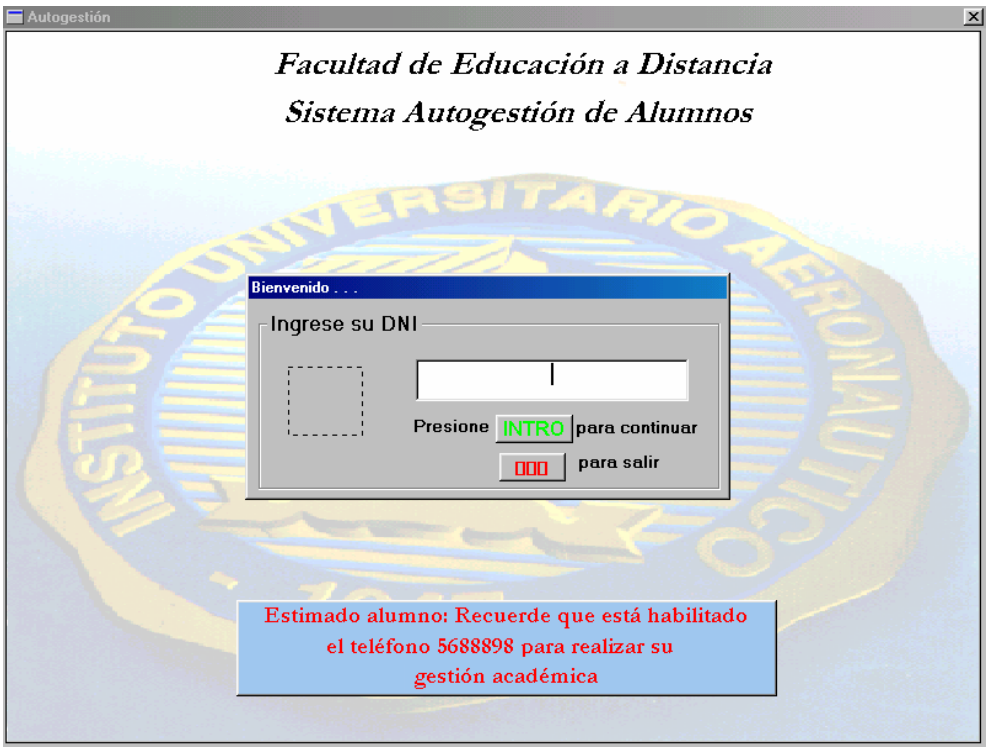


Figura 43 – Primera Pantalla para Ingresar al sistema de Autogestión aplicación GUI

Figura 44 – Segunda Pantalla para Ingresar al sistema de Autogestión aplicación GUI

Como puede observarse en las figuras 43 y 44, en la aplicación GUI se necesita displayar dos pantallas para que el alumno pueda identificarse en el sistema, mientras que en la aplicación Web, como se presenta en la Figura 45, en el portal de entrada del sitio, seleccionando la facultad, el alumno puede identificarse con su usuario y contraseña.

Figura 45 – Pantalla única para Ingresar al sistema de Autogestión aplicación WEB

Como ya se expresara en el capítulo 5 al analizar la arquitectura de ambas aplicaciones, la ventaja fundamental de la interfaz Web, a través del browser, la constituye el hecho de que el alumno se convierte en un cliente ligero, ya que desde cualquier PC puede acceder al sistema sin necesidad de trasladarse al lugar donde está alojada la aplicación como ocurría en el caso del sistema anterior implementado.

Se presenta a continuación en la Tabla 25, una herramienta checklist para los lineamientos estándar de la interface con los distintos ítems a considerar por los arquitectos de componentes y testeadores, según se explicó en el capítulo 5.

Tabla 25– Checklist del Caso de uso/ Caso de prueba "Ingresar al sistema de autogestión"

Módulo:	**Build 1.0**	
Interfaz / Pantalla:	**Menú Principal**	
Caso de Uso Asociado:	Ingresar al Sistema de Autogestión	
Confeccionado por:	**Maria Elena Ciolli**	
Revisado por:	**Jorgelina Aguirre Sotelo**	
Observaciones:	Deben repetirse los Cheklist para cada una de las pantallas relacionadas con el Módulo	
Item a Verificar	**Ciclo 1**	**Ciclo 2**
Logo de la Organización	OK	OK
Usuario Logueado	OK	OK
Sintaxis correcta de Títulos y Nombres de Campos	OK	OK

Correcta Navegabilidad en Menú	OK	OK
Correcto Orden de Ingreso de los Campos	OK	OK
Correcta Ubicación y Alineación de los Campos en la Pantalla	OK	OK
Correcta longitud de campos	NO OK Los cuadros de texto para el ingreso de usuario y contraseña son demasiado pequeños (en la aplic. Web)	OK
Formato de Letras Adecuado	OK	OK
Formatos de Códigos o ID (Solo Números y Letras en Mayúscula)	NO APLICA	NO APLICA
Formatos de Fecha y Hora (dd/mm/aaaa hh:mm)	NO APLICA	NO APLICA
Correcta Editabilidad de Campos	OK	OK
Especificación de Campos Obligatorios	NO OK	OK
Botón para editar datos	NO APLICA	NO APLICA
Botón ver datos asociados a la entidad que ha seleccionado	NO APLICA	NO APLICA
Botón Modificar el orden de una fila dentro de una lista compuesta	NO APLICA	NO APLICA
Botón para eliminar un dato que de una lista compuesta	NO APLICA	NO APLICA
Botón Buscar datos	NO APLICA	NO APLICA
Botón para Imprimir Datos	NO APLICA	NO APLICA
Botón de Volver	NO APLICA	NO APLICA
Botón de Ayuda	NO APLICA	NO APLICA
Botón Limpiar	NO APLICA	NO APLICA
Botón de Aceptación	OK	OK
Botón de Cancelación	OK	OK
Botón de Salir	OK	OK

La funcionalidad de la interfaz: “Ingresar al sistema de Autogestión” no presenta cambios significativos en cuanto a su funcionalidad específica, ya que en ambos casos permite disponer de la contraseña conociendo CP y fecha de nacimiento. En el sistema nuevo la seguridad es mucho mayor ya que, para poder acceder a una contraseña olvidada, se solicita además conocer la pregunta secreta, tal como se estila en todo manejo de contraseñas por Internet.

Los errores en los campos de ingreso se controlan en pantallas separadas en el primer caso y en la misma en el segundo, ya que se está accediendo al portal de la universidad en el nuevo sistema. En este último caso, cualquier usuario aunque no esté registrado puede acceder a información relativa a características propias de la universidad así como a novedades sobre cursos, etc.

6.3.2. Estudio comparativo de la interfaz: “Seleccionar opciones del menú principal” en ambas aplicaciones

En la Figura 46, se presenta la interface de opciones del Menú Principal de la aplicación GUI:

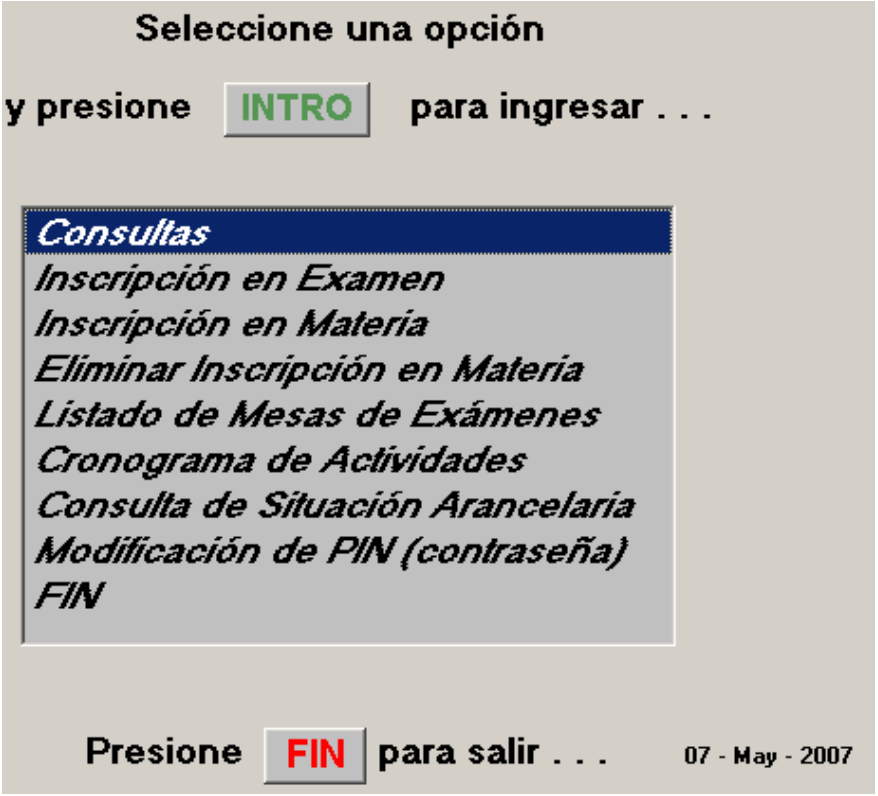


Figura 46 – Pantalla para Seleccionar opciones del menú principal en aplicación GUI

Esta interface presenta una diagramación muy básica, en la que se ofrecen al usuario las opciones fundamentales sin el valor agregado de ningún servicio adicional. Su confección fue diseñada en forma plana al solo efecto de acceder a la información digital.

La interface Web, que se muestra en la Figura 47, por el contrario, se basa en la corriente denominada de la “experiencia del usuario”, la que acentúa el refinamiento expresivo de los objetos presentes en una GUI, así como los mecanismos de interacción para generar una experiencia agradable. Se tienen en cuenta en su diseño, la utilización de los colores institucional, tónico y complementario, así como la iconografía y la tipografía adecuadas al rubro educativo.

Figura 47 – Pantalla para Seleccionar opciones del menú principal en aplicación WEB

El usuario dispone en esta pantalla de una gama de servicios y funciones a los cuales puede acceder directamente de la página principal, a través de menús desplegables, y obtener la ayuda “en línea” si fuera necesario.

6.3.3. Estudio comparativo de la interfaz: “Registrar inscripción en materia” en ambas aplicaciones

En la interfaz **Registrar inscripción en materia**, se presentan todas las materias que el alumno puede cursar, aunque no correspondan al semestre de la inscripción o no haya curso asignado a las mismas, situación que se observa al displayarse el mensaje correspondiente, como se observa en la figura 48.

La figura 48 muestra cómo el alumno debe seleccionar el curso o docente de la materia elegida para cursar, y, simultáneamente se solapa en la pantalla el comprobante de inscripción, antes que el alumno seleccione el curso o docente, lo cual debiera mostrarse en una nueva pantalla en forma asociada, una vez que el alumno haya confirmado su inscripción en la/s materias a cursar.

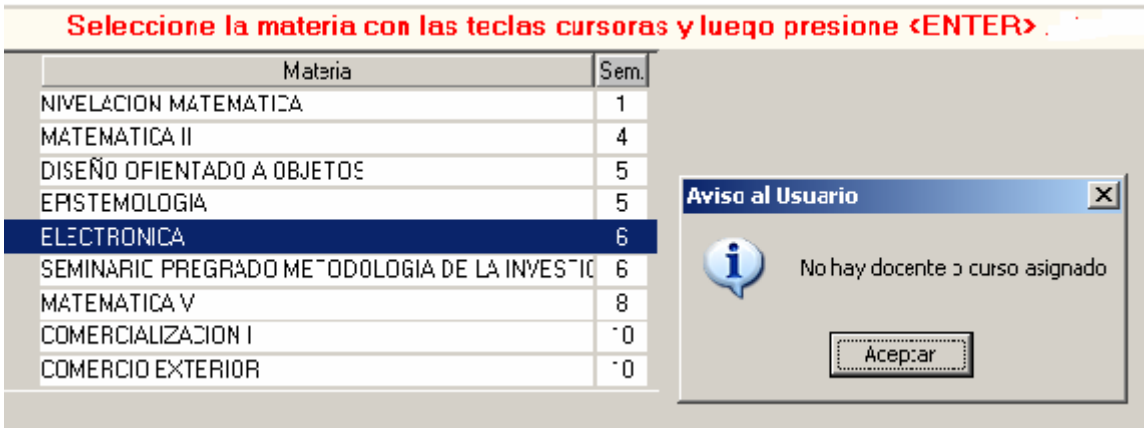


Figura 48 – Pantalla para Registrar Inscripción en materia sin curso asignado en la aplicación GUI

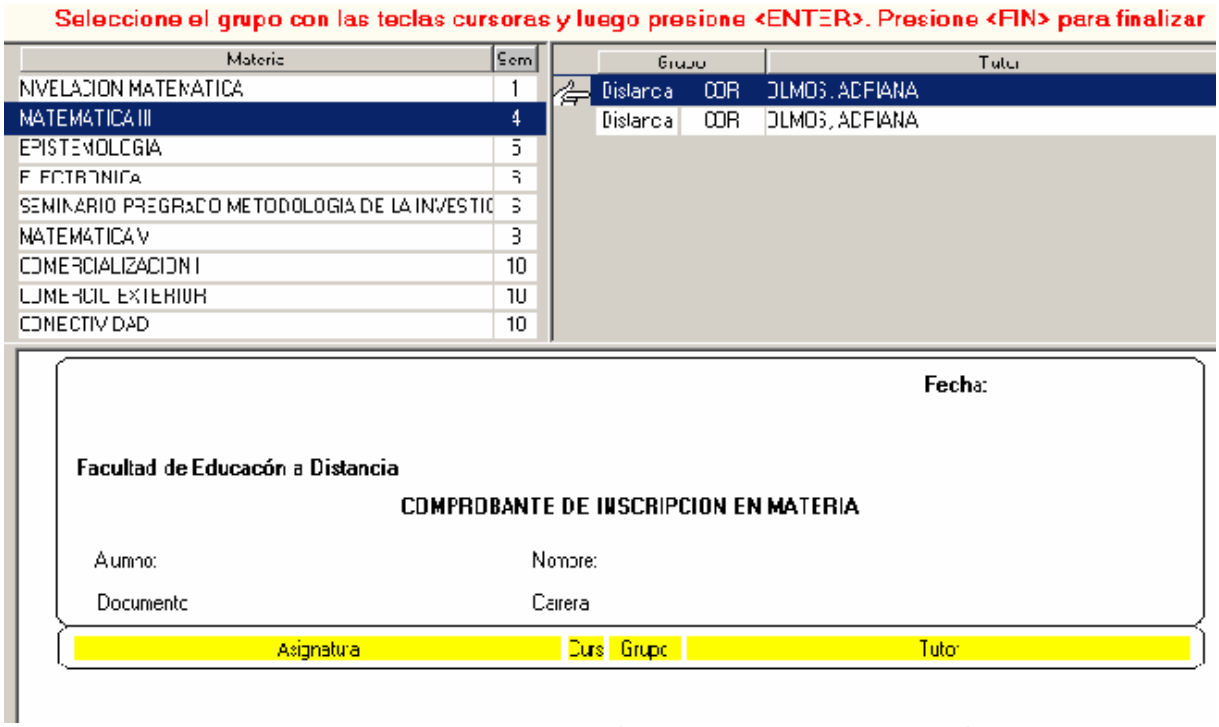


Figura 49– Pantalla para Registrar Inscripción en materia en la aplicación GUI

En la interfaces Web de la registración en materia, tal como se muestra en la Figura 50, se presentan solamente las materias que el alumno puede cursar con curso abierto, por lo tanto cualquiera de ellas que seleccione le habilitará la siguiente pantalla, lo que se observa en la Figura 50. Asimismo, podrá seleccionar el curso y el profesor.

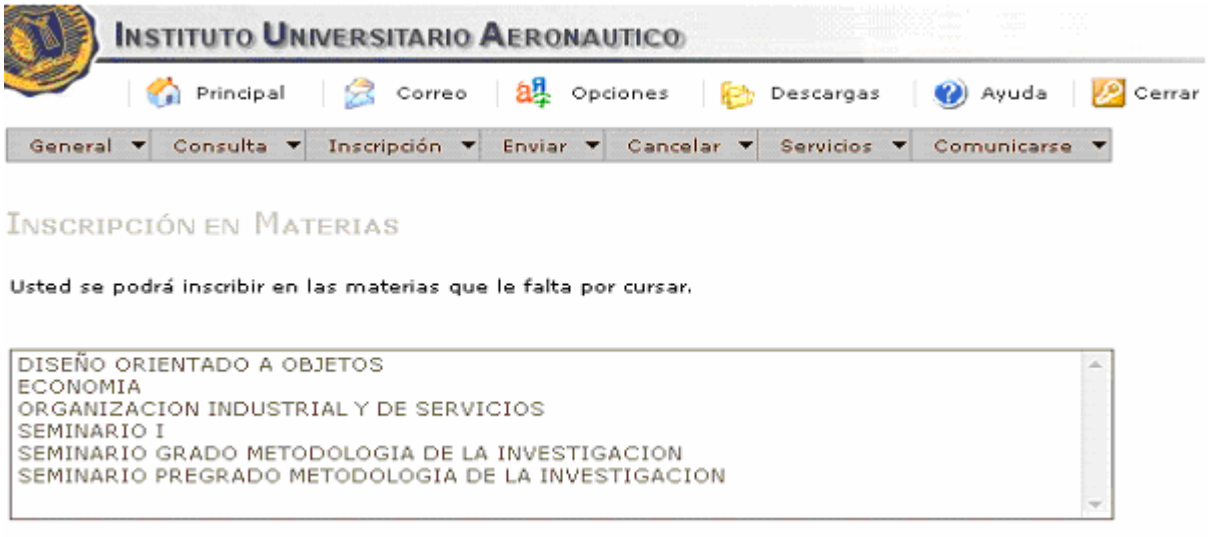


Figura 50 – Primera pantalla para Registrar Inscripción en materia en la aplicación WEB

El alumno puede retornar a la pantalla anterior o continuar con la inscripción. El comprobante de inscripción podrá imprimirse o será enviado por correo electrónico.

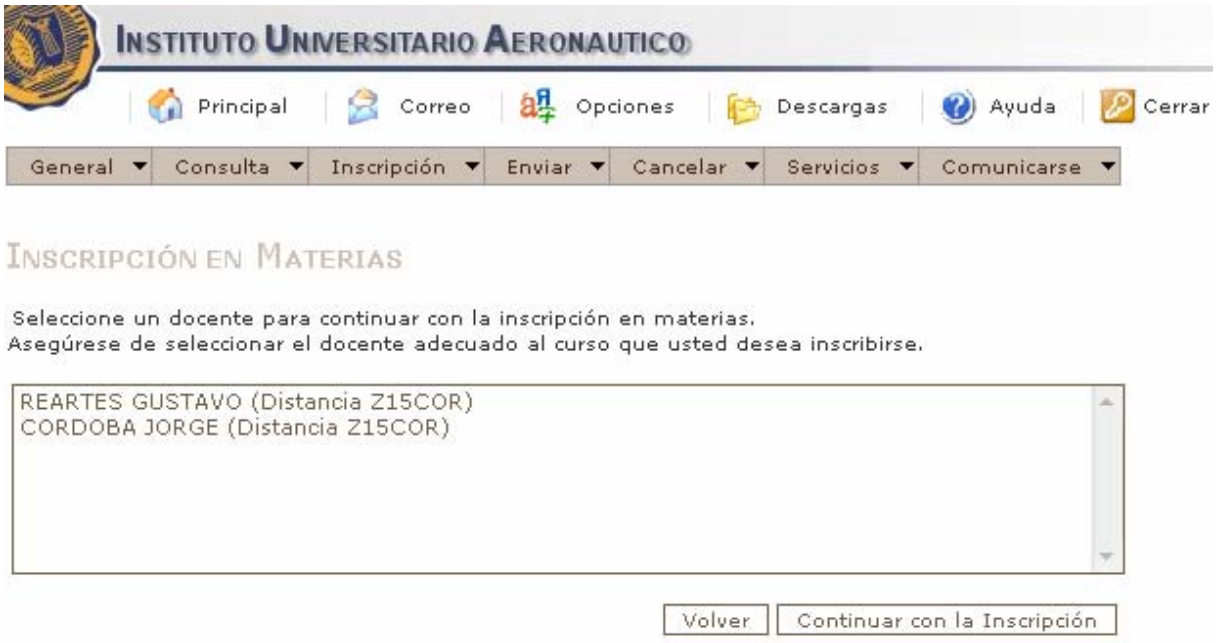


Figura 51 – Segunda pantalla para Registrar Inscripción en materia en la aplicación WEB

En general las interfaces Web del sistema han sido realizadas siguiendo los principios básicos del diseño Web, utilizando una adecuada diagramación, en la que se respetan los elementos básicos tanto en las pantallas primarias como en las secundarias. Si bien se sabe que las pantallas son 2D, por cultura y psicología se tiende a ver el espacio en profundidad que es lo que permite observar en estas interfaces, a través de una adecuada elección de los colores respetando la línea institucional. La duración de los elementos en pantalla se presenta de manera que la acción se desarrolla según el usuario, por lo que se está en presencia de un recorrido hipertextual.

Respecto a la tipografía utilizada, se puede observar que es la adecuada, se considera que el color se difumina en algunas circunstancias, por ejemplo en la elección de las materias o los docentes, donde debería conservarse el color gris casi negro sobre el blanco para favorecer la legibilidad de los textos citados.

En síntesis, en este capítulo se ha presentado la traza que se origina en los casos de uso, se realiza en los casos de prueba para el testing de funcionalidad y en las interfaces como medio único de comunicación con el usuario.

Si bien las interfaces GUI no pueden extrapolarse directamente a la Web, su funcionalidad y navegabilidad a través de los menús, constituyen una fuente de experiencia que puede reutilizarse. Los casos de uso y casos de prueba también pueden y deben reutilizarse, realizándoles las modificaciones necesarias por el agregado de funcionalidad o cambio de tecnología que se presentan en las especificaciones del sistema a migrar.

CAPÍTULO 7. HERRAMIENTAS DE GESTIÓN DE PRUEBAS

La arquitectura cliente/servidor representa un desafío significativo para la prueba del software. La naturaleza distribuida de este entorno, el rendimiento asociado al proceso de transacciones, la presencia potencial de diferentes plataformas de hardware, las complejidades de las comunicaciones en red, la necesidad de servir a múltiples clientes desde una base de datos centralizada y los requisitos de coordinación impuestos al servidor se combinan para hacer las pruebas de la arquitectura C/S considerablemente más complejas que la prueba de aplicaciones individuales.

Las herramientas automáticas de prueba están en proceso de experimentación. Existen varias categorías:

- Analizadores estáticos
- Auditores de código
- Procesadores de asertos
- Generadores de archivos de prueba
- Verificadores de prueba
- Controladores de prueba
- Comprobadores de salida

Todas las herramientas destinadas a analizar el código fuente y extraer información son llamadas *analizadores*. Hay dos categorías de analizadores para las aplicaciones web:

- Analizador HTML, que analiza fragmentos de código para obtener información relevante del HTML, como es el caso de los hiperenlaces.
- Analizador de Scripts, que analiza los scripts embebidos en el código para obtener información relevante, tal como el flujo de la aplicación, variables y sus valores.

Definidos los analizadores, corresponde introducir una referencia a los *asertos*. Un aserto es una sentencia que permite probar ciertas suposiciones sobre el programa, por ejemplo, si se escribe un método para calcular la velocidad de una partícula, se debería hacer un aserto para verificar que la velocidad calculada es menor que la de la luz. La utilidad de los asertos está probada por la experiencia. Ésta muestra que escribir asertos mientras se programa es una de las maneras más efectivas y rápidas de detectar y corregir errores. Además, un beneficio añadido es que los asertos sirven para documentar los procesos internos del programa, mejorando su mantenibilidad.

A los fines de la utilización de los asertos como precondiciones simplemente, se debería establecer la condición diseñada para la precondición como *Expresión1* de una sentencia *assert* y la sentencia *assert* se situaría al comienzo del método.

7.1. Pruebas de unidad

Una *prueba unitaria***,** como se vio en el capítulo 3, es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las *pruebas de Integración*, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

En el capítulo precedente, se desarrolló la escritura de casos de prueba para cada función no trivial o *método* en el módulo del sistema, de forma que cada caso fuese independiente del resto. Para que una prueba unitaria sea *buena* se deben cumplir los siguientes requisitos:

- *Automatizable*: no debería requerirse una intervención manual.
- *Completa*: debe cubrir la mayor cantidad de código.
- *Repetible o reutilizable*: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para *integración continua*.
- *Independiente*: la ejecución de una prueba no debe afectar a la ejecución de otra.
- *Profesional*: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

7.1.1. Ciclo de pruebas

Tal como se puede observar en la figura 52, el desarrollo del sistema proporciona la información suficiente para realizar el ciclo de pruebas. En el proceso de planificación de las pruebas de software, según el estándar IEEE829, se toma la mayor cantidad de información posible del proyecto y del software, con lo cual se elabora un plan de pruebas y con esa información se logra obtener un modelo de prueba. Mediante la generación de casos y procedimientos se conforma el software de prueba que genera, a su vez, entradas para la ejecución de las pruebas con el software a probar.

Posteriormente se realiza la evaluación de acuerdo a los resultados, verificando si los resultados obtenidos corresponden con los requerimientos del cliente. La depuración permite la localización y corrección de defectos. El análisis de la estadística de errores es útil a los fines de realizar predicciones de la fiabilidad del software y detectar las causas más habituales de error, y, por tanto, mejorar los procesos de desarrollo.

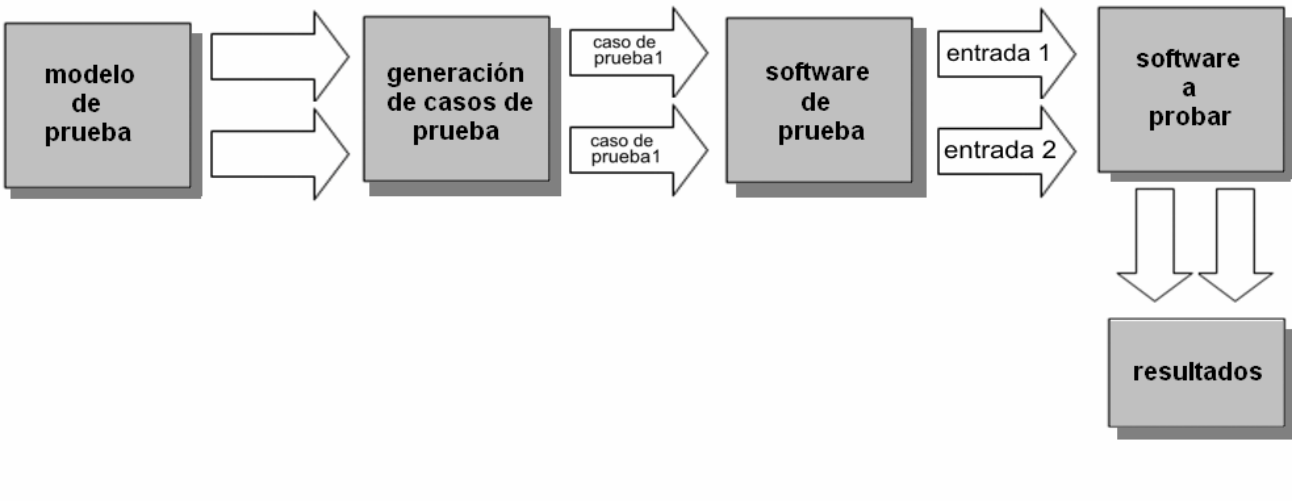


Figura 52 – Modelo de pruebas unitarias automatizada

7.1.2. Herramientas de gestión de pruebas

Se presenta a continuación una herramienta de gestión de pruebas desarrollada a medida para la generación y ejecución de casos de prueba a partir de los requisitos funcionales traducidos en casos de uso. En la figura 53 se presenta la primer pantalla de la herramienta en la que se observa la secuencia de casos de uso/prueba derivados de los casos de uso obtenidos del modelo de requerimientos, establecido en base al modelo de casos de uso planteado en el capítulo 6.

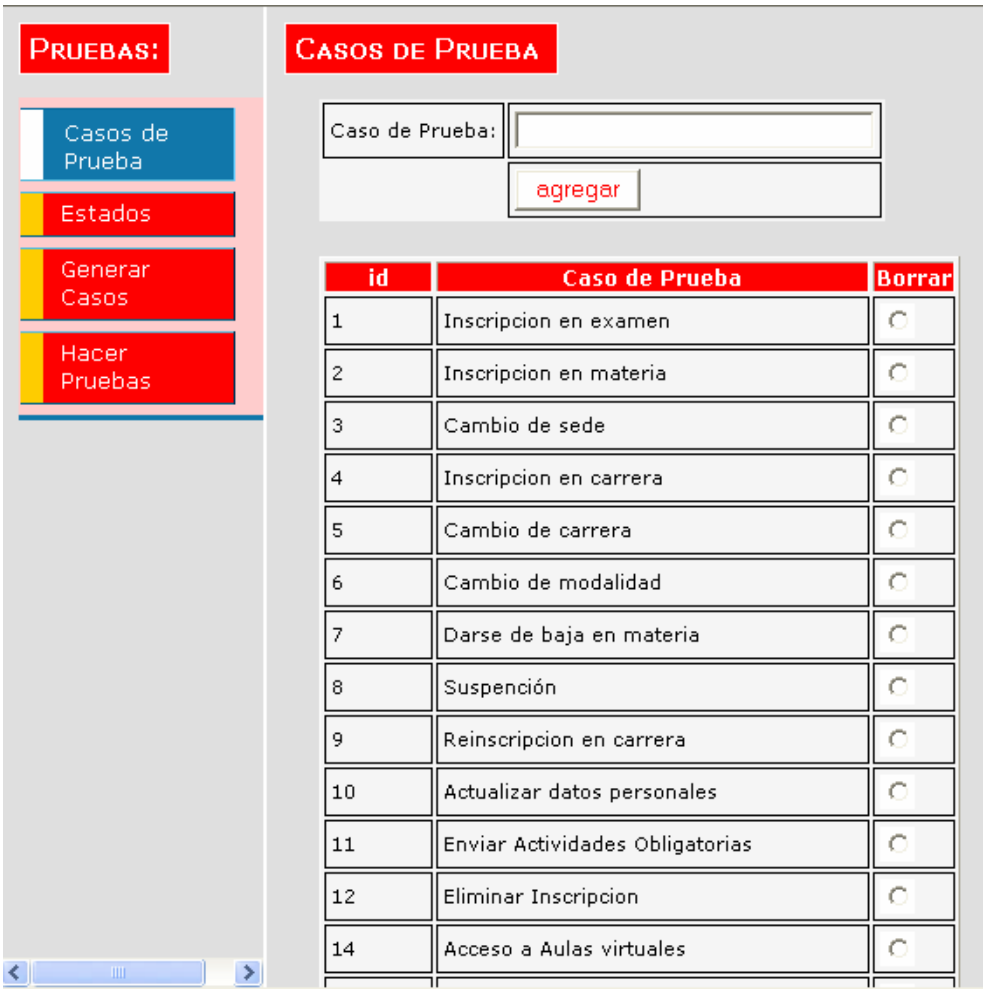


Figura 53 – Pantalla general de generación de casos de prueba

En la figura 54, se observa la siguiente pantalla del sistema, en la que se determinan los estados reconocidos o condiciones para cada caso de uso determinado en la pantalla anterior. Cada condición se obtiene utilizando el lenguaje de consulta a la base de datos de prueba.

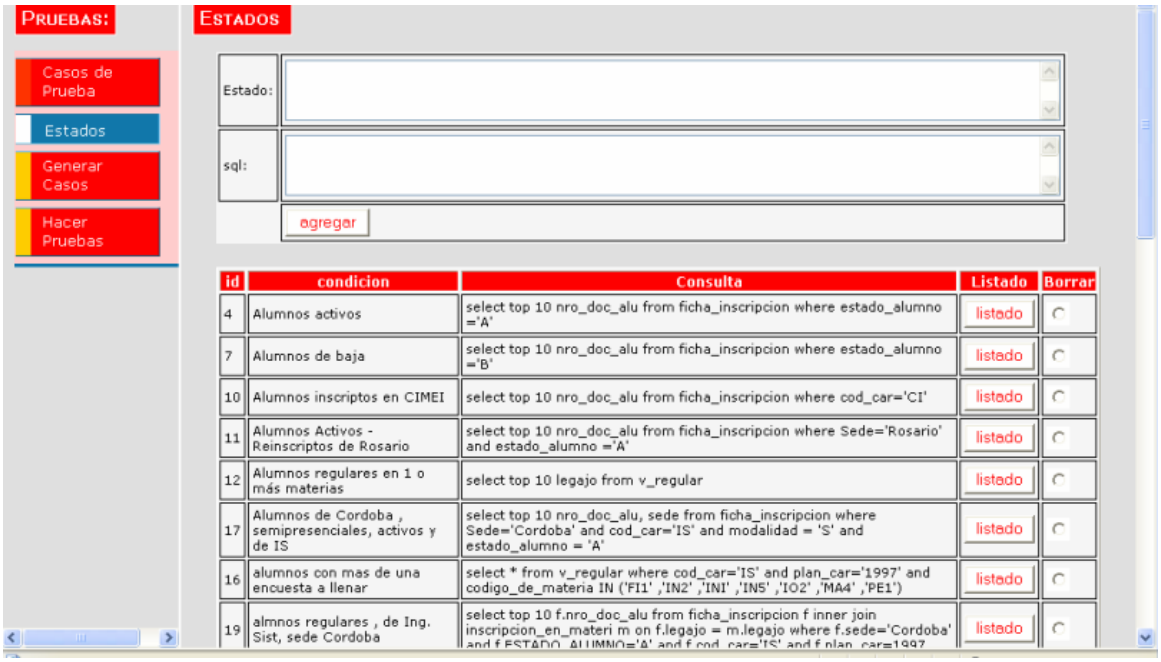


Figura 54 – Pantalla de generación de condiciones de prueba

En la figura 55, puede observarse la generación de casos de prueba en función de la selección de casos de uso con las condiciones o estados generados en la etapa anterior.

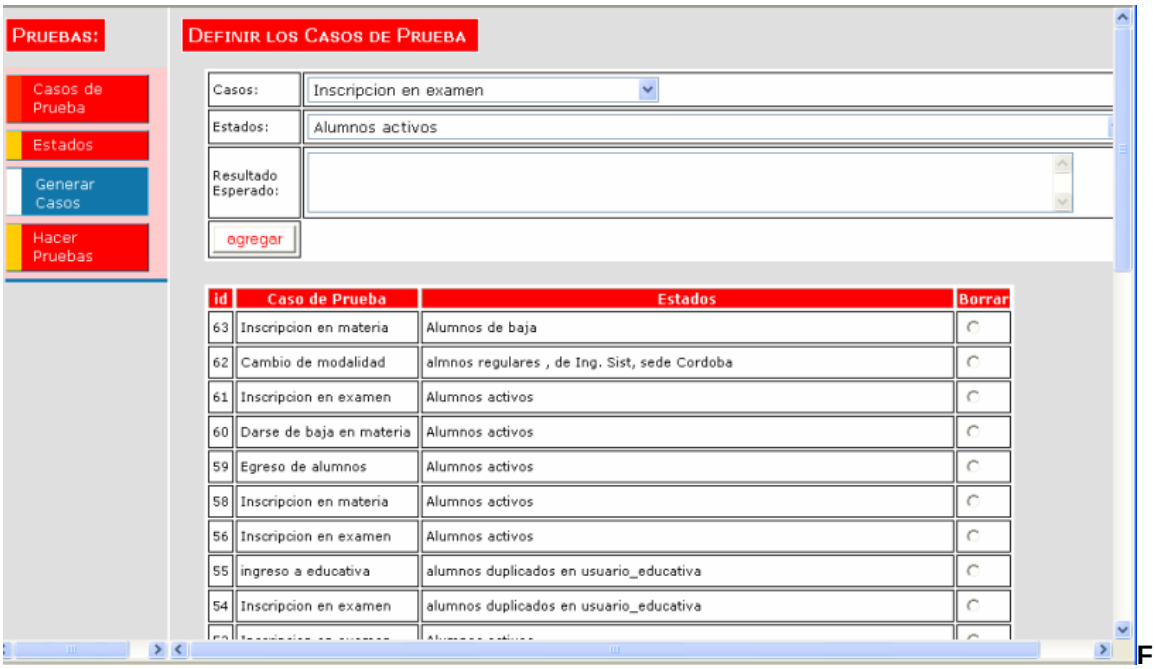


Figura 55 – Pantalla de generación de casos de prueba según condiciones de prueba

En la figura 56, se seleccionan los casos de prueba obtenidos en la etapa anterior y se generan, utilizándose el lenguaje de consultas, las entradas o inputs a la aplicación a probar.

Puede observarse en la figura un ejemplo del listado de inputs obtenidos; los cuales sirven de entrada a la aplicación de autogestión que se encuentra en la capa de lógica del negocio.

Figura 56 – Pantalla de generación de datos de prueba

Con los inputs obtenidos se realiza la prueba unitaria, para lo cual se invoca al procedimiento almacenado correspondiente, y se registran los resultados obtenidos de la ejecución, como error o esperados, para lo cual se complementa con las observaciones pertinentes, tal como se observa en la figura 57.

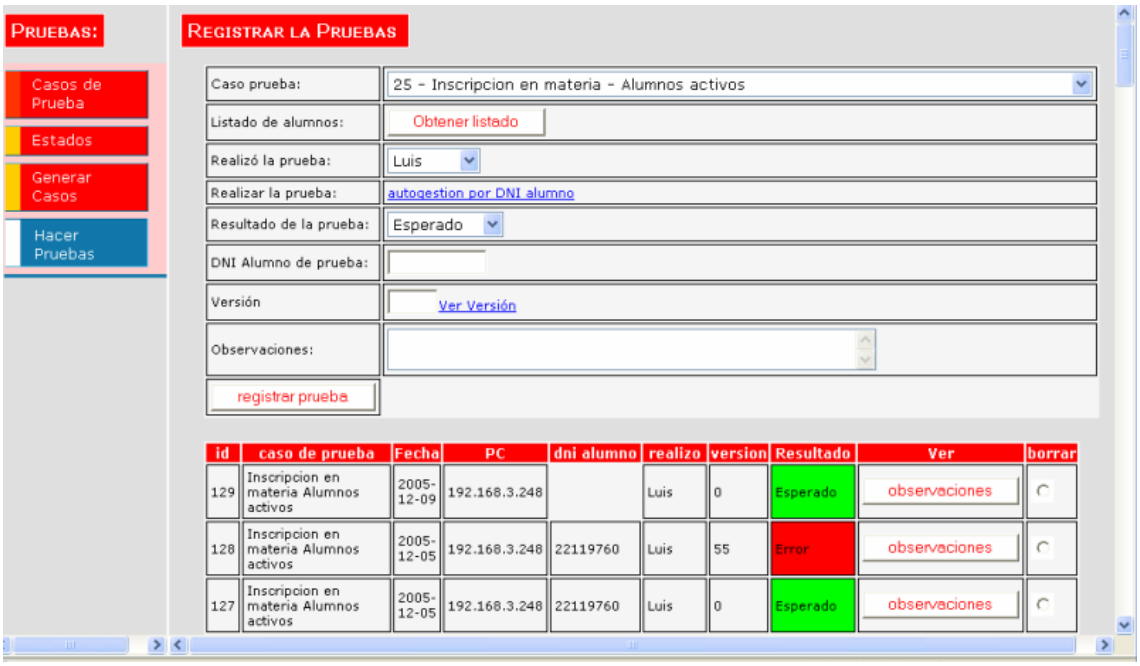


Figura 57– Pantalla de display de resultados de prueba

En la figura 58, se muestra cómo fue diseñado el modelo de la prueba de la Aplicación GUI que puede ser reutilizado para obtener los casos de prueba para la aplicación Web.

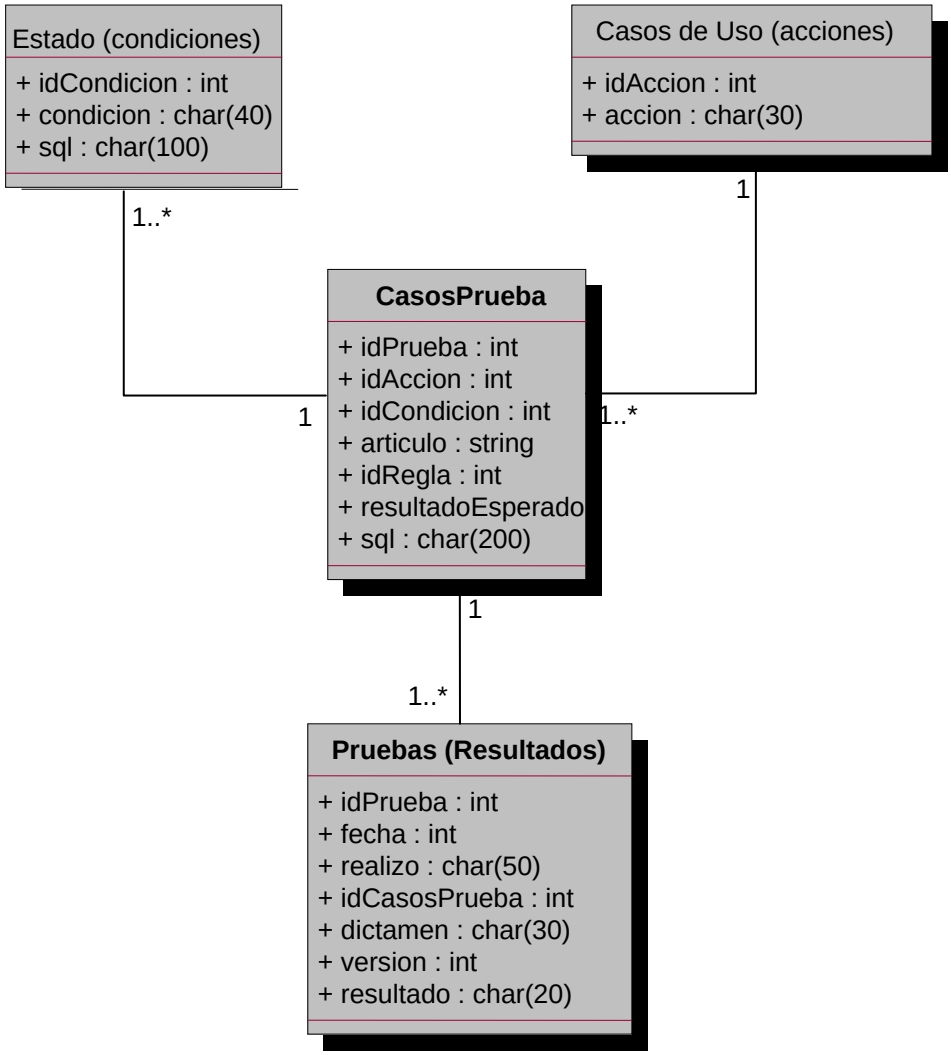


Figura 58 - Modelo conceptual de pruebas unitarias

7.2. Pruebas de integración

Tal como se definió en el capítulo 3, esta prueba se realiza una vez finalizados los ciclos de las pruebas unitarias, a los efectos de comprobar la correcta funcionalidad del sistema en su construcción general. A partir de los inputs de pruebas generados con la aplicación de pruebas unitarias, se utilizó el Robot de Rational para estudiar el comportamiento del sistema en ambas versiones.

Este robot es una herramienta de propósito general para la automatización del test para equipos de QA que realizan el testing funcional de aplicaciones cliente/servidor.

Al ejecutarse el robot, se observó que en el caso de aplicaciones Web los scripts que se van generando al grabar la prueba presentan algunas dificultades al encontrar componentes flash o java scripts; por lo tanto, se recomienda el uso de esta herramienta para la prueba conjunta de pocos casos de prueba simultáneos.

7.2.1. Características y beneficios del uso del robot para el testing

Este robot se considera una herramienta versátil de pruebas de configuración, regresión y funcionales para entornos en los que se desarrollan aplicaciones utilizado más de un IDE y/o lenguaje de programación; en nuestro caso, Power Builder en el sistema anterior y PHP o ASP en el sistema Web. Sin embargo, debe considerarse la mejora y facilidad de uso en sistemas bajo Windows que en otros entornos operativos.

Una característica importante del robot es que facilita la transición de los equipos de pruebas manuales a pruebas automatizadas. La realización de pruebas de regresión con *un robot* es un paso adelante hacia la automatización, ya que resulta fácil de utilizar y ayuda al equipo de pruebas a aprender los procesos de automatización mientras se realiza el desarrollo de los mismos. Entre las ventajas observadas durante su utilización, pueden mencionarse las siguientes:

- El robot registra automáticamente los resultados del test en un repositorio integrado, a través de un test log, coloreando el código para un rápido análisis visual. Mediante un doble click del mouse sobre una entrada, se posiciona directamente sobre la línea correspondiente del test script, asegurando, en consecuencia, un rápido análisis y corrección de los errores del test script.
- Permite a los expertos en automatización de pruebas identificar más defectos al ampliar sus scripts de pruebas con lógica condicional, para abarcar una mayor parte de la aplicación y para definir casos de prueba con llamadas a bibliotecas DLL o ejecutables externos.
- Proporciona casos de prueba para objetos comunes, como menús, listas y mapas de bits y casos de prueba especializados para los objetos específicos del entorno de desarrollo.
- Incluye herramientas para administración de test y está integrado a las soluciones propuestas en el Proceso Unificado de Desarrollo de Software para seguimiento de defectos, administración de cambios y trazabilidad de requerimientos.
- Se usa para testear aplicaciones basadas en una gran variedad de tecnologías de interfaz de usuario y está integrado a una solución Test Manager que se ocupa de proporcionar soporte para administrar todas las actividades de testing.
- Puede usarse para distribuir el testing funcional entre varias máquinas, cada una de ellas configurada en forma diferente. Se pueden correr simultáneamente los mismos tests funcionales, reduciendo el tiempo necesario para identificar los problemas con configuraciones específicas.
- Testea más allá de la interfaz de usuario de la aplicación los cientos de propiedades de los objetos componentes de la misma –tales como Controles ActiveX, OCXs, applets Java y muchos otros– con solo un click del mouse.
- Testea los controles y objetos usuales, pues permite testear cada componente de la aplicación bajo variadas condiciones proveyendo test cases para menús, listas, caracteres alfanuméricos, bitmaps y muchos otros objetos.
- Proporciona un ambiente integrado de programación. Genera test scripts en SQABasic, un ambiente MDI integrado de scripting que permite ver y editar el test script mientras se lo está grabando.
- Facilita el reuso; puesto que permite que los mismos test scripts, sin ninguna modificación, se puedan reusar para testear una aplicación que corre sobre Microsoft Windows XP, Windows, ME, Windows 2000, Windows 98 o Windows NT.

- A partir de inputs generados en las pruebas unitarias, se realizó una prueba de integración, utilizando este robot en ambas aplicaciones. En la figura 59, se presenta el script generado para el caso de prueba “Ingresar al Sistema de autogestión” en la aplicación GUI, y en la figura 60 el script para el mismo caso de prueba en la aplicación Web.

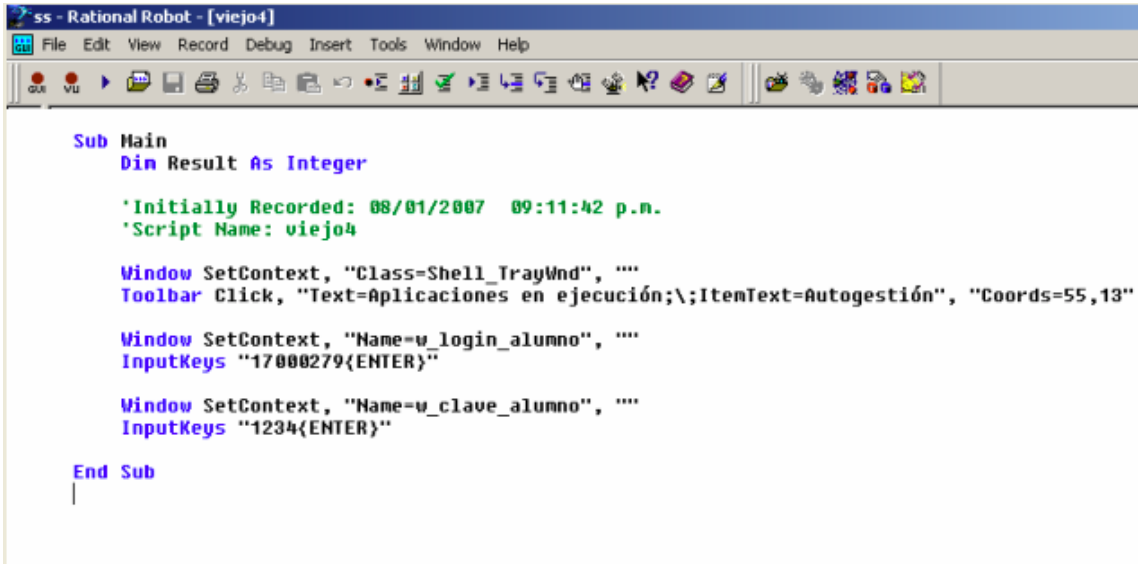


Figura 59 – Test script del sistema de autogestión en la aplicación GUI

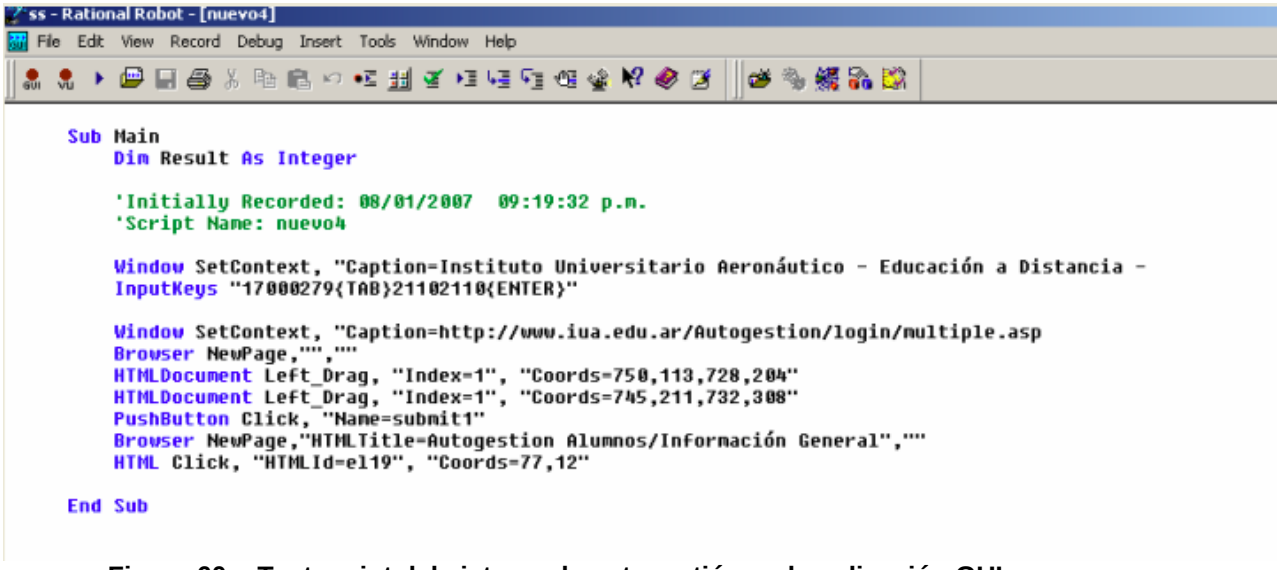


Figura 60 - Test script del sistema de autogestión en la aplicación GUI

Tal como se puede observar en las figuras anteriores, los scripts generados en SQABasic están disponibles para su edición, ya que no presentan dificultades para el análisis de su código. Al ejecutarse los test scripts presentados anteriormente, el robot genera los test logs de las pruebas de ambos sistemas, con sus detalles operativos, como puede visualizarse en las figuras 61 y 62 correspondientes al script del sistema anterior y en la figura 63 el correspondiente a la ejecución del script de la aplicación Web.

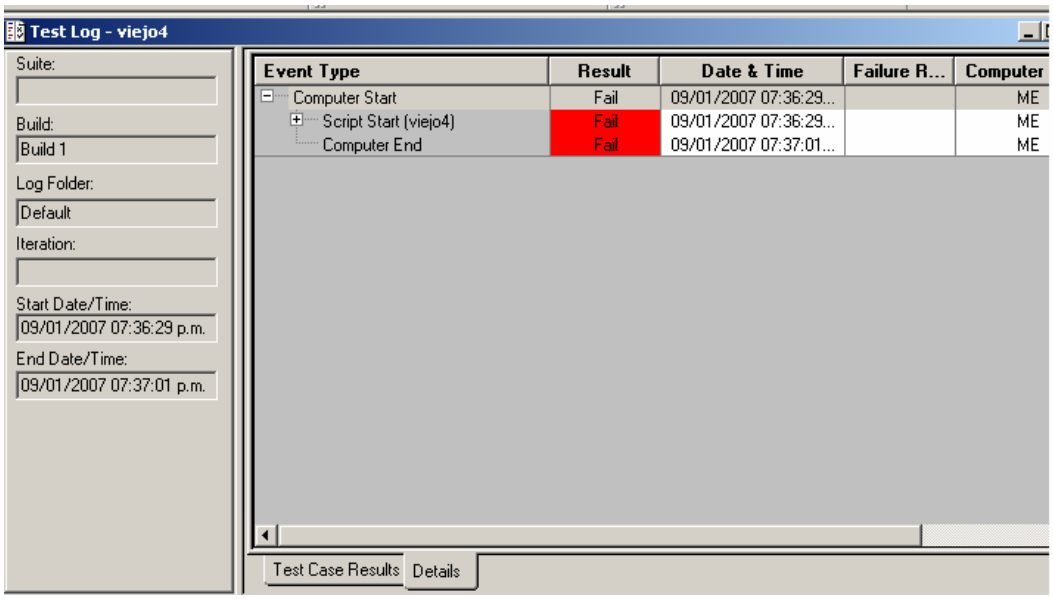


Figura 61 - Test log del sistema de autogestión en la aplicación GUI

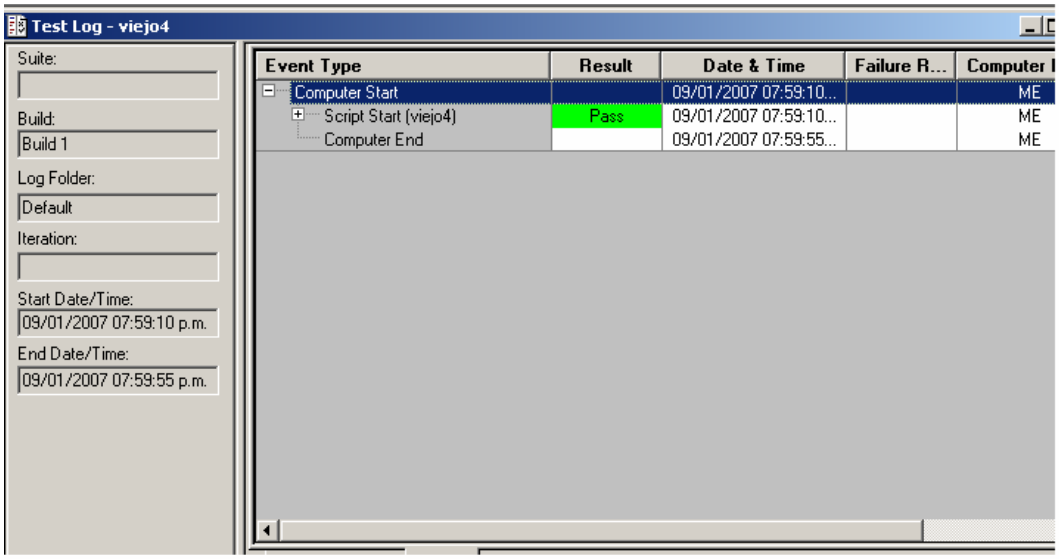


Figura 62 - Test log del sistema de autogestión en la aplicación GUI

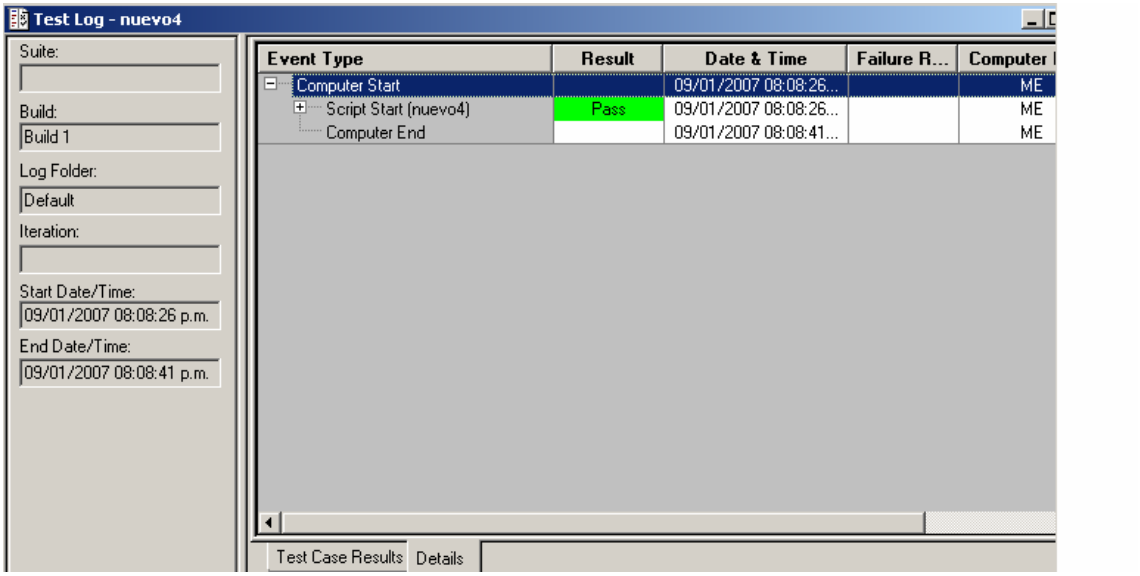


Figura 63 - Test log del sistema de autogestión en la aplicación Web

En forma similar a nuestra aplicación que generaba y realizaba las pruebas unitarias, se pueden observar los resultados de las pruebas en ambos sistemas.

7.2.2. Uso del robot para pruebas de regresión

Una funcionalidad de la herramienta que se utiliza para evitar errores al completar toda la prueba de integración es que se pueden insertar los llamados *verification points* o *puntos de verificación* para testear si el resultado de una acción se ha realizado, o no, correctamente.

Como los scripts se guardan por proyecto en forma automática, y el robot permite crear nuevas iteraciones para cada ciclo de prueba, se propone construir para cambio de versión una iteración diferente y reusar de esa forma los scripts generados en la versión anterior, utilizando a tal efecto las planillas con los casos de prueba realizados y los resultados esperados. De tal manera, gran parte de los casos de prueba del sistema anterior pueden adaptarse a los cambios de software que se producen, con lo que es posible realizarlos con esta herramienta pruebas de regresión.

7.2.3. Herramienta de chequeo de enlaces Web

Otra herramienta utilizada fue W3C Link Checker que permite chequear la validez de los enlaces de una determinada página Web. Es posible definirle un grado de recursión, es decir, que compruebe la validez de los enlaces hasta un determinado nivel de profundidad. El W3C propone en la actualidad un conjunto de tecnologías integradas que permiten desarrollar un sitio Web completamente multimedia con las existentes tecnologías interoperables del W3C utilizando XHTML (Marcado XML estructurado), CSS (Hojas de estilo), SVG (Gráficos vectoriales animados en 2D), y SMIL (Multimedia Sincronizada). Estas tecnologías han contado con el consenso de los diferentes actores del mercado Web.

La Web es un lugar libre compartido por muchos usuarios cuyas necesidades no necesariamente se conocen. Los estándares han sido diseñados para tener en mente a todas las audiencias potenciales. Es un reto a la comunidad Web el crear con estándares Web. Evita estar ligado a alguna compañía o tecnología propietaria. Se pueden usar tecnologías que son independientes de los requerimientos de las plataformas; y además

diseñar con estándares simplifica el mantenimiento del código del sitio Web porque evita tener múltiples versiones para diferentes navegadores. A continuación se presenta el resultado de la ejecución del W3C link checker sobre el sitio web del IUA.

Results

Anchors

Found 17 anchors.

Valid anchors!

List of broken links and redirects

Fragments listed are broken. See the table below to know what action to take.

Code	Occurrences	What to do
501	1	Could not check this link: method not implemented or scheme not supported.

javascript:MasInfo();

What to do: **You must change this link: people using a browser without JavaScript support will *not* be able to follow this link. See the Web Content Accessibility Guidelines on the use of scripting on the Web and the techniques on how to solve this.**
Response status code: 501
Response message: Protocol scheme 'javascript' is not supported
Line: 149 **(1)**

List of directory redirects

The links below are not broken, but the document does not use the exact URL.

http://www.iua.edu.ar/conocer redirected to
http://www.iua.edu.ar/conocer/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 80, 430

http://www.iua.edu.ar/investigacion redirected to
http://www.iua.edu.ar/investigacion/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 83, 433

http://www.iua.edu.ar/contactenos redirected to
http://www.iua.edu.ar/contactenos/ **(2)**

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 85, 435

http://www.iua.edu.ar/vida redirected to
http://www.iua.edu.ar/vida/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 82, 432

http://www.iua.edu.ar/estudiar redirected to
http://www.iua.edu.ar/estudiar/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 81, 431

http://www.iua.edu.ar/servicios redirected to
http://www.iua.edu.ar/servicios/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 84, 434

http://www.iua.edu.ar/cread2004 redirected to
http://www.iua.edu.ar/cread2004/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Line: 94

Checked 1 document in 79.4 seconds.

Notas aclaratorias sobre el resultado del rastreo sobre el sitio web

En HTML, la etiqueta utilizada para crear un vínculo es **a** (*anchor*).

Un enlace debe apuntar a una página web, a una imagen, a un servidor FTP, a cualquier otro tipo de archivo, a un correo electrónico o incluso puede apuntar hacia otros sectores de la misma página (previamente marcando estos sectores).

En **(1)** el enlace apunta a un método escrito en javascript denominado MasInfo(), el cual no será encontrado por los browsers en máquinas que no soporten JavaScript.

En **(2)** debe agregarse la barra " / " para escribir correctamente el enlace, por lo que si bien se reconoce el link, éste es redirigido corrigiéndose la referencia.

En el Anexo IV se presenta el resultado total de la ejecución del W3C link checker sobre el sitio Web del IUA.

CAPÍTULO 8. TRABAJOS RELACIONADOS

8.1. Introducción

En los anteriores Capítulos 5, 6 y 7 se aplicó la metodología presentada en este trabajo a un Caso de Estudio y se hizo un análisis comparativo empleando esta propuesta en una selección de casos de prueba del sistema original y del posterior sistema ya migrado. También se presentaron las herramientas de gestión de pruebas utilizadas en las diferentes etapas, brindando ejemplos sobre su aplicación a los sistemas estudiados. De esta manera se cubrieron los objetivos propuestos para este trabajo de tesis. No obstante, y llegado a este punto, parece conveniente revisar algunas de las recomendaciones y propuestas que más influyeron en el desarrollo de este trabajo. Estas están referidas a la migración de sistemas a la Web y varias de las cuales fueron ya mencionadas a lo largo de este trabajo, en especial en los Capítulos 2 y 3.

Con este fin, para cada uno de los principales aspectos de la migración de Sistemas a la Web se comentan las principales propuestas y su influencia o vinculación con el trabajo realizado.

8.2. Migración de sistemas heredados

Bisbal et al (1997) plantean la necesidad de una comprensión exhaustiva del sistema a migrar entre las que cita: los datos, las interfaces y las aplicaciones. En cuanto a los datos propone una instancia de mapeo de los mismos para reconocer las transformaciones requeridas antes del proceso de migración propiamente dicho, y la incorporación de la nueva funcionalidad después que el sistema migrado haya sido testeado y verificada su consistencia con el sistema original.

Brodie y Stonebraker (1995) proponen una aproximación denominada “Chicken little” basada en un módulo que permite que ambos sistemas puedan interoperar durante la migración, replicando los datos en ambas bases, lo que constituye un gran desafío para mantener la consistencia e integridad de los mismos.

Por otro lado, la metodología de migración Butterfly propuesta por Wu et al. (1997) asume que mientras el sistema a migrar deba permanecer operativo, no es necesario que ambos interoperen simultáneamente y pone énfasis en la migración de los datos en un entorno crítico separado completamente del desarrollo del nuevo sistema migrado.

Liem et al. (2006) presentan una experiencia de migración de un sistema de información académico donde la nueva aplicación ha sido totalmente rediseñada y codificada, resguardando la consistencia de los datos por la índole de la información a migrar. El sistema anterior no contaba con documentación técnica ni de procedimientos. La solución fue desarrollar versiones incrementales del nuevo sistema que se mantuvieron activas durante un tiempo considerable conviviendo con el sistema anterior, facilitando la migración progresiva de los datos y la capacitación de los usuarios.

Como puede observarse, en general todas las propuestas contemplan la necesidad de una metodología de desarrollo en donde se hace foco en el conocimiento del sistema a migrar. Difieren en la oportunidad de la puesta a régimen de ambos sistemas y en la

conversión de los datos a migrar. La mayoría de los casos presentados no contaba con documentación anterior del sistema.

En coincidencia, y teniendo en cuenta que el enfoque propuesto para este trabajo es el testing de migración, todas las "lecciones aprendidas" sirvieron de base para elaborar una propuesta metodológica "a medida" para la migración, a los fines de obtener un conocimiento profundo del sistema a migrar, tal como se presenta en el capítulo 2.

8.3. Testing en la web

Desphande, Murugesan et al. (2002) presentan en un trabajo conjunto una serie de interesantes reflexiones sobre la ingeniería Web, su naturaleza evolutiva y compleja y las características especiales del desarrollo, testeo y mantenimiento de aplicaciones Web. Sintetizan las mayores diferencias entre las aplicaciones Web y las convencionales a través del análisis de numerosas contribuciones realizadas por varios autores sobre esta temática, y, establecen categorías evolutivas para clasificar las aplicaciones Web.

Una vez identificadas estas diferencias observan que estas nuevas tecnologías deben ser tratadas por fuera de las metodologías tradicionales, y con la participación de un equipo multidisciplinario dada la naturaleza multimedial de las mismas, donde se contemplen parámetros de calidad y seguridad. Citan las diferentes metodologías existentes, resaltando la aparición de aproximaciones evolutivas en la implementación de las mismas.

Respecto al testing en la web, plantean la multidimensionalidad respecto del testing tradicional puesto que cada unidad de la aplicación (página, código, navegación, standards, etc.) debe ser probado. Citan los servicios como W3C's, HTML, CSS, y XHTML como disponibles libremente para los desarrolladores.

Aquí es necesario destacar que el servicio W3C's fue utilizado en la aplicación del caso de estudio presentado en este trabajo para examinar los hipervínculos del sitio Web del IUA, tal como se muestra en el capítulo 7, y el servicio CCS fue propuesto y es utilizado por el personal de desarrollo a los fines de estandarizar los formularios que se utilizan en las distintas páginas del sitio. Por otro lado el equipo de desarrollo es multidisciplinario ya que además de analistas y programadores intervienen diseñadores Web en las actualizaciones del mismo.

Bidart et al (2002) presentan una investigación muy completa sobre los aspectos generales del Web testing tanto en su marco teórico como práctico. Plantean las dificultades que aparecen en el testing para la Web respecto del convencional y la necesidad de la comprensión previa de los requisitos del negocio antes de emprender la tarea del testeo. Realizan una clara clasificación de los tipos de testing que deben realizarse de acuerdo al flujo de trabajo de desarrollo, basado en el modelo en V, como se plantea en el capitulo 3, y proponen la necesidad de contar con herramientas automáticas para agilizar el testing y lograr la calidad en las aplicaciones de acuerdo a una serie de criterios específicos para aplicaciones Web.

Es necesario aquí reconocer que la lectura de esta propuesta fue clarificadora para el análisis del enfoque funcional del testing propuesto en el capítulo 4, y la selección de herramientas para su automatización.

Por su parte, Cáceres y Marcos (2003) proponen el uso de metodologías ágiles para el desarrollo de aplicaciones Web que buscan el equilibrio entre proceso y esfuerzo y,

orientadas al personal asignado al desarrollo, basadas en un modelo de proceso iterativo, incremental y adaptativo. Esto permite obtener versiones completas y estables del software con mejoras sucesivas.

Es evidente que los grupos de desarrollo deben proveer soluciones inmediatas ante los requerimientos de los sectores de la organización, por lo cual se confirma la necesidad que, dada la naturaleza de las aplicaciones Web, se pueden obtener versiones estables en forma iterativa cuya mejora está dada en muchos casos en la presentación de las mismas.

Minguens y García Morales (2003) plantean la necesidad del uso de una metodología para la web, basándose en las características distintivas de estas aplicaciones y realizan un estudio de la evolución de estas metodologías haciendo foco en la metodología Web basada en UML (UWE) propuesta por Koch (2002).

En concordancia con estas ideas, la propuesta que se presenta en esta tesis ha sido desarrollada con la utilización del Lenguaje Unificado de Modelado (UML).

Andreas Spillner (2002) presenta su modelo W donde se observa la importancia del testing en todo el ciclo de desarrollo del software, pero separado del mismo, por lo que propone comenzar las actividades de testing desde las fases tempranas del ciclo con actividades "constructivas" para resolver fallas y defectos encontrados, y, considerando los riesgos que aparecen en su ejecución.

Antonia Bertolino (2003) realiza un análisis pormenorizado de la investigación y práctica del testing de software y plantea la necesidad de establecer criterios para la selección de suites de casos de prueba, entre ellos, los basados en las especificaciones de los requerimientos formalizados utilizando UML. No deja de lado la ejecución de los test (en forma manual o automatizada), y el análisis de los resultados de la misma. Menciona también la importancia y el costo de la ejecución de los tests de regresión, para los cuales hace necesaria la selección de subconjuntos de casos de prueba, a los efectos de ahorro de tiempo y esfuerzos.

Mary Harrold (2002) presenta propuestas para reducir costos en la ejecución de los test de regresión, ya que considera que los mismos representan la tercera parte del costo total del sistema. La eficiencia en estos tests se basa en el desarrollo de técnicas para identificar casos de prueba que puedan ser usados para el retesteo del software. Afirma que un test suite que valida requerimientos individuales puede ser más eficiente para usar en los test de regresión que aquellos que validan muchos requerimientos simultáneamente. Otro de sus aportes se basa en la selección de criterios para revelar tipos de fallas, por lo que una combinación de estos criterios puede lograr una mejora en el proceso de testing.

Otro aspecto importante descripto por Beizer (1995) es la construcción de un plan de testing durante la fase de recolección de requerimientos y su implementación a posteriori de la implementación del software. Un aporte interesante es el reguardo de los artefactos producidos por el testing, ya que las trazas de ejecución posibilitan distinguir aquellos casos de prueba que pasaron sin errores y los que fallaron para poder identificar el origen de las fallas potenciales del software.

Estas últimas propuestas fueron especialmente tenidas en cuenta en este trabajo al ponerse el foco en la generación de casos de prueba con condiciones y resultados esperados que se equiparan con los resultados obtenidos y se displayan los mensajes correspondientes que evidencian la corrección o falla del sistema bajo test., según puede observarse en el capítulo 7.

8.4. Reutilización de requisitos

López y Laguna (2001) plantean un enfoque de reutilización de requisitos para el éxito de un desarrollo con reutilización. Estos requisitos pueden utilizarse tanto para el mantenimiento de aplicaciones existentes como para el desarrollo de nuevas especificaciones. Establecen los modelos de dominio dentro de un área funcional diferenciable cuyo conocimiento se traduce en una serie de requisitos funcionales y no funcionales, aunque los primeros tienen un rol fundamental en la reutilización. No obstante, la complejidad de los dominios obliga a dividir el contexto en subdominios y establecer un repositorio para cada dominio y sus subdominios derivados. Para lograr la reutilización interdominios se propone un proceso de abstracción que define dominios genéricos que presenta actividades, conceptos y reglas genéricas, así como subdominios genéricos. Estos conceptos de representación de dominios se aplicaron en modelos de mecanos.

Escalona y Koch (2002) ofrecen una visión general y un estudio comparativo del tratamiento de requisitos en las actuales metodologías para la Web. Realizan una síntesis de las principales técnicas para la captura, definición y validación de requisitos. Analizando los resultados, y teniendo en cuenta que las propuestas están ordenadas por orden cronológico, resulta interesante observar el avance que han tenido los requisitos en el entorno de la web. Las primeras propuestas estaban más centradas en los requisitos de datos y los de interfaz de usuario, mientras que las propuestas más actuales resaltan la necesidad de tratar los requisitos de personalización y navegación, así como los transaccionales de forma independiente.

Según estos autores, si se compara este trabajo con otros estudios comparativos de propuestas de la Web (Koch, 1999, Barry & Lang, 2001 y Escalona, Mejías & Torres, 2002), se pueden observar la gran mayoría de las metodologías que existen son centradas en el diseño de sistemas Web en comparación con las que contemplan la especificación de requisitos.

En el capítulo 4, en la metodología propuesta para el análisis lógico y físico del sistema a migrar, se propone la reutilización de requisitos en el proceso de migración a la Web. Esta idea se materializa en el caso de estudio que se presenta en el capítulo 5, donde se aplica la división de los requisitos en dominios y subdominios a los fines de establecer, mediante una comparación de los modelos de requisitos del sistema anterior y el migrado, la obtención de los nuevos requisitos del nuevo sistema.

8.5. Generación de casos de prueba

Fernández Sanz et al. (2004) proponen un algoritmo para la generación automática de casos de prueba a partir de especificaciones UML y una serie de prácticas orientadas a la ordenación de dichos caos de prueba en función del riesgo asociado a cada uno de ellos. Plantean que a la hora de probar un software se hace necesario preguntarse: sobre las vulnerabilidades del producto, la amenazas que pudieran provocar un fallo en el sistema y las víctimas que pudieran verse afectadas por un fallo y cuáles serían sus consecuencias. Afirman que se debe hablar necesariamente de especificación de requisitos, casos de uso y caos de prueba, constituyéndose los casos de uso como autoridad de prueba para realizar la verificación del sistema.

El proceso para generación de casos de prueba se compone de: una fase de análisis que obtiene los diferentes casos de uso desde la perspectiva del usuario, una

fase de diseño donde se produce una matriz de trazabilidad entre casos de uso, clases y casos de prueba, establecer un ranking de casos de prueba ordenándolos en función del nivel de riesgo de los casos de uso asociados a ellos y, ciclar en la fase de ejecución de las pruebas tantas veces como sea necesario para conseguir el nivel deseado de defectos detectados. La propuesta se completa en la adopción de un criterio de selección de casos de prueba basado en los distintos escenarios de ejecución de caso de uso, cuya primera aproximación se basa en la selección de caminos trazados en los diagramas de actividades. Otro criterio de cobertura de pruebas se basa en Myers (1984) que dice: "Para las entradas de datos en un caso de uso se debe lograr que todas las clase de equivalencia sean ejercitadas por un caso de prueba, logrando que se ejerciten todos los valores límite".

Por su parte, Diego Vallespir (2004) centra su propuesta en la generación automática de casos de prueba unitarias para sistemas orientados a objeto, haciendo foco en las pruebas funcionales o de caja negra. Basado en la propuesta de Mc Gregor y Sykes (2001), obtiene un conjunto de reglas de derivación de los casos de prueba de los métodos, a partir de las pre y poscondiciones expresadas en OCL, y con el diagrama de estados de clase expresado con UML.

Rosaria y Robinson (2001) describen una propuesta de testing basada en la creación de un modelo que describe el comportamiento del sistema bajo testeo y la generación de casos de prueba usando algoritmos gráficos, para lo cual se utiliza un programa manejador del testing que toma las entradas o inputs de los algoritmos, determina qué acción seguir e invoca a la función que la debe ejecutar. Luego se verifica si la acción tomada reacciona correctamente a la entrada aplicada.

Esta última propuesta inspiró la implementación del software aplicado al caso de estudio de Autogestión de Alumnos explicitado en el capítulo 7 y que se encuentra en régimen en la actualidad.

8.6. Diseño y prueba de interfaces gráficas de usuario (GUI's)

Pinheiro da Silva y Paton (2000) hacen una pormenorizada descripción del modelado de interfaces gráficas de usuario mediante el uso de UML. Para ello se basan en la construcción de casos de uso para identificar actores y acciones que permitan proveer el conocimiento necesario para el diseño de la interfaz de usuario. Como los casos de uso no proveen la suficiente información de control éstos se acompañan con los diagramas de actividades a los fines de reconocer las pre y poscondiciones asociadas a los requerimientos del usuario. Acompañan a esta propuesta la presentación de modelos abstractos y concretos representados a través de diagramas de clases.

De esta fuente se rescató la confección de los diagramas de casos de uso del sistema a migrar y el migrado, cuya comparación permitió abordar el reuso de funcionalidades. Como la aplicación a migrar fue desarrollada con metodología estructurada, se utilizó la técnica de escenarios para complementar los requisitos para el modelado de las interfaces.

Linz y Daigl (1997/1998) plantean la automatización del testing de las GUI's con el uso de una herramienta de captura (Capture and Replay Tool) cuyas funciones operativas son las siguientes: Modo de captura a través de la interacción manual del usuario con la aplicación, programación realizada por la herramienta de la captura a través de la generación de los test scripts, checkpoints que pueden ser insertados por el

usuario y, la posibilidad de repetir los tests scripts para verificar las posibles fallas en su ejecución, las que son acusadas por la herramienta. Proponen además la generación de una librería de test scripts que completen las especificaciones de la plantilla de prueba de la interface. Por cada test script se presenta una especificación de su utilización.

En este trabajo de tesis se utilizó como herramienta de testeo de interface el Robot de Rational que cumple los requisitos especificados en el planteo de Linz y cuyos resultados se presentan en el capítulo 7.

Por último cabe considerar la propuesta de Memon y Soffa (2001) sobre una técnica de testing de regresión de GUI's que determina los casos de prueba que pueden reusarse o desecharse y repararse después de una modificación de la GUI. Para su implementación usan la técnica de representación de las interfaces a través de árboles y de lenguaje de control OCL. Para la selección de los casos de prueba tienen en cuenta aquéllos que sirven para probar interfaces que se han modificado y reparar los que pueden ser reutilizados.

Basado es estas ideas es que se propone en las conclusiones y trabajos futuros el reuso de casos de prueba para el testing de regresión, teniendo en cuenta la continua evolución de los sitios Web.

CONCLUSIONES Y TRABAJO FUTURO

En la actualidad, la migración a la Web de sistemas de arquitectura cliente servidor es una necesidad imperiosa para toda organización; no sólo a los fines comerciales y competitivos, sino para favorecer las comunicaciones y fundamentalmente la "igualdad de oportunidades" independientemente de la distancia y condición socio económica de las personas.

En este trabajo se presentó una metodología para el análisis lógico y físico de las aplicaciones distribuidas a ser migradas a entornos Web que pone especial atención en el testing de migración. El enfoque propuesto se basa en la reutilización de casos de pruebas existentes, resultantes de la trazabilidad con los casos de uso planteados para el sistema distribuido, y la automatización de las pruebas unitarias, de integración y de regresión.

Esta propuesta se materializó en base a un sistema ya migrado, del que se reconstruyeron todas las etapas previas al testing de acuerdo a la metodología aplicada en su momento, poniendo el foco en los requisitos como base para las pruebas. Es por ello que este trabajo se basó en el testing funcional, a lo que se propone incorporar luego otras técnicas como revisiones de documentación, de código y aplicar otros tipos de testing como los de stress y volumen.

La estrategia de prueba aplicada fue la de caja negra, por lo que se utilizó un subconjunto de casos de uso fundamentales y se produjo su derivación en casos de prueba. Del análisis de los casos de prueba generados en el sistema primitivo se probó la posibilidad de realizar su rehúso en el sistema Web migrado, encontrándose modificaciones originadas en las posibilidades tecnológicas que ofrece la Web y en el agregado de funcionalidad debida a la evolución dinámica de los sistemas.

Respecto a las recomendaciones a tener en cuenta al migrar un proceso a la Web, desde el punto de vista del testing, cabe realizar algunas consideraciones:

- En el nuevo contexto los tiempos disponibles para llevar a cabo el proceso de testing son mucho menores. Se escuchó decir de un gerente de sistemas: "*Decidimos a la mañana, programamos a la tarde y probamos a la noche*".
- La volatilidad de los requerimientos es mucho mayor.
- El nivel de documentación es bajo y el cambio tecnológico es permanente.
- Los equipos de trabajo son más jóvenes, menos experimentados, a veces distribuidos geográficamente, multidisciplinarios y con menor respaldo tecnológico.

Con respecto a la *automatización del testing,* se utilizó una herramienta diseñada a medida para la generación de datos de prueba en función de los casos de prueba planteados en las plantillas presentadas. Con estas *entradas*, la aplicación hace las pruebas unitarias de los programas almacenados y ya modificados para el nuevo sistema. Esta aplicación genera un *test log* que identifica y guarda en un repositorio la versión de la aplicación utilizada y su resultado a los fines de ser consultada. Un aspecto a destacar de esta herramienta es la posibilidad de obtener estadísticas sobre los casos de prueba más invocados, que permitan generar un subconjunto importante para los futuros test de regresión y realizar seguimiento y documentación de las pruebas.

El manejo de errores también pudo documentarse automáticamente ya que la herramienta automática especifica el error encontrado en la prueba de los módulos, por lo que puede realizarse "post mortem" una estadística de los mismos a los efectos de medir la cantidad de defectos encontrados y reparados en las distintas versiones del producto software hasta llegar a su puesta en régimen.

Otro punto clave a ser considerado es la definición de casos de prueba durante el análisis del sistema. Esta práctica significa un importante ahorro de tiempo y es una forma de documentar la aplicación anticipándose al problema. En este estudio quedó demostrado que utilizando los casos de uso se podrían desarrollar los casos de prueba y luego reutilizar ambos para la migración, e incluso, para su rehúso en aplicaciones similares o en regresiones posteriores sobre el mismo sitio. Otra ventaja encontrada en el diseño temprano de los casos de prueba es la posibilidad de descubrir errores en la especificación de requerimientos por omisión o error en la definición de los mismos.

Respecto a la prueba de integración se aplicó el *Rational Robot* que tiene la ventaja de ser un software gratuito provisto para las universidades. Si bien resultó útil a los fines de las pruebas con las *entradas* generadas en la aplicación anterior, evidenció una mejor performance en la aplicación distribuida, ya que al ejecutar el robot en la aplicación Web se evidenciaron algunos conflictos con objetos flash, situación que fue reportada a la empresa proveedora.

De acuerdo a las pruebas realizadas, no corresponde el reuso de los scripts de la obtenidos de la aplicación tradicional a la de la Web, ya que sus interfaces son diferentes en su totalidad, pero sí resulta conveniente comparar los *test logs* que se obtienen de las ejecuciones con el robot en ambos sistemas para obtener un conjunto de casos de prueba congruentes entre ambas aplicaciones.

Debe tenerse en cuenta, además, la utilización de esta herramienta para los test de regresión dentro del mismo entorno y la posibilidad de realizar pruebas simultáneas utilizando distintos navegadores. Asimismo, resulta importante y queda como propuesta la *registración***,** a partir de los *test logs*, de los problemas encontrados en base a los resultados esperados planteados en los casos de prueba, para aprender de ellos, no volver a cometer los mismos errores y categorizarlos antes de su resolución.

En tanto, para chequear la estructura del sitio Web merece destacarse el *W3C link checker*, que resultó ser una herramienta de gran utilidad y que además ofrece opciones de estandarización para la creación de páginas Web tales como hojas de estilo, etc.

Los *checklist* realizados sobre las interfaces de la aplicación distribuida original pudieron ser totalmente reusados para la aplicación Web teniendo en cuenta las diferencias en la utilización del *browser*. Estos checklists constituyen una herramienta que puede ser utilizada no sólo por los testeadores sino también por desarrolladores y usuarios, ya que ahorran tiempo de prueba, son fáciles de entender y mantener y pueden crearse a partir de problemas encontrados, como por ejemplo, en los checklists de validación de páginas.

Debe también tenerse en cuenta es que las sucesivas modificaciones en las interfaces requieren que los casos de prueba puedan ser reusables a través de las distintas versiones operativas del software, por lo que aquéllos cuya funcionalidad haya quedado obsoleta deben ser desechados. Así, en la aplicación desarrollada, queda como trabajo futuro la administración de los casos de prueba según su grado de obsolescencia.

Una conclusión importante es la necesidad de realizar el testing de migración de los sistemas distribuidos separándolos en subsistemas de acuerdo a la importancia de su implementación. Así por ejemplo, en el caso de estudio aquí tratado se tomó el *subsistema de autogestión de alumnos*, ya que se consideró que el mismo era prioritario dentro del entorno general de la problemática de la organización involucrada. Cabe acotar que la propuesta presentada en este trabajo está utilizándose actualmente en el ámbito del Instituto Universitario Aeronáutico.

Una vez completada esta etapa, se prevé aplicar esta metodología para la futura migración del *subsistema docente*, cuyo análisis y diseño están planificados para un futuro próximo. En este caso, la implementación de la metodología acompañará en forma total el proceso de migración.

A partir de estas experiencias también se ha previsto profundizar la investigación sobre la herramienta *robot de Rational*, potenciando sus capacidades de integración con todas las otras herramientas como *Rose, Requisite Pro y Clear Quest*, a los fines de poder implementar el proceso iterativo en las pruebas de regresión en el sistema migrado.

He llegado así a completar los objetivos que fueron planteados al comenzar ésta tesis, que en resumen, fueron los de presentar una metodología de análisis para la migración de aplicaciones distribuidas a entornos Web y aplicarla a un caso de estudio de un sistema real, en este caso el Sistema de Gestión Académica de una institución universitaria. El enfoque propuesto se orientó al testing, con reutilización de casos de prueba y la utilización de herramientas automáticas. Comencé entonces un largo camino cuyo final parecía muy lejano y por momentos hasta inalcanzable, y ahora al haber llegado, compruebo que es en realidad solo el final de una primer etapa de un camino que quizás sea infinito.

En efecto, debo reconocer que su desarrollo me dio la oportunidad de ahondar en los procesos involucrados en la verificación y validación de sistemas, comprobando que se trata de un mundo complejo, variado, multifacético y de renovada vigencia, que terminó resultando fascinante. La incesante evolución de la tecnología y la creciente difusión de los sistemas de computación en todos los ámbitos del quehacer humano permiten prever que la migración de sistemas a nuevos contextos será un problema recurrente. Y la problemática del testing será naturalmente un renovado desafío en el que espero continuar involucrada.

ANEXOS

Cabe señalar que los anexos del presente trabajo reúnen una selección de material bibliográfico consultado a lo largo del desarrollo de la investigación y que se incluyen dada la importancia de los mismos.

ANEXO I – Metodologías de desarrollo de software

I.1 Objetivo de las metodologías de desarrollo de software

En general, y, atendiendo a distintas definiciones genéricas, se puede considerar que una metodología representa el camino o guía para el desarrollo de software en forma sistemática. Los métodos de desarrollo de software pueden dividirse en dos grupos: función/dato y orientados a objetos, clasificación basada en el análisis efectuado por Piattini (Piattini y Calvo, 2004) según se observa en la tabla 25.

Tabla 25 - Métodos de desarrollo de software

Orientado a función/dato	Orientado a objetos
Énfasis en la transformación de datos	Énfasis en la abstracción de datos
Funciones y datos tratados como entidades separadas	Funciones y datos encapsulados en entidades fuertemente relacionadas
Difícil de entender y modificar	Facilidades de mantenimiento

a) Orientado a función/dato

Aquellos métodos en los cuales las funciones y/o los datos son tratados como entidades independientes. Estos sistemas resultan difíciles de mantener. El mayor problema es que las funciones generalmente dependen de la estructura de los datos. A menudo diferentes tipos de datos tienen distintos formatos y se necesita verificar el tipo del dato (con sentencias If-Then o CASE), produciendo programas difíciles de leer y modificar. Si se desea hacer alguna modificación en la estructura de los datos se debe modificar en todos los lugares donde es utilizado. Otro problema es que una persona no piensa naturalmente en términos de una estructura. La especificación de requerimientos se hace en lenguaje común, se especifica la funcionalidad que debe tener el sistema y no en cómo se deben estructurar los datos.

El análisis estructurado fue la metodología de análisis de sistemas que tuvo uno de los impactos más grandes en el ambiente de producción del software. Desde sus orígenes a mediados de los años setenta, con los libros de Tom Demarco y Chris Gane y Trish Sarson, y luego durante la década siguiente se extendió rápidamente hasta volverse en una metodología estándar. En los ochenta, muchos autores trabajaron en su modernización y en la incorporación de nuevas herramientas y estrategias de modelamiento.

El Análisis Estructurado Moderno es una propuesta de Edward Yourdon que, establece el modelo standard para el análisis estructurado e incorpora las ideas, criterios y herramientas presentados por los libros que lo precedieron, en una estructura de trabajo muy organizada y consistente.

El diseño estructurado de sistemas se preocupa por la identificación, selección y organización de los módulos y sus relaciones. Se comienza con la especificación resultante del proceso de análisis, se realiza una descomposición del sistema en módulos estructurados en jerarquías, con características tales que permitan la

implementación de un sistema que no requiera elevados costos de mantenimiento. La idea original del diseño estructurado fue presentada en la década de los '70, por Larry Constantine, y continuada posteriormente por varios autores tales como Myers, Yourdon y Stevens.

b) Orientado a objetos

Son aquellos métodos en los cuales datos y funciones están altamente relacionados. El énfasis está centrado en la abstracción de datos. Se piensa en forma natural, los objetos son mapeados a entidades del mundo real. Los programas son fácilmente mantenibles y extensibles por medio de la construcción de subclases. Varios métodos de desarrollo de software han sido propuestos para cada uno de estos grupos, desde finales de los 80 y durante los 90. Cada uno de ellos introduce un proceso de desarrollo, un conjunto de modelos y una notación para explicitar los mismos. La orientación a objetos requiere de una metodología que integre el proceso de desarrollo y un lenguaje de modelamiento con herramientas y técnicas adecuadas.

Al final de los 90, Grady Booch, James Rumbaugh e Ivar Jacobson aunaron esfuerzos para combinar las mejores características de sus métodos en uno unificado. Surgió así el Proceso Unificado de Desarrollo de software que utiliza como notación el Lenguaje Unificado de Modelado (UML), que se ha convertido en un estándar en la industria del software.

I.2 El proceso de especificación de requerimientos

Consiste en establecer de un modo conciso, claro y preciso el conjunto de requisitos que deben ser satisfechos por el software a desarrollar. El objetivo es determinar en forma total y consistente los requisitos de software. El análisis se realiza sobre la salida resultante, la descomposición de los datos, el procesamiento de los mismos, las bases de datos y las interfaces de usuario.

Se debe considerar que un requisito es una condición o característica que debe tener el programa para satisfacer un documento formal. Estos requisitos pueden ser funcionales, de rendimiento o de interfaces. Los primeros especifican la función que el programa debe realizar, los segundos especifican una característica numérica; y los últimos determinan las características de las interfaces, usuario-software, software-hardware y software-software.

Tabla 26 - Técnicas para la especificación de requisitos

Especificación de requisitos	
Actividades a realizar	Definir y desarrollar los requisitos del software y de las interfaces.
Técnicas a usar	Técnicas orientadas a los procesos: Análisis estructurado: diagramas de flujo de datos (DFD), diccionario de datos (DD), especificación de procesos. Diagramas de actividades. Técnicas orientadas a los datos: Diagramas entidad relación y diagramas de datos. Técnicas orientadas a los objetos.

	Diagramas de clases/objetos Jerarquía de clases/objetos Técnicas formales de especificación: Técnicas relacionales: ecuaciones implícitas, relaciones recurrentes, axiomas algebraicos. Técnicas orientadas al estado: tablas de decisión, de eventos, de transición, mecanismos de estado finitos, etc. Técnicas de prototipación.

Los beneficios de una definición temprana y precisa de los requisitos del sistema han sido ampliamente enfatizados en el contexto del desarrollo de un producto software. Numerosos autores han señalado que la ingeniería de requisitos juega un papel estratégico a la hora de desarrollar software lo cual es más relevante en los entornos de desarrollo mediante la reutilización sistemática de elementos software existentes. Por un lado, el desarrollado con reutilización requiere la selección adecuada de los elementos dentro del previsiblemente amplio conjunto de ellos almacenado en un repositorio. Y por otro lado, la reutilización supone una mejora sustancial en la productividad de sistemas software. De manera que la precisa especificación de los requisitos del software es un elemento clave para aprovechar las ventajas de la reutilización.

Los requisitos software son el producto final de un proceso que se inicia con la fase de elicitación y que continúa con el análisis de requisitos del software. Este análisis de requisitos trata con dominios específicos organizacionales para indicar en cada dominio cuáles son las características que debe cumplir un producto software en términos de lograr un uso efectivo de la información acorde con los objetivos generales de la organización. En este sentido, se tienen dos aspectos relevantes. Por un lado, que el análisis de requisitos debe conducir a un documento de especificación de requisitos del software. Y por otro lado, que dichos requisitos deben atender tanto al nivel de cada dominio como al nivel más general de la organización.

El documento de especificación define los atributos deseables del software a desarrollar. Dentro de esos requisitos se encuentra la funcionalidad, es decir los requisitos funcionales, que especifica los detalles más relevantes de las funciones que el producto realizará, con independencia de los requisitos no funcionales y de las restricciones de diseño. De acuerdo con Presman, la funcionalidad debe cubrir la representación y la comprensión del ambiente específico y debe contener todas las funciones esenciales (funcionalidad delimitada) que el software deberá realizar. En concreto, los requisitos funcionales se refieren a las funciones específicas del software y definen qué es lo que se espera que realice el producto software que se desarrollará.

El principal problema de la especificación de los requisitos funcionales estriba en que ésta generalmente consiste en una colección de frases en lenguaje natural con los correspondientes problemas de ambigüedad, escalabilidad y trazabilidad.

Los casos de uso (UC) y los casos de uso del negocio (BUC), propuestos por Jacobson, han surgido como una estrategia de captura de requisitos ampliamente aceptada y que representa los requisitos funcionales como una colección de escenarios de interacción entre el usuario y el sistema. Aunque resuelven las deficiencias de escalabilidad y trazabilidad, y proporcionan facilidad para la descripción, los casos de uso no superan la ambigüedad del lenguaje natural. Se requieren enfoques que permitan derivar los casos de usos en concordancia con algún formalismo sintáctico y semántico, lo cual ya ha sido señalado por Lee y otros. En este contexto, Laguna ha propuesto determinar la funcionalidad inicial del sistema software a partir de la descripción de la forma actual del quehacer del usuario.

I.*3 Determinación de la interfaz del usuario*

Hay varios estudios estadísticos de sistemas interactivos que indican que, entre el 40% y el 60% del código y datos de los programas están dedicados a la implementación de la interfaz hombre-máquina. Este hecho pone en evidencia la importancia en costos de mantenimiento de un adecuado diseño de la interfaz hombre-máquina. La especificación de la interfaz de usuario es, probablemente, la actividad que más tiempo consume y en la que el usuario más interesado está. Eso involucra cuatro problemas relacionados:

- La elección de dispositivos de entrada y salida (por ejemplo, las terminales alfanuméricas o gráficas, dispositivos de lectura óptica, impresoras de gran velocidad, etc.). Por ejemplo, en algunos casos, un requisito importante es la posibilidad de hacer análisis estadístico del desempeño de las funciones de algunas de las áreas de organización (como volúmenes de venta de ciertos productos relacionados al periodo anual, o los volúmenes de ventas generales relacionados al periodo mensual, etc.), o proyecciones financieras respecto a mercaderías en stock. En esos casos, la elección de una terminal gráfica para la gerencia (de ventas, de stock o financiera) puede ser un requisito fundamental. También, si se identifican volúmenes muy elevados de facturación, una impresora de gran velocidad puede ser muy importante.
- El formato de las entradas que fluyen de los agentes externos hacia el sistema. Es preciso especificar con gran detalle las características de los datos ingresados y como ellos serán aceptados.
- El formato de las salidas que fluyen del sistema hacia los agentes externos. ¿Se representan ellos en la pantalla de la computadora o en informes impresos? Comúnmente el formato de los informes impresos ya es predeterminado y, es común que el usuario quiera mantener estos formatos. Si el agente externo es gubernamental, habitualmente los informes que se desean tienen un formato muy estricto.
- La secuencia y el escalonamiento de entradas y salidas en el componente en línea. Estos escalonamientos deben diseñarse con mucho cuidado.
- El diseño de interfaces de usuario ha evolucionado desde una estructura de menús esencialmente jerárquica, en dos dimensiones, a una estructura que agregó dimensiones al superponer ventanas.

Piattini señala que es probable que la tendencia sea la de agregar dimensionalidad con más tipos de medios que serán altamente portables y personales, a la vez de la utilización de tecnologías de comunicaciones para conseguir una gran conectividad.

La actividad de las interfaces de usuario comienza ya en la fase de análisis con el trabajo conjunto de analistas y usuarios al definir la interacción que se desea con la aplicación, de hecho, los casos de uso carecen de sentido sino se acompañan con la definición de la interfaz.

Para el diseño de la interfaz es recomendable la utilización de un prototipo, que en algunos casos suele evolucionar hasta constituir parte de la aplicación final. El diseño de pantallas se transforma entonces en un proceso ordenado, que comienza en la etapa de requisitos y finalizan con la implementación. Existen ciclos de vida específicos para guiar este proceso.

La disciplina de diseño de interfaces ha experimentado un gran impulso con el desarrollo de de las aplicaciones web, como veremos más adelante, teniendo en cuenta los estudios para fomentar la “usabilidad”, y, otros valiosos aportes para el diseño entre los que podemos citar a Nielsen y Spolsky.

I.4 Metodologías de diseño Web

La crisis del software de alguna manera se está repitiendo en el diseño de los sitios Web, bien por falta de personal capacitado para desarrollar este tipo de aplicaciones o por la complejidad y lo extenso de las mismas. Lo cierto es que las compañías invierten mucho dinero en el desarrollo de sus sitios Web, y por lo general cuando son interrogadas sobre su efectividad real, son pocas las que tienen esperanzas en el retomo de la inversión total hecha para su creación, esto se debe en buena medida a la repetición de los errores cometidos en los años 60. La audiencia, quienes son los verdaderos usuarios de los sitios Web, no es consultada sobre sus requerimientos. El ignorar el grado de amigabilidad, legibilidad y comprensión de un sitio Web se paga casi siempre con el desuso.

Los problemas de diseño Web han sido analizados por diversos autores –Nielsen (Nielsen,1990) y Cabero (Cabero, 1995), - destacando los conflictos de diseño como la navegación, la estructura del sitio, la interfaz, y los medios. También, los problemas propios de diseño de software como las metodologías, los equipos interdisciplinarios, los conflictos de diseño comunicativos e instruccional, que deben ser estudiados desde diferentes puntos de vista para encontrar soluciones integrales.

- Los conflictos de navegación y de estructuración del sitio Web: estos problemas están ligados al sentimiento de extravío, es decir, la sensación de sentirse perdido dentro de la Web, la cual ocurre cuando hay un desborde cognitivo por exceso de información o porque la estructura del sitio es pobre y no indica dónde se encuentran los recursos. Una pregunta clave en la navegación y estructuración del sitio Web y que siempre debemos tener presentes es ¿dónde está el usuario? Así mismo, los usuarios tienen problemas para localizar información específica ya que ésta puede no estar bien estructurada, lo que nos lleva a pensar en la segunda pregunta clave a responder en el diseño ¿Qué ofrece el sitio? ¿Qué hay y dónde está?

- Los conflictos de equipos interdisciplinarios: Son problemas propios de la ingeniería de software, ya que para el correcto desarrollo de aplicaciones educativas es indispensable trabajar con equipos interdisciplinarios, conformados por expertos en los temas tratados, así como expertos en multimedia, en audiovisuales, programadores, diseñadores gráficos entre otros. Hoy existe un consenso sobre el carácter multidisciplinario del desarrollo de software educativo, tanto dentro como fuera de línea. Sin embargo, la realidad nos dice que la integración de equipos de profesionales de diversas áreas, aun es problemática y según Rojas, en muchos casos, en el desarrollo de software educativo la mayoría de las tareas aun son desempeñadas o lideradas por los ingenieros de software.

- El problema de la metodología está relacionado con la construcción de) software y su ciclo de vida. Resulta un error emprender un desarrollo si antes no existe una metodología (esta lección costó millones de dólares y a ello se debe la existencia de la ingeniería de software). Además, son pocas las metodologías de desarrollo de sistemas de información Web (SIW), y en muchos casos su difusión es escasa. El desarrollo de un SIW por ser un área muy reciente en la ingeniería de software cuenta con un número de especialistas muy limitado para la demanda del mercado Rojas y Nielsen.

- Los conflictos de diseño instruccional: Estos problemas ya no son generales del diseño Web, sino que obedecen directamente a los sitios Web educativos analizados en esta investigación. Están ligado al desborde cognitivo y en buena medida a la navegación, algunos autores -Lan, Loh- indican que la desorientación está estrechamente relacionada con la libertad de navegación, y con los lineamientos pedagógicos y teorías educativas donde se fundamente la

metodología del curso. Junto a este problema también se debe considerar el definir y establecer el tipo de comunicación (bidireccional) del curso.

- Los problemas de la interfaz Web radican en que la experiencia previa en las interfaces gráficas de usuario (GUI) no pueden ser extrapoladas directamente, ya que existen diferencias que deben ser consideradas en el diseño de las interfaces Web.

El desarrollo de aplicaciones Web involucra decisiones no triviales de diseño e implementación que inevitablemente influyen en todo el proceso de desarrollo, afectando la división de tareas. Los problemas involucrados, como el diseño del modelo del dominio y la construcción de la interfaz de usuario, tienen requerimientos disjuntos que deben ser tratados por separado.

El alcance de la aplicación y el tipo de usuarios a los que estará dirigida son consideraciones tan importantes como las tecnologías elegidas para realizar la implementación. Así como las tecnologías pueden limitar la funcionalidad de la aplicación, decisiones de diseño equivocadas también pueden reducir su capacidad de extensión y reusabilidad. Es por ello que el uso de una metodología de diseño y de tecnologías que se adapten naturalmente a ésta, son de vital importancia para el desarrollo de aplicaciones complejas.

Existen en la actualidad tecnologías ampliamente usadas para el desarrollo de aplicaciones Web, pero muchas de ellas obligan al desarrollador a mezclar aspectos conceptuales y de presentación, lo que sucede principalmente con aquellas tecnologías no basadas en objetos,

La elección de tecnologías complejas demora el proceso e incrementa los costos, pero en ocasiones permite adecuarse a metodologías de diseño más fácilmente. Tal es el caso de las tecnologías orientadas a objetos, las cuales tienden a demorar el desarrollo en etapas tempranas. El tiempo de desarrollo en la actualidad es crítico, tanto por razones de marketing como por límites en el presupuesto y los recursos, pero la adopción de estas tecnologías hace que el mantenimiento se transforme en una actividad más simple, la división en capas sea tarea natural del desarrollo y el tiempo invertido en el diseño facilite el trabajo necesario para el resto de las actividades.

I.5 Aplicaciones Web y la importancia del desarrollo en capas

Las aplicaciones hipermedia han evolucionado en los últimos años y se han concentrado mayormente en la Web. Las antiguas aplicaciones distribuidas dieron lugar a aplicaciones dinámicas, de constante actualización e incluso personalizables, capaces de adaptarse a los tipos de usuarios y en casos avanzados, a cada usuario en particular. Estas características encuentran el medio ideal en la web, ya que de otra forma sería costoso su mantenimiento y evolución.

La complejidad del desarrollo ocurre a diferentes niveles: dominios de aplicación sofisticados (financieros, médicos, geográficos, etc.); la necesidad de proveer acceso de navegación simple a grandes cantidades de datos multimediales, y por último la aparición de nuevos dispositivos para los cuales se deben construir interfaces web fáciles de usar. Esta complejidad en los desarrollos de software sólo puede ser alcanzada mediante la separación de los asuntos de modelización en forma clara y modular.

La metodología OOHDM, ha sido utilizada para diseñar diferentes tipos de aplicaciones hipermedia como galerías interactivas, presentaciones multimedia y como veremos en este artículo, aplicaciones web. El éxito de esta metodología es la clara identificación de los tres diferentes niveles de diseño en forma independiente de la implementación.

La justificación de tanto trabajo puede encontrarse en cualquier aplicación que

requiera navegación: en términos de programación orientada a objetos, si los elementos por los que se navega son los del diseño conceptual se estaría mezclando la funcionalidad hipermedia con el comportamiento propio del objeto. Por otro lado, si los nodos de la red de navegación tienen la capacidad de definir su apariencia, se estaría limitando la extensión de la aplicación para ofrecer nuevas presentaciones del mismo elemento y eventualmente se estaría dificultando la personalización de la interfaz.

Es necesario, entonces, mantener separadas las distintas decisiones de diseño según su naturaleza (conceptual, navegacional, de interfaz) y aplicar las tecnologías adecuadas a cada capa en el proceso de implementación.

Las metodologías tradicionales de ingeniería de software, o las metodologías para sistemas de desarrollo de información, no contienen una buena abstracción capaz de facilitar la tarea de especificar aplicaciones hipermedia. El tamaño, la complejidad y el número de aplicaciones crecen en forma acelerada en la actualidad, por lo cual una metodología de diseño sistemática es necesaria para disminuir la complejidad y admitir evolución y reusabilidad.

Producir aplicaciones en las cuales el usuario pueda aprovechar el potencial del paradigma de la navegación de sitios web, mientras ejecuta transacciones sobre bases de información, es una tarea muy difícil de lograr.

En primer lugar, la navegación posee algunos problemas. Una estructura de navegación robusta es una de las claves del éxito en las aplicaciones hipermedia. Si el usuario entiende dónde puede ir y cómo llegar al lugar deseado, es una buena señal de que la aplicación ha sido bien diseñada.

Construir la interfaz de una aplicación web es también una tarea compleja; no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

En hipermedia existen requerimientos que deben ser satisfechos en un entorno de desarrollo unificado. Por un lado, la navegación y el comportamiento funcional de la aplicación deberían ser integrados. Por otro lado, durante el proceso de diseño se debería poder desacoplar las decisiones de diseño relacionadas con la estructura navegacional de la aplicación, de aquellas relacionadas con el modelo del dominio.

OOHDM propone el desarrollo de aplicaciones hipermedia a través de un proceso compuesto por cuatro etapas: diseño conceptual, diseño navegacional, diseño de interfaces abstractas e implementación (Silva y Mercerat (2001), “Construcción de aplicaciones web utilizando una metodología de Orientación a objetos (OOHDM)”).

I.6 Establecer una medida de éxito

Un aspecto importante a tener en cuenta en todo sitio web es establecer unos criterios que puedan determinar el éxito del mismo, o lo que es lo mismo, si cumple con los objetivos para los cuales fue desarrollado. En la mayor parte de los casos la medida estará directamente relacionada con el objetivo del sitio, siendo el número de visitas, el de solicitudes, de ficheros descargados o de ventas realizadas parámetros habituales que rigen este parámetro.

En el caso del libro digital el éxito vendría dado por el número de descargas de los capítulos, en el de un congreso, el número de personas que finalmente se registrarán en el mismo, en el caso de la web de la infancia sería el número de niños de la ciudad de Lleida que lo consultan y participan en sus juegos, en el caso de la estantería virtual el número de profesores que han depositado material docente correspondiente a sus asignaturas y el número de alumnos que lo utilizan, en el centro excursionista será en el número de personas que acceden a la parte pública y en el número de socios que consultan tanto la parte pública cómo la privada y la frecuencia con que lo hacen, etc. Otra manera de considerar el éxito de un sitio es su clasificación en los buscadores y

directorios de sitios web más importantes.

I.7 Prototipado en los requisitos

- Maquetas. Las maquetas son básicamente una sola página de representaciones estáticas del espacio de diseño. Se usan para mejorar el diseño y facilitar la comunicación entre los diseñadores, usuarios e implicados.
- Boceto. Los bocetos son maquetas rápidas, pequeñas para desarrollar un amplio espectro de ideas de diseño. Se usan típicamente en la primera etapa del diseño, muchas veces de que se haya acabado la fase de análisis de requisitos. La clave de los bocetos es su velocidad. Cada boceto no puede costar más de 15 ó 20 segundos, de esta manera se pueden generar una gran cantidad de bocetos en poco tiempo.
- Maqueta de papel. Son representaciones de una mayor cualidad que los bocetos. Permite explorar el diseño de la página, aspectos estéticos. Permite una comprensión más realista de los límites del tamaño de la página, del espacio de diseño, identificar posibles dificultades en el proceso de diseño y comenzar la evaluación con los usuarios.
- Maqueta digital. El prototipo digital constituye un paso más en la elaboración de prototipos el cual es realizado con herramientas de diseño gráfico, siendo, por tanto más costosos y elaborados. Permiten afinar aspectos importantes del diseño cómo son los colores, los contrastes, las fuentes tipográficas, etc. que el prototipo de papel no proporciona.

a) Evaluación en la fase de requisitos

Realizar encuestas es especialmente útil en las etapas más iniciales del proyecto. Esta técnica especialmente indicada para conseguir una definición precisa de la audiencia siendo además una técnica con una buena relación coste-beneficio.

Éstas, además, en función de esta audiencia a la cual va destinada casi siempre pueden realizarse a través del correo electrónico y tecnología basada en Internet (alcanzando así un espectro de población muy amplio y diverso).

Entrevistas y/o Grupos de Discusión (“Focus Groups”) con implicados (stakeholders) haciendo uso de maquetas o prototipos de papel proporcionan reacciones subjetivas acerca de suposiciones que ayudan a entender su entorno y cómo tratan de resolver sus problemas. El carácter individual de las entrevistas hace que los resultados obtenidos carezcan de influencias externas. Por el contrario las influencias del grupo ensalzan aquellas ideas que un miembro destapa. Por lo tanto combinar ambas técnicas suele ser altamente beneficioso.

Evaluaciones a partir de descripciones formales de escenarios de casos de uso describen los requerimientos del sistema en el contexto de las especificaciones funcionales mostrando como se efectúan los procesos de negocios y que actores o perfiles de usuario intervienen en estos a través de las secuencia de tareas descritas para cada uno de los escenarios.

Evaluaciones del tipo recorrido cognitivo donde los usuarios evalúan preferentemente maquetas digitales son especialmente útiles cando nos encontramos en una iteración un poco adelantada del modelo de proceso.

b) Diseño

Durante la fase de análisis y especificación de los requisitos el objetivo es documentar lo que el sitio debe hacer. En la fase de diseño es importante determinar como será el sitio estructuralmente y gráficamente. En esta parte presentamos dos aspectos importantes en el diseño de un sitio web, el análisis de tareas y la arquitectura de la información. En los aspectos de prototipado y evaluación se presentan la ordenación de tarjetas, la maqueta digital y el storyboard de uso.

c) Análisis jerárquico de tareas (HTA)

El análisis jerárquico de tareas (HTA Hierarchical Task Analysis) desarrollado por Annett y Duncan, es la técnica de análisis de tareas más conocida y más antigua, que sigue siendo válida aunque nuevas metodologías han aparecido.

En HTA se realiza una descripción de tareas en términos de operaciones y planes.

Las operaciones (descomposición en subtareas) son actividades que realizan las personas para alcanzar un objetivo, y los planes son una descripción de las condiciones que se tienen que dar cuando se realiza cada una de las actividades. Las operaciones se pueden descomponer de forma jerárquica y se asigna un plan a cada una de las subtareas que aparecen. Se define un objetivo como un estado determinado del sistema que puede quiere alcanzar el usuario. Aunque se habla de objetivos y tareas, la representación que se realiza describe únicamente la descomposición jerárquica en subtareas de las tareas que aparecen en el sistema.

El formato gráfico se parece a un árbol con ramas y subramas en función de las necesidades. A la hora describir la descomposición de unas tareas en subtareas podemos representar cuatro tipos de descomposiciones:

- Secuencia. Descomposición en un conjunto ordenado temporalmente de una secuencia de tareas.
- Selección. Conjunto de tareas de las que se tendrá que elegir una de ellas.
- Iteración. Repetición de un subconjunto de tareas.
- Tarea unitaria. Actividad indivisible (según el nivel de detalle dado).

El análisis de tarea implica tres etapas enlazadas: recogida de información, diagramación y análisis. Los procedimientos de recogida de información incluyen la revisión de la documentación existente (por ejemplo, manuales de funcionamiento, procedimientos, informes de seguridad, estudios de análisis de tareas previos, diseños, imágenes, prototipos, etc.), que permitan establecer qué hacen las personas en circunstancias especificas (normales y anormales), entrevistas y cuestionarios (descripciones por parte de personas experimentadas como hacen las cosas, que informaciones necesitan y como determinan si la tarea se puede realizar satisfactoriamente).

Algunas tareas se pueden desglosar con mayor detalle en secuencias. Un plan describe el conjunto de operaciones necesarias para llevar a cabo una actividad, o bien, muestra las circunstancias por las que una operación es realizada antes que otra.

Estos planes se añaden a la tabla jerárquica. La descripción de la información se realiza en forma de tabla o en forma de diagrama de árbol que describa las relaciones entre tareas y subtareas.

I.8 Comparación entre diseño de interfaces Web y diseño de interfaces gráficas de usuario (GUI)

El diseño de interfaces Web comparte similitudes con el de las interfaces gráficas de usuario (GUI). Ya que ambas interfaces son interactivas, ambos son diseños de software y en principio persiguen objetivos semejantes. Sin embargo, no son exactamente la misma actividad y los resultados de un tipo de interfaz no pueden extrapolarse directamente al otro. En este apartado, se identifica la problemática para entender la diferencia entre diseño Web y el software tradicional.

Los problemas a los que se enfrente al diseño ***Web*** son los siguientes:

- Diversidad de dispositivos de ‘display’. En el diseño de GUI tradicional el diseñador conoce de antemano el dispositivo de display (pantalla, proyección), pudiendo

controlar cada pixel de ella, es decir, el diseñador conoce para qué sistema se está diseñando (ancho y largo en pixeles- de pantalla definido, número de colores, etc.). En el diseño Web esta presunción es imposible, no existe regias, no es controlable el dispositivo con el cual interactúa el usuario, este podría estar viendo el sitio Web bien desde una computadora de escritorio, o una estación de trabajo, o bien desde su móvil, o su televisor y cada uno de estos dispositivos tiene una área de display diferente, como puede observarse en la tabla 2VII.

Tabla 27 - Tipos de display

Área de display en pixeles	*Porcentaje*	*Display*
640 x 480 pixeles = 307200 pixeles	100	Estándar de PC (VGA)
320 x 240 pixeles = 76800 pixeles	25	Std CE
100 x 180 pixeles = 16000 plxeles	5	Teléfono Móvil típico

- Diversidad de dispositivos de conexión. Al igual que en el problema de los disp!ay, el diseñador de interfaz Web se enfrenta con las disparidades de velocidades de conexión. En el diseño tradicional se puede "conocer" (usualmente estimar) el tiempo entre un requerimiento del usuario y la respuesta emitida por el sistema, ya que se conoce el sistema y las características de su arquitectura, pero por la variedad de la arquitectura Web es imposible conocer las velocidades de conexión, y por ende, los tiempos de respuestas, es decir, el usuario tendrá un tiempo de respuesta diferente dependiendo del dispositivo de conexión.

Tabla 28 - *Tipos de conexión a Internet*

Conexión	Velocidad
MODEM RTB	**28 Kb por segundo**
RDSI 64Kbps	**64 Kb por segundo**
RDSI 128 Kbps	**128 Kb por segundo**
TI, ADSL	**2 MB por segundo**

- Control de la navegación: En la GUI tradicional el diseñador conoce exactamente el recorrido. La navegación de los usuarios define exactamente el orden de aparición de las pantallas, se puede habilitar o no las opciones de un menú dependiendo de la ubicación del usuario en el programa, en la Web este control se pierde, es más, se desconoce de qué página proviene el usuario, ya que podría venir de un enlace interno del sitio Web o de un enlace externo, por ejemplo de un buscador, o directamente desde su opción de sitios favoritos.
- Diferencia entre cliente y usuario: Cuando se diseña una GUI tradicional, se tiene la ventaja de conocer al cliente, se puede y de hecho se hacen estudios para probar las interfaces (prototipos). Cuando se diseña para una empresa un sitio Web, no se está diseñando para los dientes, es decir, para la empresa, ni para sus empleados, se esta diseñando para los usuarios del sitio Web y esto hay que tenerlo muy presente.

- Formar parte de un todo: El software tradicional es entendido como una unidad, es en esencia una aplicación cerrada, esto implica la utilización de premisas (válidas en dichos entornos) como lo es que el usuario ve e interactúa con el todo. Los usuarios Web no son los mismos de las GUI tradicionales, por lo tanto no se debe considerar que estos vayan a familiarizarse con el sitio Web y es ingenuo pensar que leerán las ayudas en línea como lo hace en una GUI tradicional. El usuario está familiarizado con la Web, es decir, obedece a metáforas, patrones de diseños ya establecidos y no debemos imponer un estilo propio que rompa las ideas de navegación del usuario, si esto ocurre es muy probable que el usuario realice una sola visita al sitio Web y al verse frustrado decida abandonar la página.
- Orden de ejecución de los eventos: Los eventos de una GUI siempre están controlados por la aplicación, en cambio en la Web cuando el usuario accede a una página del servidor Web para visualizada, el browser baja la página al cliente, es decir, en la máquina del usuario, cediendo el control a éste, hasta que el usuario dispare el evento ENTER que es cuando se envía el resultado al servidor para su procesamiento. Es allí cuando las variables toman los valores ingresadas por el usuario, esto difiere del orden de ejecución de los eventos de una aplicación GUI tradicional puesto que en ella se tiene el control durante todas las fases, según lo establecen Musciano y Kennedy (2002).

I.9 Desarrollo de sitios Web

En el desarrollo de sitios Web podemos identificar dos niveles de diseño, que están relacionados al nivel de abstracción, necesario para modelar las estructuras y los componentes. El primer nivel de diseño llamado enfoque sistemático pretende analizar la estructura. Autores como Halasz, Hardman, Garzotto, Mondelo, Marqués y Chaar, han creando metodologías que han sido en algunos casos adaptaciones de las ya existentes en el área de multimedia, probando un enfoque de reutilización de las metodologías (reutilización de componentes de software), y en otros casos, metodologías completamente nuevas que han nacido de los entornos Web y que en buena medida han dado resultados satisfactorios. El segundo nivel de diseño llamado enfoque de detalle, caracterizado por el análisis en detalle de sus elementos y como pueden integrarse para construir la página Web.

a) Diseño. Enfoque sistemático

El nivel de diseño llamado enfoque sistemático, pretende construir modelos de las estructuras que soportan la navegación del sitio Web. Los primeros modelos estaban ligados al paradigma relacional y en buena medida son los más divulgados, sin embargo tienen la limitación de que son metodologías para el desarrollo de software multimedia y que requieren de pequeñas adaptaciones para llegar a ser metodologías de diseño de sitios Web. La mayoría de las metodologías de diseños de sitios Web están ligadas al paradigma de orientación a objetos, el cual pretende capturar la estructura del sitio Web y el comportamiento de los elementos de dicha estructura. Estas metodologías están orientadas al desarrollo de sitios Web. Los modelos más utilizados en el diseño de herramientas multimedia son:

- Modelo de diseño de hipertexto HDM (Hypertext Design Model). Propuesto por Garzotto (2001), es el primer modelo de multimedia que se publicó. Introduce la terminología de la multimedia y los conceptos de entidad, tipos de entidades y estructura de acceso, su mayor logro es el permitir presentar la información de varias formas.
- Modelo de referencia Dexter. Es un modelo genérico propuesto por Halasz y Schwartz para sistemas hipertexto, publicado en 1994, divide al sistema en tres capas o niveles: La capa de almacenamiento, donde se describe la estructura de un conjunto finito de componentes y de dos operaciones básicas denominadas

"resolver" y "accesor" para recuperar los componentes; la capa de componentes, que permiten especificar la presentación, los atributos, las andas, y los hijos de un componente. La capa de ejecución, que permite la instanciación de un componente al usuario. Sus detractores indican que fue destinado para hipertexto, no consideran el tiempo ni la necesidad de sincronizar los componentes.

- Modelo Ámsterdam. Este modelo desarrollado por Hardman, Bulterman y Van Rossum (1993), cubre las fallas del modelo Dexter, es decir, toma en consideración el tiempo y cumple con el criterio de sincronización indispensable en los sistemas multimedia, para ello considera dos relaciones temporales entre los datos, una de ellas llamada colección, que define una relación temporal entre componentes que estarán presentes conjuntamente, y la otra es la sincronización que define una relación temporal que se refiere al orden relativo de presentación.
- Metodología RMM (Relationship Management Methodology). Está basada en los conceptos implantados en el Modelo de diseño de hipertexto HDM, es decir, en las entidades y los tipos de entidades. Su objetivo es mejorar la navegación a través de un análisis de las entidades del sistema. En teoría, se obtiene una navegación más estructurada y logra que ésta sea más intuitiva para el usuario. La metodología fue creada por Isakowitz, Stohr y Balasubramanian en 1995. Los conceptos de slices y m-slices, que consisten en la agrupación de datos de una entidad en diferentes pantallas, es una de las aportaciones más importantes de esta metodología. Fue la primera metodología completa que se publica para la creación de software multimedia. Su problema principal es que no permite realizar consultas a partir de dos entidades y como está muy atado al modelo entidad relación (modelo E-R) cuando se define una relación (M:N) se obliga a descomponerlas en dos relaciones (1:N) copiando el modelo E-R. Además, no considera las consultas a la base de datos para la creación de páginas Web dinámicas.
- Modelo de Diseño Hipermedia Orientado a Objeto OOHDM (The Object-Oriented Hypermedia Design Model). Propuesto por D. Schwabe, G. Rossi, and S. Barbosa, en 1996, es el sucesor del modelo de diseño de hipertexto HDM, se trata de una metodología que se fundamenta en la orientación a objeto. Propone las siguientes fases: Diseño conceptual o análisis de dominio que utiliza el método de análisis orientado a objeto para obtener esquemas conceptuales de las ciases y de las relaciones entre las mismas. Utiliza las técnicas de modelado de objeto" llamada notación OMT para el diseño de navegación, donde se define la estructura de navegación por medio de modelos, es decir, a través de diferentes vistas del esquema conceptual; la fase de diseño de interfaz abstracta, se apoya en un modelo orientado a objeto para especificar la estructura y el comportamiento de la interfaz del sistema, este modelo se crea a través de tres tipos de diagramas: diagramas abstractos para cada clase, diagramas de configuración para reflejar los eventos externos y diagrama de estado para señalar el comportamiento dinámico; y por último, la fase de implementación, es decir, la construcción de los programas en programación orientada a objeto.

Los modelos más utilizados en el diseño de sitios Web son:

- Modelo DRMM (Dynamíc Relatíonship Management Model). Presentado por Rojas Durán(1998), está basado en el modelo RMM sólo que considera las aplicaciones dinámicas, permitiendo el diseño de pantallas con resultados de consultas realizadas por el usuario a una base de datos.
- Método del diseño de sitios Web centrados en el usuario, (WSDM: A user-centered design method for Web sites) presentado por De Troyer (1997), este método toma muchos de los conceptos previos de diseño de documentos hipermedia, y adiciona

un concepto clave para los sitios Web, la audiencia, definiéndola como la clase de los usuarios que determina los requerimientos del sitio Web.

- Técnicas de Diseño de Hipermedia Estructurado (Structured Hypermedia Design Technique (SHDT)) presentada por Bichler & Nusser (1996), es una técnica de modelado y un ambiente de diseño basado en la computadora, útil para facilitar la creación de prototipos, considera las páginas estáticas y dinámicas.

En un área de estudio como la de multimedia, que se caracteriza por conocer a priori el sistema de visualización, e incluso, se llegan a prefijar estándares sobre los equipos, no se puede pretender que sus resultados sean extrapolados directamente a otra área de estudio como la Web, que se caracteriza por la multiplicidad de los sistemas de visualización. Por esta razón, es conveniente analizar e identificar las características de las imágenes en las diferentes estructuras y niveles de un sitio Web.

Se deben identificar las características de una imagen (gráfica, auditiva, fotografía y el vídeo) para el reforzamiento de la función del apartado del sitio Web que la contiene, y así apoyar en el refuerzo del mensaje, evitando ser un elemento contradictorio del mismo.

b) Diseño. Enfoque de detalle

Un enfoque detallado en el diseño de sitios Web esta más ligado a las características propias de sus elementos, y en cómo éstos pueden integrarse para la construcción del mensaje.

A continuación se presenta una lista de herramientas y enfoques que tratan de orientamos en la creación de la página Web:

- Herramientas de autor presentan un ambiente integrado para la creación de páginas Web y ofrecen plantillas de diseño, sin embargo, no dan ninguna orientación sobre cómo componer, y equilibrar los pesos de los elementos que la integran, y más aun, si se sigue algunas plantillas de diseño se puede llegar a serias contradicciones de los lineamientos de usabilidad. Un ejemplo de este tipo de herramienta lo representa el software Director de la casa Macromedia.
- Marcos (frames). Es una herramienta de detalle para crear páginas Web, que en sus inicios prometió resolver los problemas de estructuración y pretendió dar un cuerpo sólido a un sitio Web, su ventaja consiste en que las páginas de contenido se desarrollan independiente de las de navegación, el resultado final, un estruendoso fracaso, viola muchos principios de usabilidad y lamentablemente por su éxito inicial existen muchos sitios que utilizan los marcos y rompen con los “criterios tradicionales” de la Web.
- Hojas de Estilo en Cascada (CCS) son herramientas que permiten administrar los estilos de presentación de un sitio Web, tanto al diseñador como al usuario final, su uso garantiza la consistencia, sin embargo, no todos los navegadores lo soportan. Antes de la introducción de las hojas de estilo, los creadores de páginas Web sólo tenían un control parcial sobre el aspecto final de sus páginas. Las hojas de estilo permiten un mayor control sobre el aspecto de los documentos. Con ellas se pueden especificar muchos atributos, tales como: colores, márgenes, alineación de elementos, tipos y tamaños de letras y muchos más. Es posible utilizar bordes para hacer que ciertos elementos resalten del resto de un documento. También ayudan a especificar diferentes fuentes para párrafos, títulos, subtítulos, entre otros.
- Además, puede emplearse hojas de estilo como patrones o páginas maestras, de forma tal, que múltiples páginas puedan tener el mismo aspecto.
- Lenguaje Scripting Son lenguajes de programación donde se hacen pequeñas rutinas para ser ejecutadas de lado del cliente y así dar más poder de dinamismo e

interactividad a las páginas Web. Permiten empotrar subprogramas en los documentos Web utilizando un lenguaje de programación (por ejemplo JavaScript) o permiten cargar y ejecutar applets basados en Java, independientemente de la plataforma utilizada para la comunicación en Internet. Durante su ejecución puede generar contenido dinámico, interactuar con el usuario, validar datos de formularios, o incluso, crear ventanas y ejecutar aplicaciones independientes de las páginas. Estos programas sobrepasan el modelo HTML original. Los applets son una desviación del modelo básico de la Web, sólo debemos recordar que los servidores realizaban casi todo el trabajo y en el lado del cliente sólo la visualización, los lenguajes scripting cambiaron esta filosofía, otorgándole al cliente la posibilidad de ejecutar programas. Los applets también permiten extender las características de los navegadores de los dientes sin obligar a los usuarios a cambiar de navegador. La ventaja de los applets es que a través de ellos se puede proporcionar interfaces de usuario más ricas a la Web. El inconveniente es que un programa del lado del cliente utiliza recursos de la computadora, es decir, aumenta la carga de trabajo en éste. Además el código también "pesa" (aumenta el tamaño de la página) lo que hace necesario un ancho de banda mayor, y si la aplicación requiere datos del servidor, también incrementa el peso en el sitio Web.

Es indudable, que la funcionalidad didáctica de una imagen como elemento de una página Web debe considerarse al diseñar los entornos educativos, así como su ubicación dentro del sitio Web y dentro de la misma página ya que dependiendo de su lugar se le otorga cierto grado de potencialidad visual.

I.10 Metodologías ágiles para el desarrollo de aplicaciones Web

La aparición de aplicaciones y sitios Web proporciona la explotación de otros mercados y servicios antes impensables como el comercio electrónico, la enseñanza virtual, etc., y esto conlleva un importante crecimiento en el desarrollo del software sobre dicha tecnología. Ahora bien, desde el punto de vista de la ingeniería del software es importante dotar de los mecanismos adecuados, para que la realización de este tipo de aplicaciones satisfaga las necesidades tanto de los usuarios como de los clientes que contratan el desarrollo de este tipo de aplicaciones. Pero actualmente no existe una metodología universalmente aceptada, que guíe en el proceso de desarrollo de aplicaciones Web, según lo expresan Cáceres y Marcos (2003) en su artículo "Procesos ágiles para el desarrollo de aplicaciones Web".

Existen criterios universalmente aceptados acerca del desarrollo software. Por ejemplo, y según afirman Jacobson y otros, el modelo de proceso más adecuado para el desarrollo de software es un proceso iterativo e incremental, puesto que a diferencia de otros modelos de proceso, como por ejemplo el modelo en cascada, permite la obtención de diversas versiones del producto software antes de la entrega final del mismo y la depuración y validación progresiva del mismo, lo que sin duda redundará en un software más satisfactorio para usuarios y cliente. Además y según indica Conallen (1999), con este tipo de proceso es posible añadir o modificar requisitos que no han sido detectados con anterioridad. Aún no existe ninguna propuesta universalmente aceptada para el desarrollo Web, pero

Fraternali explica que una posible solución al desarrollo adecuado de aplicaciones Web, sería combinar los ciclos de vida tradicionales con las propuestas de diseño para el desarrollo de aplicaciones hipermedia. De hecho, algunos de los trabajos existentes, relacionados con la tecnología hipermedia y Web, combinan el tratamiento de esas características especiales, con el uso de un modelo de proceso iterativo e incremental. En cualquier caso los métodos clásicos no son adecuados para el desarrollo de aplicaciones Web, puesto que no contemplen determinadas características específicas de este tipo de aplicaciones. Por otra parte, las metodologías tradicionales generalmente imponen un proceso de desarrollo demasiado pesado y burocrático

según afirma Fowler, lo que impide un desarrollo ágil y rápido para este tipo de aplicaciones. Como reacción a estas metodologías clásicas, recientemente ha aparecido un nuevo paradigma de ciclo de vida del software. Son las metodologías y procesos de desarrollo denominados ágiles, que garantizan un proceso de desarrollo suficiente pero no excesivo.

a) Las metodologías tradicionales versus las metodologías ágiles

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el objetivo de conseguir un software más eficiente y predecible. Para ello, se hace un especial hincapié en la planificación total de todo el trabajo a realizar y una vez que esta todo detallado, comienza el ciclo de desarrollo del producto software. Este planteamiento está basado en el resto de disciplinas de ingeniería, a pesar de que el software no pueda considerarse como la construcción de una obra clásica de ingeniería.

Con estas metodologías se lleva trabajando desde hace tiempo y no ha habido en ningún caso ninguna experiencia traumática acerca de su uso. Pero aún así, han recibido diversas críticas, y la más común hace referencia a su carácter excesivamente burocrático, y como afirma Fowler, este hecho ha llevado a identificarlas como metodologías monumentales.

Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.

En contraposición a estas metodologías clásicas, en los últimos años ha aparecido un nuevo grupo de metodologías, que se identifica como metodologías ágiles. Aportan como novedad, nuevos métodos de trabajo que apuestan por una cantidad apropiada de proceso. Es decir, ni se pierden en una excesiva cantidad de cuestiones burocráticas ni defienden tampoco la falta total de proceso. Buscan el equilibrio en la relación proceso/esfuerzo. Las diferencias existentes entre ambos grupos de metodologías surgen por un enfoque y objetivos diferentes. Y como principales diferencias, Fowler identifica las siguientes:

- Las metodologías ágiles son adaptativas más que predictivas. Las metodologías tradicionales potencian la planificación detallada de prácticamente todo el desarrollo software a largo plazo. Pero cuando se produce un cambio, toda esta planificación puede venirse abajo. Sin embargo, las metodologías ágiles proponen procesos que se adaptan y progresan con el cambio, llegando incluso hasta el punto de cambiar ellos mismos.
- Las metodologías ágiles están orientadas al personal más que orientadas al proceso. Intentan trabajar con la naturaleza del personal asignado al desarrollo, mas que contra ellos, de tal forma que permiten que la actividad de desarrollo software se convierta en una actividad grata e interesante.

Existen diversas metodologías que coinciden en llamarse metodologías ágiles. Y aunque entre ellas comparten muchas características tienen también diferencias significativas. A continuación se presentan algunas de las metodologías ágiles más representativas.

- Extreme Programming (XP) - La programación extrema, formulada por Kent Beck, concede una gran importancia a las pruebas del software (testing). Aunque la mayoría de los procesos las tienen en cuenta, generalmente lo contemplan de una forma demasiado ligera y superficial. Sin embargo, XP lo toma como base para el desarrollo y cada programador que escribe código también escribe los casos de prueba. Estos forman parte del proceso continuo de generación de código y se integra continuamente con ello, lo que garantiza una plataforma estable para el

futuro desarrollo. Sobre dicha plataforma se genera un proceso de diseño evolutivo, que es la base del sistema y que se enriquece con cada iteración. Nunca se generan diseños futuros. Según afirma Fowler, el resultado es un proceso de diseño que combina adecuadamente la disciplina con la adaptabilidad.

- Open source - Open source apuesta por la distribución de trabajo entre diferentes equipos, al igual que ocurre con la mayoría de los procesos adaptativos. La mayoría de los proyectos open source cuentan con supervisores de código. Estos supervisores de código, son las únicas personas autorizadas para realizar un cambio en el repositorio del código fuente. Por otra parte, el resto del personal puede realizar cualquier cambio en el código base. Sin embargo, el supervisor del código es la persona responsable de coordinar y de mantener la consistencia del diseño del software. Una de las principales ventajas de los desarrollos open source es que la depuración es altamente paralelizable, aunque un gran número de personas puedan verse involucradas. Cuando se soluciona un error, se envía la solución al supervisor de código, lo que garantiza que alguien realiza la modificación de forma fiable mientras otra parte del personal se dedica a las tareas de depuración.

b) Desarrollo de aplicaciones web con procesos ágiles

Lamentablemente en la actualidad, debido generalmente al exceso de burocracia de las metodologías tradicionales se ha pasado, en la mayoría de los casos, a no utilizar ningún método de trabajo específico y a trabajar “a destajo” con el único y erróneo objetivo de ahorrar tiempo y dinero. Aplicar cierto grado de disciplina ayudará en el proceso de desarrollo y siempre es mejor utilizar un proceso ágil que ninguna otra cosa. Sin embargo, un proceso ágil es adecuado para el desarrollo de cierto tipo de aplicaciones, por lo que no se deberían utilizar estos métodos para cualquier tipo de desarrollo software.

Los procesos ágiles son una buena elección cuando se trabaja con requisitos desconocidos o variables. Si no existen requisitos estables, no existe una gran posibilidad de tener un diseño estable y de seguir un proceso totalmente planificado, que no vaya a variar ni en tiempo ni en dinero. En estas situaciones, un proceso adaptativo será mucho más efectivo que un proceso predictivo. Por otra parte, los procesos de desarrollo adaptativos también facilitan la generación rápida de prototipos y de versiones previos a la entrega final, lo cual agradará al cliente. Pero la mayor barrera que habrá que salvar será convencer al cliente de que no existen una planificación y una forma fija de hacer las cosas. En cualquier caso, lo que se garantiza es un menor riesgo ante la posibilidad de cambios en los requisitos. Porque los cambios existen, y los procesos adaptativos permitirán estos cambios lo que en definitiva, garantizará que el producto final sea el deseado por el cliente. Según afirma Booch, todas estas razones son las que hacen que los procesos indicados por las comunidades de Extreme Programming y de Open Source hayan suscitado tanto interés.

En general, las aplicaciones Web cumplen la mayor parte de las características mencionadas en el párrafo anterior, por lo que la utilización de procesos ágiles podría ser beneficiosa para este tipo de desarrollos. La necesidad del cliente que contrata un desarrollo Web es que su producto esté disponible en la red lo más pronto posible. Si no se contempla esta necesidad, la aplicación no resultará un producto satisfactorio para el cliente. Puesto que los procesos ágiles permiten obtener versiones de producto previas a la versión final, si se aplican adecuadamente estos procesos, el cliente podrá disponer de forma rápida de alguna versión intermedia. Además el ciclo de desarrollo de la mayoría de los sitios y aplicaciones Web es extremadamente corto. Esto implica que generalmente no se aplique ningún tipo de proceso, pero sin duda y como se mencionaba anteriormente más vale un proceso ágil que nada. Por otra parte, los

desarrollos Web se perciben como desarrollos sencillos y los desarrolladores son sometidos a una gran presión de trabajo para terminar lo más pronto posible. Esta forma de trabajar va a implicar sin duda alguna modificaciones. Luego sería conveniente garantizar un proceso de desarrollo adaptable a los cambios. Otra cuestión fundamental a tener en cuenta es que las aplicaciones Web se desarrollan sin conocer los perfiles de los usuarios finales de las mismas, o lo que es lo mismo sin conocer los requisitos de usuario del sistema. Sin lugar a dudas esto implicará cambios en los requisitos inicialmente detectados, lo que nos lleva de nuevo a la elección de un proceso adaptativo. Por lo tanto, podríamos concluir que este tipo de procesos son especialmente aplicables al desarrollo de aplicaciones para la Web.

En este sentido, en el proyecto MIDAS presentado por Cáceres y Marcos, se está trabajando en la definición de una metodología para el desarrollo de aplicaciones Web cuyo modelo de proceso sea iterativo, incremental y adaptativo.

Este modelo ha sido definido en base a las características de aplicaciones Web y de las nuevas tendencias sobre procesos ágiles. Se ha propuesto un proceso iterativo para garantizar la realimentación de información y de requisitos una vez iniciado el desarrollo, así como la validación continua del sistema de información Web.

Esto permite que cada iteración contemple ciclos de desarrollo completos y cortos, y obtener así rápidamente, una nueva versión con mejoras sobre las versiones anteriores. Se ha propuesto un proceso incremental con la finalidad de obtener el sistema final tras la realización de diferentes etapas. Cada etapa está ligada a la realización de un determinado tipo de tareas.

El final de cada etapa proporciona además, una versión estable del software, que gracias a esas tareas concretas, cuenta con unas características específicas. Esto permite entregas al cliente de forma rápida y ágil. El proceso es además adaptativo lo que permite que pueda ser aplicado a diferentes aplicaciones Web. De esta forma, cada tipo de aplicación conllevará la ejecución de un determinado conjunto de tareas y pasará por unas determinadas etapas.

ANEXO II -Tratamiento de requisitos en propuestas para la Web. Un estudio comparativo (de Escalona y Koch: Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo)

II.1 Técnicas de Evaluación de requisitos

El desarrollo de sistemas de aplicaciones web, agrupa una serie de características que lo hacen diferente del desarrollo de otros sistemas Para facilitar la comprensión de las propuestas, antes de presentarlas, presentamos una clasificación de requisitos relevantes en sistemas Web.

Sin embargo, la terminología usada no es siempre la misma. Para facilitar la comprensión de las propuestas, antes de presentarlas, presentamos una clasificación de requisitos relevantes en sistemas Web.

- Requisitos de datos, también denominados requisitos de contenido, requisitos conceptuales o requisitos de almacenamiento de información. Éstos requisitos responden a la pregunta de qué información debe almacenar y administrar el sistema.
- Requisitos de interfaz (al usuario), también llamados en algunas propuestas requisitos de interacción o de usuario. Responden a la pregunta de cómo va a interactuar el usuario con el sistema.
- Requisitos navegacionales, recogen las necesidades de navegación del usuario.
- Requisitos de personalización, describen cómo debe adaptarse el sistema en función de qué usuario interactúe con él y de la descripción actual de dicho usuario.
- Requisitos transaccionales o funcionales internos, recogen qué debe hacer el sistema de forma interna, sin incluir aspectos de interfaz o interacción. También son conocidos en el ambiente web como requisitos de servicios.
- Requisitos no funcionales, son por ejemplo los requisitos de portabilidad, de reutilización, de entorno de desarrollo, de usabilidad, de disponibilidad, etc.

En este apartado se van a presentar aquellas técnicas que incluyen el proceso de definición de requisitos dentro del ciclo de vida. Alguna de las propuestas que describimos cubría la captura y definición de requisitos desde sus primeras propuestas. Otras, como se verán, la han incluido en revisiones a sus trabajos iniciales. El criterio que se ha seguido para presentar las propuestas es el cronológico, desde la primera publicación que incluye tratamiento de requisitos. Ello nos permitirá en la comparativa evaluar la evolución de las propuestas para requisitos.

a) WSDM: Web Site Design Method

WSDM es una propuesta para el desarrollo de sitios web, en la que el sistema se define en base a los grupos de usuarios. Su proceso de desarrollo se divide en cuatro fases: modelo de usuario, diseño conceptual, diseño de la implementación e implementación. La fase que más repercusión tiene para este trabajo es la primera en la que intenta detectar los perfiles de usuarios para los cuales se construye la aplicación. Para ello, se deben realizar dos tareas:

- Clasificación de usuarios: en este paso se deben identificar y clasificar a los usuarios que van a hacer uso del sistema.
- Para ello, WSDM propone el estudio del entorno de la organización donde se vaya a implantar el sistema y los procesos que se vayan a generar, describiendo las relaciones entre usuarios y actividades que realizan estos usuarios.

- Para la representación gráfica de estas relaciones WSDM propone una especie de mapas de conceptos de roles y actividades.
- Descripción de los grupos de usuarios: en esta segunda etapa se describen con más detalles los grupos de usuarios detectados en la etapa anterior. Para ello, se debe elaborar un diccionario de datos, en principio con formato libre, en el que indican los requisitos de almacenamiento de información, requisitos funcionales y de seguridad para cada grupo de usuarios.

El resto de las fases del proceso de WSDM se hacen en base a la clasificación de usuarios que se realiza en esta primera etapa.

b) SOHDM: Scenario-based Object-Oriented Hypermedia Design Methodology

Esta propuesta presenta la necesidad de disponer de un proceso que permita capturar las necesidades del sistema.

Para ello, propone el uso de escenarios. El proceso de definición de requisitos parte de la realización de un diagrama de contexto tal y como se propone en diagramas de flujos de datos (DFD) de Yourdon. En este diagrama de contexto se identifican las entidades externas que se comunican con el sistema, así como los eventos que provocan esa comunicación. La lista de eventos es una tabla que indica en qué eventos puede participar cada entidad. Por cada evento diferente SOHDM propone elaborar un escenario. Éstos son representados gráficamente mediante los denominados SACs2 (Scenario Activity Chart). Cada escenario describe el proceso de interacción entre el usuario y el sistema cuando se produce un evento determinado especificando el flujo de actividades, los objetos involucrados y las transacciones realizadas. SOHDM propone un proceso para conseguir a partir de estos escenarios el modelo conceptual del sistema que es representado mediante un diagrama de clases. El proceso de SOHDM continúa reagrupando estas clases para conseguir un modelo de clases navegacionales del sistema.

c) RNA: Relationship-Navegational Analysis

RNA plantea una secuencia de pasos centrándose fundamentalmente en el flujo de trabajo de análisis. El proceso de trabajo que presenta RNA se basa en la realización de las siguientes fases:

- Fase 1- Análisis del entorno: El propósito de esta fase es el de estudiar las características de la audiencia. Consiste en determinar y clasificar a los usuarios finales de la aplicación en grupos según sus perfiles.
- Fase 2- Elementos de interés: En esta fase se listan todos los elementos de interés de la aplicación. Por elementos de interés se entienden los documentos, las pantallas que se van a requerir, la información, etc.
- Fase 3- Análisis del conocimiento: Esta fase consiste en desarrollar un esquema que represente a la aplicación. Para ello RNA propone identificar los objetos, los procesos y las operaciones que se van a poder realizar en la aplicación, así como las relaciones que se producen entre estos elementos.
- Fase 4- Análisis de la navegación: En esta fase el esquema obtenido en la fase anterior es enriquecido con las posibilidades de navegación dentro de la aplicación.
- Fase 5- Implementación del análisis: Una vez obtenido el esquema final en el que ya se encuentran incluidos los aspectos de navegación, se pasa el esquema a un lenguaje entendible por la máquina.

La propuesta de RNA es quizás una de las que más ha resaltado la necesidad de trabajar con la especificación de requisitos, incluyendo tareas como el análisis del

entorno y de los elementos de interés. Además resulta interesante pues plantea la necesidad de analizar los requisitos conceptuales de manera independiente a los navegacionales.

d) HFPM: Hypermedia Flexible Process Modeling

La propuesta de HFPM describe un proceso detallado que cubre todo el ciclo de vida de un proyecto software.

HFPM propone un total de trece fases para las cuales se especifican a su vez una serie de tareas. Para este estudio es principalmente relevante la primera fase denominada de modelado de requisitos cuyas tareas son las siguientes:

- Descripción breve del problema. No indica ninguna técnica concreta pudiendo realizarse esta descripción mediante el lenguaje natural.
- Descripción de los requisitos funcionales mediante casos de uso.
- Realizar un modelo de datos para esos casos de uso, proponiendo el uso de un modelo de clases.
- Modelar la interfaz de usuario. Para ello, propone el uso de sketches y prototipos que permitan presentar los datos al usuario.
- Modelar los requisitos no funcionales. En éstos incluyen la navegación, la seguridad, etc.

Se puede observar que la propuesta de HFPM ofrece mayor detalle a la hora de realizar el tratamiento de los requisitos. Sin embargo, no ofrece técnicas concretas, especialmente a la hora de trabajar con los requisitos no funcionales.

e) OOHDM: Object Oriented Hypermedia Design Model

OOHDM es una propuesta metodológica ampliamente aceptada para el desarrollo de aplicaciones de la web. En sus comienzos no contemplaba la fase de captura y definición de requisitos, pero actualmente propone el uso de user interaction diagrams (UIDs). Esta propuesta parte de los casos de uso que considera una técnica muy difundida, ampliamente aceptada y fácilmente entendible por los usuarios y clientes no expertos, pero que resulta ambigua para el equipo de desarrollo en fases posteriores del ciclo de vida. Igualmente, resalta la necesidad de empezar el diseño del sistema, especialmente en los entornos web, teniendo un claro y amplio conocimiento de las necesidades de interacción, o lo que es lo mismo de la forma en la que el usuario va a comunicarse con el sistema.

Partiendo de estas dos premisas, OOHDM propone que la comunicación con el usuario se realice utilizando los casos de uso y a partir de ellos los analistas elaboran los UIDs. Estos UIDs son modelos gráficos que representan la interacción entre el usuario y el sistema, sin considerar aspectos específicos de la interfaz. El proceso de transformación de un caso de uso a un UIDs es descripto detalladamente en la propuesta, y se basa en detectar la interacción necesaria para la realización del caso de uso. OOHDM centra el desarrollo de un sistema de información web en el entorno del modelo conceptual de clases. Este diagrama debe surgir de los requisitos que se definan del sistema, pero los casos de uso resultan demasiado ambiguos para ello. Así, propone refinar el proceso de desarrollo descripto en UML de forma que de los casos de uso se generen los UIDs que concreticen más la definición de los requisitos para, a partir de ellos.

f) UWE: UML-Based Web Engineering

UML-Based Web Engineering (UWE) es una propuesta metodológica basada en el

Proceso Unificado y UML para el desarrollo de aplicaciones web. UWE cubre todo el ciclo de vida de este tipo de aplicaciones centrando además su atención en aplicaciones personalizadas (adaptivas). Para este trabajo, nos interesa principalmente analizar la propuesta de captura de requisitos de UWE. Esta metodología distingue entre la tarea de elicitar requisitos, definir y validar los requisitos. El resultado final de la captura de requisitos en UWE es un modelo de casos de uso acompañado de documentación que describe los usuarios del sistema, las reglas de adaptación, los casos de uso y la interfaz.

UWE clasifica los requisitos en dos grandes grupos: funcionales y no funcionales. Los requisitos funcionales tratados por UWE son:

- requisitos relacionados con el contenido
- requisitos relacionados con la estructura
- requisitos relacionados con la presentación
- requisitos relacionados con la adaptación
- requisitos relacionados con los usuarios

Además, UWE propone como técnicas apropiadas para la captura de los requisitos de un sistema web las entrevistas, los cuestionarios y los checklists y los casos de uso, los escenarios y el glosario para la definición de requisitos. Para la validación propone walk-throughs, auditorías y prototipos.

g) W2000

W2000 supone una propuesta que amplía la notación de UML con conceptos para modelar elementos de multimedia heredados de la propuesta HDM (Hypermedia Design Model). El proceso de desarrollo de W2000 se divide en tres etapas: análisis de requisitos, diseño de hipermedia y diseño funcional. El primero de ellos es el que resulta interesante para este trabajo. La especificación de requisitos en W2000 se divide en dos subactividades: análisis de requisitos funcionales y análisis de requisitos navegacionales. La especificación de requisitos comienza haciendo un estudio de los diferentes roles de usuario que van a interactuar con el sistema. Cada actor potencialmente distinto tendrá su modelo de requisitos de navegación y de requisitos funcionales. El modelo de requisitos funcionales es representado como un modelo de casos de uso tal y como se propone en UML. En él se representa la funcionalidad principal asociada a cada rol y las interacciones que se producen entre el sistema y cada rol. El segundo modelo consiste en otro diagrama de casos de uso pero que no representa funcionalidad sino posibilidades de navegación de cada actor. La representación gráfica es realizada con una extensión de UML.

h) UWA: Ubiquituos Web Applications

UWA ha nacido de la colaboración entre diferentes grupos de trabajo, por lo que resulta realmente una agrupación de propuestas y técnicas. En concreto, la propuesta de W2000 se encuentra incluida en UWA. Sin embargo, W2000 ha sido incluida en UWA sólo en la fase de diseño hipermedia, siendo ambas propuestas diferentes en la fase de definición de requisitos. Por esta razón han sido incluidos en este trabajo en forma separada. El proceso de captura de requisitos en UWA comienza definiendo los diferentes roles de usuario que pueden interactuar con el sistema, los objetivos globales del sistemas y la relación entre éstos. . El proceso continúa haciendo un refinamiento de esos objetivos globales, concretándolos en subobjetivos. Estos subobjetivos son estudiados y refinados para detectar conflictos entre ellos. De esta forma, se concretizan aún más dividiéndolos en requisitos. Los requisitos son clasificados en varios tipos: de contenido, de estructura de contenido, de acceso, de

navegación, de presentación, de operaciones de usuario y de operaciones del sistema.

De esta forma, los requisitos se van refinando hasta que solo pertenezcan a uno de estos grupos. Y finalmente los requerimientos son asignados a artefactos de diseño o a reglas de customización.

Para definir los objetivos, UWA propone una notación propia, basada en una plantilla. La definición de los actores y la relación con los objetivos se hace usando un diagrama basado en casos de uso. Por último, para definir y refinar los subobjetivos y los requisitos, utilizan una notación gráfica propia que denominan grafo de refinamiento de objetivos, el refinamiento de este grafo permite ir representando la relación entre los requisitos y hacer un seguimiento para validar la consecución de los objetivos del sistema. Una vez que los requisitos son detectados, hacen uso de XML para definirlos de una manera formal.

i) NDT - Navigational Development Techniques

NDT (Navigational Development Techniques) es una técnica para especificar, analizar y diseñar el aspecto de la navegación en aplicaciones web. Para este trabajo, solo es relevante la propuesta que ofrece para la definición y captura de requisitos. El flujo de especificación de requisitos de NDT comienza con la fase de captura de requisitos y estudio del entorno. Para ello, plantea el uso de técnicas como las entrevistas o el brainstorming y JAD. Tras esta fase, se propone la definición de los objetivos del sistema. En base a estos objetivos, el proceso continúa definiendo los requisitos que el sistema debe cumplir para cubrir los objetivos marcados. NDT clasifica los requisitos en:

- Requisitos de almacenamiento de información
- Requisitos de actores
- Requisitos funcionales
- Requisitos de interacción, representados mediante:
    - Frases, que recogen cómo se va a recuperar la información del sistema utilizando un lenguaje especial denominado BNL (Bounded Natural Language).
    - Prototipos de visualización, que representan la navegación del sistema, la visualización de los datos y la interacción con el usuario.
- Requisitos no funcionales

Todo el proceso de definición y captura de requisitos y objetivos que propone NDT se basa principalmente en plantillas o patrones, pero también hace uso de otras técnicas de definición de requisitos como son los casos de uso y los glosarios. La propuesta ofrece una plantilla para cada tipo de requisito, lo que permite describir los requisitos y objetivos de una forma estructurada. Algunos de los campos de los patrones son cerrados, es decir, solo pueden tomar valores predeterminados. Estos campos permiten que en el resto del proceso del ciclo de vida de NDT se puedan conseguir resultados de forma sistemática. El flujo de trabajo de especificación de requisitos termina proponiendo la revisión del catálogo de requisitos y el desarrollo de una matriz de trazabilidad que permite evaluar si todos los objetivos han sido cubiertos en la especificación. La propuesta viene acompañada de una herramienta case, NDT-Tool, que facilita la cumplimentación de los patrones.

II.2 Comparativa de las propuestas

En base a la clasificación de requisitos que se realizaron al principio del documento, la primera comparativa que se va a realizar de las propuestas estudiadas

consiste en ver qué tipos de requisitos contempla cada propuesta. En la tabla 1 se presentan los diferentes requisitos y se indica cuáles de ellos son tratados en cada metodología.

Analizando los resultados, y teniendo en cuenta que las propuestas están ordenadas por orden cronológico, resulta interesante observar el avance que han tenido los requisitos en el entorno de la web. Las primeras propuestas estaban más centradas en los requisitos de datos y los de interfaz de usuario. Observando la tabla puede verse que las propuestas más actuales resaltan la necesidad de tratar los requisitos de personalización y navegación, así como los transaccionales de forma independiente. Esta idea de separación de conceptos se observó desde el principio en las propuestas para la web (HDM, EORM, OOHDM, etc.) pero solo se planteaba en fases avanzadas del proceso de desarrollo, principalmente en diseño. Se puede observar la tendencia actual de partir de esta separación de concepto ya en la fase de especificación de requisitos.

Tabla 29 - Tipos de requisitos contemplados en cada propuesta

	Req. datos	Req. interfaz al usuario	Req. navegacionales	Req. personalización	Req. transacionales	Req. no funcionales
WSDM	✓			✓		✓
SOHDM	✓	✓			✓	
RNA	✓	✓	✓		✓	
HFPM	✓	✓	✓			✓
OOHDM	✓	✓	✓			
UWE	✓	✓	✓	✓		✓
W2000			✓	✓	✓	
UWA	✓	✓	✓	✓	✓	
NDT	✓	✓	✓	✓	✓	✓
DDDP	✓	✓	✓	✓	✓	✓

Analizando los resultados, y teniendo en cuenta que las propuestas están ordenadas por orden cronológico, resulta interesante observar el avance que han tenido los requisitos en el entorno de la web. Las primeras propuestas estaban más centradas en los requisitos de datos y los de interfaz de usuario. Observando la tabla puede verse que las propuestas más actuales resaltan la necesidad de tratar los requisitos de personalización y navegación, así como los transaccionales de forma independiente. Esta idea de separación de conceptos se observó desde el principio en las propuestas para la web (HDM, EORM, OOHDM, etc.) pero solo se planteaba en fases avanzadas del proceso de desarrollo, principalmente en diseño. Se puede observar la tendencia actual de partir de esta separación de concepto ya en la fase de especificación de requisitos.

También resulta necesario resaltar, aunque no pueda derivarse directamente de la tabla, el hecho de la multiplicidad de términos. La mayoría de las propuestas trabajan con los mismos conceptos pero la terminología usada para nombrarlos es diferente, lo que hace bastante complejo el comparar lo que cada una de ellas propone. Un esfuerzo interesante sería el de unificar la terminología.

Con respecto a la definición de requisitos, se puede observar que es el aspecto central del tratamiento de requisitos para todas las propuestas. Se puede concluir en que existe una tendencia a usar la técnica de casos de uso como base.

Otra conclusión muy relevante que se obtiene de este estudio es el hecho de la poca importancia que las propuestas han prestado a la validación de requisitos. Las técnicas de validación que proponen se basan principalmente en la revisión de los modelos y resultados de la definición de requisitos. La gran mayoría de las propuestas ni siquiera contemplan esta fase en su proceso de ingeniería de requisitos.

ANEXO III - El plan de pruebas

El estándar IEEE 829-1983 describe los tipos de documentos que pueden producirse durante el proceso de prueba:

Plan de prueba. Especificación de los requerimientos para el diseño de los casos de prueba

Suite de prueba

Caso de prueba

Descripción del procedimiento de prueba

Descripción del ítem a probar (IUT en Binder)

Bitácora de pruebas

Reporte de incidentes de prueba

Resumen de pruebas

El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de un proceso de pruebas. Nótese que puede haber un plan global que explicite el énfasis a realizar sobre los distintos tipos de pruebas (verificación, integración e integración).

Puede haber un plan global que explicite el énfasis a realizar sobre los distintos tipos de pruebas (verificación, integración e integración). Un plan de pruebas incluye:

1. Identificador del plan. Preferiblemente de alguna forma mnemónica que permita relacionarlo con su alcance, por ej. TP-Global (plan global del proceso de pruebas), TP-Req-Seguridad1 (plan de verificación del requerimiento 1 de seguridad), TP-Contr-X (plan de verificación del contrato asociado al evento de sistema X), TP-Unit-Despachador.iniciar (plan de prueba unitario para el método iniciar de la clase Despachador). Como todo artefacto del desarrollo, está sujeto a control de configuración, por lo que debe distinguirse adicionalmente la versión y fecha del plan.
2. Alcance. Indica el tipo de prueba y las propiedades/elementos del software a ser probado.
3. Ítems a probar. Indica la configuración a probar y las condiciones mínimas que debe cumplir para comenzar a aplicarle el plan. Por un lado, es difícil y riesgoso probar una configuración que aún reporta fallas; por otro lado, si esperamos a que todos los módulos estén perfectos, puede que detectemos fallas graves demasiado tarde.
4. Estrategia. Describe la técnica, patrón y/o herramientas a utilizarse en el diseño de los casos de prueba. Por ejemplo, en el caso de pruebas unitarias de un procedimiento, esta sección podría indicar: "Se aplicará la estrategia caja-negra de fronteras de la precondición" o "Ejercicio de los caminos ciclo máticos válidos". En lo posible la estrategia debe precisar el número mínimo de casos de prueba a diseñar, por ej. 100% de las fronteras, 60% de los caminos ciclomáticos. La estrategia también explicita el grado de automatización que se exigirá, tanto para la generación de casos de prueba como para su ejecución.
5. Categorización de la configuración. Explicita las condiciones bajo las cuales, el plan debe ser:
    - Suspendido,
    - Repetido;
    - Culminado.

En algunas circunstancias (las cuales deben ser explicitadas) el proceso de prueba debe suspenderse en vista de los defectos o fallas que se han detectado. Al corregirse los defectos, el proceso de prueba previsto por el plan puede continuar, pero debe explicitarse a partir de qué punto, ya que puede ser necesario repetir algunas pruebas. Los criterios de culminación pueden ser tan simples como aprobar el número mínimo de casos de prueba diseñados o tan complejos como tomar en cuenta no sólo el número mínimo, sino también el tiempo previsto para las pruebas y la tasa de detección de fallas.

6. Tangibles. Explicita los documentos a entregarse al culminar el proceso previsto por el plan. Por ejemplo, subplanes, especificación de pruebas, casos de prueba, resumen gerencial del proceso y bitácora de pruebas.
7. Procedimientos especiales Identifica el grafo de las tareas necesarias para preparar y ejecutar las pruebas, así como cualquier habilidad especial que se requiere.
8. Recursos. Especifica las propiedades necesarias y deseables del ambiente de prueba, incluyendo las características del hardware, el software de sistemas (p. ej. el sistema de operación), cualquier otro software necesario para llevar a cabo las pruebas, así como la colocación específica del software a probar (p. ej. qué módulos se colocan en qué máquinas de una red local) y la configuración del software de apoyo. La sección incluye un estimado de los recursos humanos necesarios para el proceso. También se indican cualquier requerimiento especial del proceso: actualización de licencias, espacio de oficina, tiempo en la máquina de producción, seguridad.
9. Calendario. Esta sección describe los hitos del proceso de prueba y el grafo de dependencia en el tiempo de las tareas a realizar.
10. Manejo de riesgos Explicita los riesgos del plan, las acciones mitigantes y de contingencia.
11. Responsables. Especifica quién es el responsable de cada una de las tareas previstas en el plan.

ANEXO IV – W3C link checker

En este Anexo se presenta el resultado total de la ejecución del W3C link checker sobre el sitio web del IUA, cuyo análisis se presentó en el capítulo 7.

Link Checker: http://www.iua.edu.ar

- SKIP NAVIGATION DOCS
- DOWNLOAD
- FEEDBACK
- VALIDATOR

Processing http://www.iua.edu.ar/home/default.asp

Settings used:

- Accept: application/xhtml+xml, text/html, */*;q=0.5
- Accept-Language: es-ar
- Sleeping 1 second between requests to each server

Go to the results.

For reliable link checking results, check HTML validity first. See also CSS validity.

Back to the link checker.

```
Parsing...
 done (465 lines in 0.0s).
Checking anchors...
 done.
Checking link http://www.iua.edu.ar/Common/images/contactenos.jpg
HEAD http://www.iua.edu.ar/Common/images/contactenos.jpg  fetched in
1.5s
Checking link http://www.iua.edu.ar/Common/images/logo4.jpg
HEAD http://www.iua.edu.ar/Common/images/logo4.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/top9.jpg
HEAD http://www.iua.edu.ar/Common/images/top9.jpg  fetched in 1.2s
Checking link javascript:MasInfo();
HEAD javascript:MasInfo();  fetched in 0.0s
Checking link http://www.iua.edu.ar/Common/images/flash6.jpg
HEAD http://www.iua.edu.ar/Common/images/flash6.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/investigacion
HEAD http://www.iua.edu.ar/investigacion
HEAD http://www.iua.edu.ar/investigacion/  fetched in 4.3s
Checking link http://www.iua.edu.ar/Common/images/top4.jpg
HEAD http://www.iua.edu.ar/Common/images/top4.jpg  fetched in 1.3s
Checking link http://www.iua.edu.ar/Common/images/top10.jpg
HEAD http://www.iua.edu.ar/Common/images/top10.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Distancia/default.asp
HEAD http://www.iua.edu.ar/Distancia/default.asp  fetched in 1.6s
Checking link http://www.iua.edu.ar/Common/images/top2.jpg
HEAD http://www.iua.edu.ar/Common/images/top2.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/flash3.jpg
HEAD http://www.iua.edu.ar/Common/images/flash3.jpg  fetched in 1.3s
Checking link http://www.iua.edu.ar/Common/images/top1.jpg
HEAD http://www.iua.edu.ar/Common/images/top1.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/servicios
HEAD http://www.iua.edu.ar/servicios
HEAD http://www.iua.edu.ar/servicios/  fetched in 2.8s
```

```
Checking link http://noticias.iua.edu.ar/article.php3?id_article=201
HEAD http://noticias.iua.edu.ar/article.php3?id_article=201  fetched
in 1.8s
Checking link http://www.iua.edu.ar/Common/images/logo3.jpg
HEAD http://www.iua.edu.ar/Common/images/logo3.jpg  fetched in 0.5s
Checking link http://www.iua.edu.ar/Common/images/top5.jpg
HEAD http://www.iua.edu.ar/Common/images/top5.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/noti-bottom1.jpg
HEAD http://www.iua.edu.ar/Common/images/noti-bottom1.jpg  fetched in
1.2s
Checking link http://www.iua.edu.ar/Common/images/top11.jpg
HEAD http://www.iua.edu.ar/Common/images/top11.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/flash7.jpg
HEAD http://www.iua.edu.ar/Common/images/flash7.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/top6.jpg
HEAD http://www.iua.edu.ar/Common/images/top6.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/noti-bottom2.jpg
HEAD http://www.iua.edu.ar/Common/images/noti-bottom2.jpg  fetched in
1.2s
Checking link http://www.iua.edu.ar/Common/images/buscar.jpg
HEAD http://www.iua.edu.ar/Common/images/buscar.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/logo5.jpg
HEAD http://www.iua.edu.ar/Common/images/logo5.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/flash5.jpg
HEAD http://www.iua.edu.ar/Common/images/flash5.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/texto2.jpg
HEAD http://www.iua.edu.ar/Common/images/texto2.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Hojas_de_estilos/contenido.css
HEAD http://www.iua.edu.ar/Hojas_de_estilos/contenido.css  fetched in
1.2s
Checking link http://www.iua.edu.ar/Common/images/flash4.jpg
HEAD http://www.iua.edu.ar/Common/images/flash4.jpg  fetched in 1.2s
Checking link http://noticias.iua.edu.ar/article.php3?id_article=200
HEAD http://noticias.iua.edu.ar/article.php3?id_article=200  fetched
in 0.4s
Checking link http://www.iua.edu.ar/Common/images/logo1.jpg
HEAD http://www.iua.edu.ar/Common/images/logo1.jpg  fetched in 0.5s
Checking link http://www.iua.edu.ar/Common/images/noti-bottom3.jpg
HEAD http://www.iua.edu.ar/Common/images/noti-bottom3.jpg  fetched in
1.2s
Checking link http://www.iua.edu.ar/Common/images/top7.jpg
HEAD http://www.iua.edu.ar/Common/images/top7.jpg  fetched in 1.3s
Checking link http://www.iua.edu.ar/Hojas_de_Estilos/noticia.css
HEAD http://www.iua.edu.ar/Hojas_de_Estilos/noticia.css  fetched in
1.2s
Checking link http://www.iua.edu.ar/home/noticia/noticia.js
HEAD http://www.iua.edu.ar/home/noticia/noticia.js  fetched in 1.2s
Checking link http://noticias.iua.edu.ar/article.php3?id_article=198
HEAD http://noticias.iua.edu.ar/article.php3?id_article=198  fetched
in 0.9s
Checking link http://www.iua.edu.ar/Common/images/top3.jpg
HEAD http://www.iua.edu.ar/Common/images/top3.jpg  fetched in 0.4s
Checking link http://www.iua.edu.ar/Ingenieria/default.asp
HEAD http://www.iua.edu.ar/Ingenieria/default.asp  fetched in 1.3s
Checking link http://www.iua.edu.ar/Common/images/flash8.jpg
HEAD http://www.iua.edu.ar/Common/images/flash8.jpg  fetched in 1.3s
Checking link http://www.iua.edu.ar/Common/images/logo2.jpg
HEAD http://www.iua.edu.ar/Common/images/logo2.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/top8.jpg
HEAD http://www.iua.edu.ar/Common/images/top8.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/noti-right.jpg
HEAD http://www.iua.edu.ar/Common/images/noti-right.jpg  fetched in
1.3s
Checking link http://www.iua.edu.ar/Common/images/noti-left.jpg
HEAD http://www.iua.edu.ar/Common/images/noti-left.jpg  fetched in
1.3s
Checking link http://www.iua.edu.ar/Common/images/flash2.jpg
HEAD http://www.iua.edu.ar/Common/images/flash2.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/noti-top.jpg
HEAD http://www.iua.edu.ar/Common/images/noti-top.jpg  fetched in 1.2s
```

```
Checking link http://www.iua.edu.ar/Common/images/separador1.jpg
HEAD http://www.iua.edu.ar/Common/images/separador1.jpg  fetched in
1.2s
Checking link http://www.iua.edu.ar/conocer
HEAD http://www.iua.edu.ar/conocer
HEAD http://www.iua.edu.ar/conocer/  fetched in 2.5s
Checking link http://www.iua.edu.ar/Common/images/flecha.jpg
HEAD http://www.iua.edu.ar/Common/images/flecha.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/vida/egresados.asp
HEAD http://www.iua.edu.ar/vida/egresados.asp  fetched in 1.4s
Checking link http://www.iua.edu.ar/Common/images/mapa.jpg
HEAD http://www.iua.edu.ar/Common/images/mapa.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/Common/images/flash1.jpg
HEAD http://www.iua.edu.ar/Common/images/flash1.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/contactenos
HEAD http://www.iua.edu.ar/contactenos
HEAD http://www.iua.edu.ar/contactenos/  fetched in 2.6s
Checking link http://www.iua.edu.ar/vida
HEAD http://www.iua.edu.ar/vida
HEAD http://www.iua.edu.ar/vida/  fetched in 2.5s
Checking link http://www.iua.edu.ar/estudiar
HEAD http://www.iua.edu.ar/estudiar
HEAD http://www.iua.edu.ar/estudiar/  fetched in 3.0s
Checking link http://www.iua.edu.ar/Common/images/mail.jpg
HEAD http://www.iua.edu.ar/Common/images/mail.jpg  fetched in 1.2s
Checking link http://www.iua.edu.ar/cread2004
HEAD http://www.iua.edu.ar/cread2004
HEAD http://www.iua.edu.ar/cread2004/  fetched in 2.7s
Processed in 79.4s.
```

Results

Anchors

Found 17 anchors.

Valid anchors!

List of broken links and redirects

Fragments listed are broken. See the table below to know what action to take.

Code	Occurrences	What to do
501	1	Could not check this link: method not implemented or scheme not supported.

javascript:MasInfo();

What to do: **You must change this link: people using a browser without JavaScript support will *not* be able to follow this link. See the Web Content Accessibility Guidelines on the use of scripting on the Web and the techniques on how to solve this.**

Response status code: 501

Response message: Protocol scheme 'javascript' is not supported

Line: 149

List of directory redirects

The links below are not broken, but the document does not use the exact URL.

http://www.iua.edu.ar/conocer redirected to
http://www.iua.edu.ar/conocer/
What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 80, 430

http://www.iua.edu.ar/investigacion redirected to
http://www.iua.edu.ar/investigacion/
What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 83, 433

http://www.iua.edu.ar/contactenos redirected to
http://www.iua.edu.ar/contactenos/
What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 85, 435

http://www.iua.edu.ar/vida redirected to
http://www.iua.edu.ar/vida/
What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 82, 432

http://www.iua.edu.ar/estudiar redirected to
http://www.iua.edu.ar/estudiar/
What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 81, 431

http://www.iua.edu.ar/servicios redirected to
http://www.iua.edu.ar/servicios/
What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Lines: 84, 434

http://www.iua.edu.ar/cread2004 redirected to
http://www.iua.edu.ar/cread2004/

What to do: **Add a trailing slash to the URL.**
Response status code: 301 -> 200
Response message: Moved Permanently -> OK
Line: 94

Checked 1 document in 79.4 seconds.

W3C Link Checker
version 4.2 (c) 1999-2005 W3C

REFERENCIAS BIBLIOGRÁFICAS

A QA Focus Document. (2005). *Use of automated tools for web sites accessibility.* Recuperado de: http://www.ukoln.ac.uk/qa-focus/documents/briefings/briefing-02/html/

Baeza-Yates, R.; Rivera Loaiza, C. (2003). *Ubicuidad y Usabilidad en la Web*. Departamento de Ciencias de la Computación, Universidad de Chile. Recuperado de: http://www.dcc.uchile.cl/%7Erbaeza/inf/usabilidad.html

Beizer B. (1995). *Black-Box Testing. Techniques for Functional Testing of software and systems*. New York. U:S:A. John Wiley & Sons.

Bertolino A. (2003). *Software Testing Research and Practice*. Electronic Edition (Springer Link).

Bisbal J., Lawless D., Wu B., Grimson J. (1999). *Legacy Information System Migration: A brief Review of Problems, Solutions and Research Issues* . Computer Science Department, Trinity College. Dublin.

Brodie M., Stonebraker, M. (1995). *Migrating legacy systems: Gateways, Interfaces and the Incremental Approach.* Morgan Kaufmann Publishers.

Cabero J. (1995). *Navegando, construyendo: la utilización de los hipertextos en la enseñanza***.** Universidad de Sevilla. Recuperado de: http://edutec.rediris.es/documentos/1996/hiper.html

Cáceres P., Marcos E. (2003). *Procesos Ágiles para el desarrollo de aplicaciones web*. Universidad Rey Juan Carlos. Madrid. Recuperado de: http://www.dlsi.ua.es/webe01/articulos/s112.pdf

Desphande Y., Murugesan S., Ginige A., Hansen S. (2002). *Web Engineering*. Journal of Web Engineering. Vol. 1, No. 1.

Escalona M.J. & Koch N. (2002). *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* Universidad de Sevilla. Sevilla*.* http://lsiweb.lsi.us.es/docs/informes/LSI-2002-4.pdf

Espiñeira, Sheldon y asociados (2005). *El problema de la migración.* Florida. U.S.A. Recuperado de: http:// www.pc-news.com/detalle.asp

Fernández Sanz L. et al. (2004). *Mejora de la calidad en desarrollos orientados a objetos utilizando especificaciones UML para la obtención y precedencia de casos de prueba.* Universidad de Madrid. Artículo publicado en Revista de Procesos y Métricas de la Tecnología de la Información.de la Asoc. Española de Sistemas Informáticos.

Glass Robert L. (2000). *Has Web Development Changed the Meaning of Testing?* Recuperado de: http://www.stickymind.com.

Harrold M. (2000). *Testing : A Roadmap*. College of Computing. Georgia Institute of Technology. Atlanta. Copyright ACM.

Hartman A., Kirskin K. & Nagin K. (2002). *A test execution environment running abstract test for distributed software.* Haifa University. Haifa, Israel. *From Proceeding* (374) Software Engineering and Applications - 2002.
Recuperado de: http://www.agedis.de/documents/p144-hartman.pdf

Hetzel B. (1998) *The complete guide to software testing*. Recuperado de: http://www.wileyeurope.com/cda/cover/0,,0471414352%7Cexcerpt,00.pdf

Jacobson, I., Booch, G. & Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software.* Madrid: Pearson.

Koch, N. & Kraus, A. (2002).*The expresive Power of UML-based Web Engineering*. Munich: Universidad de Munich.

Liem I., Wahyudin D., Schatten A. (2006). *Data Integration: an experience of Information System Migration.* Proceedings of the International Conference on Information Integration, Web-Applications and Services.

Linz T. , Daigl M. (1997/1998). *Capture and Replay GUI Testing Tools*. Results of the ESSI PIE 24306. Germany.

López O., Laguna, M. García, F. (2001). *Reutilización de Requisitos en el Modelo Mecano*. Valladolid: Universidad de Valladolid. Recuperado de: http://www.giro.infor.uva.es/Publications/2001/LGLM01/JIRA_Laguna.pdf

Memon A.T., Pollack M. & Soffa M.L. (2001). *Automated Test Oracles for GUIs. Regression Testing of GUIs*. Proceedings of the 8th ACM SIGSOFT international symposium on Foundations of software engineering: twenty-first century applications San Diego, California, U.S.A.

Mínguens Sanz D., García Morales E. (2003). *Metodologías para el desarrollo de aplicaciones Web: UWE*. Recuperado de: http://www.eici.ucm.cl/Academicos/ygomez/descargas/Ing_Sw2/apuntes/DASBD-Metodolog-ADasParaElDesarrolloDeaplicacionesWeb_UWE.pdf

Murugesan, S. et al. (2001). *Web engineering: A new discipline for development of Web-based systems.* 2001. Proceedings of the International Conference on Software Engineering. 2001. Los Angeles. U.S.A.

Myers, Glenford J. (1984). *El arte de probar software*. Buenos Aires: El Ateneo.

Musciano, Ch. & Kennedy, B. (2002). "*Html And Xhtml, The Definitive Guide"* , School & Library Binding .

Nguyen, H. (2003).*Testing aplication on the web*. Indiana. U.S.A:Wiley Publishing.

NIELSEN,J. (1990). *Hypertext and hypermedia*. Boston: Academic Press

Olsina, L. A. (1999). *Metodología Cuantitativa para la evaluación y comparación de la calidad de sitios Web* [Tesis Doctoral]. Universidad Nacional de La Plata. Facultad de Ciencias Exactas. Recuperada de: http://gidis.ing.unlpam.edu.ar/downloads/pdfs/Web-site_QEM_VF.pdf

Ortiz, Molina, Moros. *El Modelo del Negocio como base del Modelo de Requisitos*. Universidad de Murcia. Murcia.

Pinheiro da Silva P., Paton N. (2000). *User Interface Modelling with UML*. University of Manchester. Manchester.

Piattini, M., Calvo-Manzano, J., Cervera, J. & Fernández, L. (2004). *Análisis y Diseño de Aplicaciones Informáticas de Gestión*. México: Alfaomega.

Pressman, Roger (2003). *Ingeniería de Software. Un enfoque práctico.* Madrid. España: Mc.Graw Hill.

Ricca F.; Tonella, P. (2001). *Analysis and testing of web applications*. ICSE 2001

Rosaria S., Robinson H. (2001). *Applying Models in your Testing Process.* Intelligent Search Test Group. Microsoft Corporation.

Ryden Jesper. *"Web Application Testing, A Methodoly with Tools and Considerations"*

Sánchez Díaz J., Pastor López, O. (2001). *Generación automática de prototipos de interface de usuario a partir de modelos de requisitos*.

Schwabe, D. Rossi, G. *The Object-Oriented Hypermedia Design Model (OOHDM)*. (1996). *Proceedings of the ACM International Conference on Hipertext* (Hypertext'96). Washington DC, March, 16-20,

Silva, D., Mercerat, B. (2001). *Construyendo Aplicaciones Web con una Metodología de Diseño Orientada a Objetos*. Revista Colombiana de Computación.

Sommerville, Ian. (2005). *Ingeniería de Software.* Editorial Pearson Educación.

Spillner A. (2002). *The W-Model. Strengthening the Bond between Development and Test*. University of Applied Sciences Bremen. Germany.

Spool, J. (2005). *Common Questions & Answers About Usability Testing*. From User Interface Engineering. Andover. Canadá. Recuperado de: http://www.uie.com/articles/usability_testing_mistakes

Vallespir D. (2004). *Generación Automática de Casos de Prueba Unitarios para Objetos*. Universidad de la República. Montevideo.

Villa L. (2004). *El porqué de la migración desde el entorno host a web*. Recuperado de: http://www.grancomo.com/e/el_porque_de_la_migracion_desde_el_entorno_host_a_web.php

Visconti Z., Bidart F., Catherine & Mujica A., Jorge (2002). *Web Testing. Aspectos teóricos y prácticos*. Universidad Técnica Federico Santa María. Departamento de Informática. Santiago de Chile.
Recuperado de: http://www.inf.utfsm.cl/~visconti/testing/Documentos/WebTesting.pdf.

Wegner, Meter & Goldin, Dina (1999). *Models of interaction*. Brown University. Recuperado de: http://www.cs.brown.edu/people/pw/papers/ec99.pdf.

Wu, B., Lawless, D., Bisbal, J., Richardson, R., Grimson, J., Wade, V., O'Sullivan, D. (1997) *The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems*. Third IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '97), 1997

Yourdon, E. (1993). *Análisis Estructurado Moderno*. México: Prentice Hall.

Zúñiga J., Rossainz López M. (2002) *Introducción a la Ingeniería Web Basada en UML.*