UNIVERSIDAD CARLOS III

ESCUELA POLITÉCNICA SUPERIOR CARLOS III

Ingeniería Técnica en Informática de Gestión

PROYECTO FIN DE CARRERA

Las Pruebas de Integración como Proceso de la Calidad del Software en el Ámbito de las Telecomunicaciones

Nuria Gómez Rodríguez

Marzo/2015

UNIVERSIDAD CARLOS III

ESCUELA POLITÉCNICA SUPERIOR CARLOS III

Ingeniería Técnica en Informática de Gestión

PROYECTO FIN DE CARRERA

Las Pruebas de Integración como Proceso de la Calidad del Software en el Ámbito de las Telecomunicaciones

Tutor/Director: **Dolores Cuadra**
Autor: **Nuria Gómez Rodríguez**

Marzo/2015

ÍNDICE

1 Introducción 7
1.1 Objetivos 7
1.2 Motivación 8
1.2.1 La Necesidad de las Pruebas 8
1.2.2 Conceptos Básicos 9
2 Estado del Arte 13
2.1 Ciclo de Vida del Producto 13
2.2 Modelo Genérico en V 14
2.2.1 Pruebas de Componentes 16
2.2.2 Pruebas de Integración 17
2.2.3 Pruebas de Sistema 19
2.2.4 Pruebas de Aceptación 21
2.2.5 Otros Tipos de Pruebas 22
2.3 Principios Fundamentales de las Pruebas de Software 26
2.4 ¿Cuándo podemos decir que las pruebas realizadas han sido suficientes? 31
2.5 Criterios de Finalización 33
2.6 Diferencias entre Diseñar, Desarrollar y Probar 35
2.7 Personalidad de un Buen Probador 38
2.8 Importancia de los Requisitos 41
2.9 SQA 45
2.10 Estándares de Calidad de Software 46
3 Paradigmas de las Pruebas 48
3.1 Técnicas Dinámicas de Diseño de Pruebas 48

3.1.1 Pruebas de Caja Blanca 49
3.1.2 Pruebas de Caja Negra 52
3.1.3 Técnicas Basadas en la Experiencia 56
3.2 Ciclo de Vida de las Pruebas de Integración 57
3.2.1 Planificación 58
3.2.2 Análisis y Diseño 60
3.2.3 Implementación y Ejecución 61
3.2.4 Evaluación 63
3.2.5 Cierre 64
3.2.6 Monitorización y Control 65
3.2.7 Mejora Continua 66
3.3 Roger S. Pressman VS ISTQB 68
4 Caso de Estudio 70
4.1 Gestión de la Configuración 70
4.1.1 Middleware 71
4.1.2 Entornos de pruebas 77
4.1.3 Instalaciones 79
4.2 Herramientas de Prueba 83
4.2.1 Quality Center 84
4.2.2 PVCS 90
4.2.3 Jtrac Estimaciones 94
4.2.4 Jtrac Gestión Entornos 96
4.2.5 Bases de Datos 98
4.2.6 SecureCRT 99
4.2.7 Email 101

4.3 Ciclo de Vida del Caso de Estudio 101
4.3.1 Planificación del Caso de Estudio 102
4.3.2 Análisis y Diseño del Caso de Estudio 108
4.3.3 Implementación y Ejecución del Caso de Estudio 123
4.3.4 Evaluación del Caso de Estudio 142
4.3.5 Cierre del Caso de Estudio 143
4.3.6 Monitorización y Control del Caso de Estudio 145
4.3.7 Mejora Continua del Caso de Estudio 147
5 Conclusiones y Líneas Futuras 149
5.1 Conclusiones 149
5.2 Líneas Futuras 150
6 Referencias 152
7 Anexos 153
7.1 Caso 1, Ejecución 1 153
7.2 Caso 1, Ejecución 2 156
7.3 Caso 2, Ejecución 2 161
7.4 Caso 3, Ejecución 2 167
7.5 Caso 6, Ejecución 1 169
7.6 Caso 9 171
7.7 Caso 11 173
7.8 Caso 10 182
7.9 Caso 12 185
7.10 Caso 13 187
7.11 Caso 14 189
7.12 Caso 6, Ejecución 2 190

7.13 Caso 7.....193

7.14 Caso 8.....195

TABLA DE ILUSTRACIONES

Figura 1-1: Principios Fundamentales de las Pruebas 11
Figura 2-1: Ciclo de Vida en Cascada 13
Figura 2-2: Modelo Genérico en V 15
Figura 2-3: Coste relativo por etapa en la que se detecta un error 32
Figura 2-4: Requisitos 43
Figura 3-1: Técnicas de Caja Blanca 49
Figura 3-2: Cobertura Sentencias VS Cobertura Ramas 51
Figura 3-3: Técnicas de Caja Negra 52
Figura 3-4: Clases de Equivalencia VS Valores Límite 56
Figura 3-5: Fases del Ciclo de Vida de las Pruebas de Integración 58
Figura 3-6: Diagrama de Actividades en la Planificación de Pruebas 58
Figura 3-7: Diagrama de actividades en la fase de Análisis y Diseño 60
Figura 3-8: Diagrama de Actividades en la Fase de Implementación y Ejecución 62
Figura 3-9: Diagrama Actividades en la Fase Evaluación 63
Figura 3-10: Diagrama Actividades Fase de Cierre 64
Figura 3-11: Diagrama Actividades de la Fase Monitorización y Control 65
Figura 3-12: Diagrama de Actividades de la Fase de Mejora Continua 67
Figura 4-1: Middleware 71
Figura 4-2: Mdw - Punto a Punto 72
Figura 4-3: Mdw - Multipunto 72
Figura 4-4: Mdw - Adaptadores 75
Figura 4-5: Canon de Operaciones 77
Figura 4-6: Entornos de Pruebas 79
Figura 4-7: Estados Casos de Prueba 87
Figura 4-8: Estados de los Defectos 89
Figura 4-9: Diagrama de Estados en PVCS 93
Figura 4-10: Estados Jtrac Estimaciones 96
Figura 4-11: Estados Jtrac GE 98
Figura 4-12: Organigrama 102

Figura 4-13: Funcionalidad incluida en la Estimación....104
Figura 4-14: Planificación Release 107
Figura 4-15: Leyenda Colores Planificación....108
Figura 4-16: Sistemas Impactados 110
Figura 4-17: Matriz de Requisitos....111
Figura 4-18: Hitos de Reingeniería Web....111
Figura 4-19: Estrategia de Pruebas....112
Figura 4-20: Plan de Pruebas de Reingeniería Web....122
Figura 4-21: Recursos o Datos....124
Figura 4-22: Planificación de Reingeniería Web 124
Figura 4-23: Estado PP en TestLab, lunes 18 125
Figura 4-24: Plantilla Incidencias - MDWAE_TI_2130....127
Figura 4-25: Estado TestLab martes 19 128
Figura 4-26: Plantilla Incidencias - WEB_TI_1770 131
Figura 4-27: Estado TestLab, miércoles 20 132
Figura 4-28: Estado TestLab, jueves 21....133
Figura 4-29: Estado TestLab, viernes 22 136
Figura 4-30: Estado TestLab, lunes 25....138
Figura 4-31: Estado TestLab, martes 26 140
Figura 4-32: Planificación Final Reingeniería Web....145
Figura 4-33: Status de la Release, miércoles 13....146

1 Introducción

Como informática he aterrizado en una de las muchas consultoras que se dedican a realizar parte de los servicios que requieren las grandes empresas. La consultora para la que trabajo es especialista en calidad de software, en concreto de las pruebas de software.

Tras más de cinco años trabajando en el mundo de las pruebas integradas, y con el proyecto final de carrera por presentar, qué mejor opción que exponer sobre lo que mejor conozco hasta el momento. Además, no es solo un trabajo, ha llegado a convertirse en una filosofía de trabajo pues considero a las pruebas una parte primordial para conseguir la calidad que el cliente final requiere.

A lo largo del documento que nos ocupa vamos a entrar de lleno en la ingeniería del software y más en concreto en la parte de calidad del software, centrada en la realización de pruebas para depurar el código creado.

A continuación, paso a explicar los objetivos y la motivación que supuso el caldo primitivo de este proyecto.

1.1 Objetivos

Para explicar la idea y la filosofía de la realización de pruebas, emplearé una metodología de pruebas que teorice los principios, aporte definiciones e instrumentos para modelar un plan de pruebas. Esta idea está basada en la certificación ISTQB en la que estoy titulada y que sin duda, será el apoyo teórico a este proyecto. Además, partiré de una base teórica imprescindible, se trata del libro de Roger S. Pressman titulado "Ingeniería del Software - Un enfoque práctico" que trata sobre la ingeniería del software y sus principios básicos, muchos de los cuales son aplicados por la certificación ISTQB y de ahí el caldo primitivo de este proyecto: realizar una comparación entre las teorías de Pressman e ISTQB identificando sus verdaderas utilidades en la práctica real de un trabajo de calidad de software.

Durante estos años en el mundo del testeo he desempeñado diferentes roles que me han aportado diferentes puntos de vista del objetivo y por ello, la idea es explosionar el proceso completo de pruebas visto desde que llega un proyecto hasta que lo damos por finalizado.

1.2 Motivación

La motivación única y principal es la exposición de la importancia de la realización de pruebas para conseguir un producto de calidad a lo largo del ciclo de vida del software. Y lo argumento a través de los siguientes apartados.

1.2.1 La Necesidad de las Pruebas

Desde hace unos 25 años, el software cada vez está más presente en los negocios, empresas, en nuestras vidas cotidianas y debemos ser conscientes de ello. Hasta el punto de que casi cualquier acción que queramos realizar a lo largo del día implica interactuar con el software de uno u otro modo. Desde el programa informático de citas que utiliza nuestro dentista, a los carteles informativos que vemos en el metro, las máquinas que controlan nuestro cuerpo durante una operación, sacar dinero de un cajero, el hecho de llamar por teléfono, que se extraiga el tren de aterrizaje de un avión, o realizar una compra... Podríamos pasarnos un día entero poniendo ejemplos sobre lo incluida que está la informática en nuestras vidas. La sociedad cada día requiere de sistemas informáticos más completos, robustos y eficaces o lo que es lo mismo, sistemas de software de gran calidad. Sobre todo en aquellos de los que dependen nuestras propias vidas, ya que lo cierto es que se produzca un error en el programa de citas por mucho que nos afecte no es vital, pero siempre querremos evitar un fallo en la máquina que controla nuestra vida mientras estamos bajo los efectos de la anestesia general en una operación. Con este ejemplo queda claro que el impacto de un fallo de un sistema a otro es muy diferente: pérdida de tiempo, problema medioambiental, pérdidas económicas o incluso de vidas humanas. Y por ello, tendrán diferente coste de control. Pero sin duda alguna, el factor común a todas ellas es que se

debe cuidar mucho la calidad de los servicios disponibles para nuestro bienestar. Es en este punto donde entra la ingeniería de software y más en concreto la fase de calidad de software. Una fase que día a día cobra más fuerza porque la experiencia está demostrando que fallos en el software van a existir, así que cuanto antes se localicen, menores serán las consecuencias que puedan ocasionar.

Un fallo puede ser ocasionado por diversos motivos [2]:

- El principal es el fallo humano. Ya sea un fallo en documentación, en codificación, o provocado por las condiciones en las que se generó el software, como presión temporal, complejidad de infraestructura o código o cambio tecnológico.

- En otras ocasiones son las condiciones ambientales las que pueden ocasionar fallos al modificar el hardware y afectar a la ejecución del software. Magnetismo, radiación, contaminación, etc.

Para apoyar aún más lo comentado en base a las pruebas, siempre viene bien apoyarse en las palabras de un doctor en ingeniería como es Pressman [1]: "La creciente inclusión del software como un elemento más de muchos sistemas y la importancia de los costes asociados a un fallo del mismo, están motivando la creación de pruebas minuciosas y bien planificadas. No es raro que una organización de desarrollo de software gaste el 40 por 100 del esfuerzo total de un proyecto en la prueba. En casos extremos, la prueba de software para actividades críticas (por ejemplo: control tráfico aéreo, control de reactores nucleares) puede costar ¡de tres a cinco veces más que el resto de los pasos de la ingeniería del software juntos!"

1.2.2 Conceptos Básicos

El software es elaborado por seres humanos que pueden cometer errores que dan lugar a defectos. Estos defectos pueden generar un fallo [3].

Sin ejecución:

- **Error**: Desviación entre el comportamiento real y el comportamiento esperado, no cumpliendo un requisito establecido.
- **Defecto (defect, fault)**: Anomalía en un componente o sistema que puede dar lugar a que no se lleve a cabo correctamente una función determinada. El término "bug" se aplica históricamente a los defectos en informática.

Con ejecución:

- **Fallo (failure)**: Manifestación de un defecto.
- **Equivocación (mistake)**: Acción humana que da lugar a un resultado incorrecto.
- **Enmascaramiento del error**: Varios estados de error se compensan mutuamente, no aparece el efecto del error.
- **Depuración**: Localización y corrección de errores internos. Incluye tanto el hecho de localizar el error como de corregirlo.
- **Prueba**: Proceso de ejecución de un programa para descubrir un error.
- **Caso de Prueba**: Unidad mínima y elemental de diseño y ejecución de pruebas que verifica por completo una salida, interfaz, componente o funcionalidad de una aplicación.

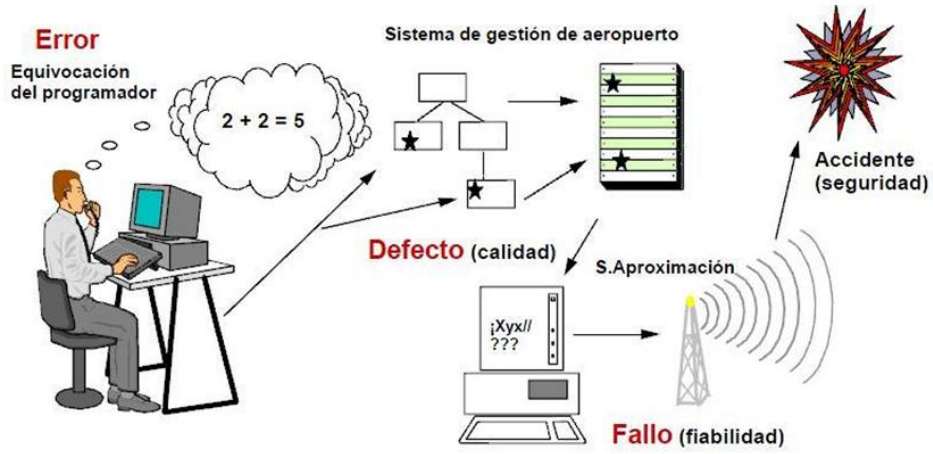


Figura 1-1: Principios Fundamentales de las Pruebas

Todo caso de prueba debe constar de:

- **Precondiciones**: Listado de indicaciones que deben cumplirse previamente a la ejecución del caso de prueba. Puede incluir desde acciones de hardware, instalaciones de software, estado del entorno de prueba, ejecución de algún otro caso de prueba, ejecución de alguna función previa, existencia de datos concretos en el sistema, especificaciones a nivel máquina y/o aplicación, ...

- **Condiciones de entrada**: Datos imprescindibles para poder lanzar una prueba, produciendo un único resultado esperado. El dato de entrada es el detonador para que una prueba pueda ejecutarse y sin dato, no hay prueba. Eso sí, deben realizarse pruebas tanto con datos válidos como con datos no válidos para averiguar cómo se comporta el sistema ante un caso de error, ya que debería tenerlos controlados.Esto dará robustez al sistema de pruebas y finalmente al producto que se esté probando.

- **Resultados esperados**: Es lo que se quiere obtener al ejecutar la prueba. Es una especificación del comportamiento esperado, así se evita que se puedan hacer validaciones equívocas sobre el resultado obtenido, pues el resultado queda ceñido a lo que viene explicado en el caso de prueba. Pueden ser datos de salida o acciones que el sistema realiza, pero tienen que estar perfectamente descritos.

- **Poscondiciones**: Listado de indicaciones que debe ser satisfecho tras la ejecución de una prueba o un procedimiento de pruebas. Es la aclaración a cómo debe quedar el entorno, sistema, aplicación, ... tras la ejecución de una prueba dada.

- **Procedimiento de pruebas o Pasos a Seguir**: Guía de estados consecutivos a seguir para llevar a cabo la ejecución de una prueba. Es posible especificar cada paso de manera que tenga sus propias precondiciones y poscondiciones o verificaciones.

- **Datos de prueba o Recursos**: Son los valores bajo los que se ejecutará una determinada prueba. Los datos de prueba se escogen en base al resultado que se espera que produzcan en la prueba, por ejemplo, para una prueba OK el dato de prueba debe estar dentro de los rangos esperados por el sistema. Mientras que para obtener una prueba de error, el dato de prueba a utilizar o recurso debe estar fuera del rango esperado para producir un fallo y así estudiar el comportamiento del sistema frente a entradas inesperadas. Pueden ser reales o ficticios, esto dependerá del entorno en el que se lancen las pruebas.

2 Estado del Arte

A lo largo de este capítulo se establece el marco de trabajo. En primer lugar, se enmarcan las pruebas dentro de la ingeniería del software y formando parte de un ciclo de vida como producto que es el software. En segundo lugar, se exponen los diferentes tipos de prueba existentes. Y por último, se revelan una serie de valores, obtenidos la mayoría de ellos de ISTQB, que dan soporte al sentido de la realización de pruebas como una fase fundamental en el ciclo de vida.

2.1 Ciclo de Vida del Producto

Los modelos de procedimiento relacionan métodos de desarrollo de software con fases de proyecto para permitir un desarrollo del proyecto estandarizado. Un ejemplo claro de ello, y seguramente se trate del modelo más extendido es el **modelo en cascada**. Se trata de un modelo claro y conciso, es básico pero con más de un defecto considerable, como es su rigidez.

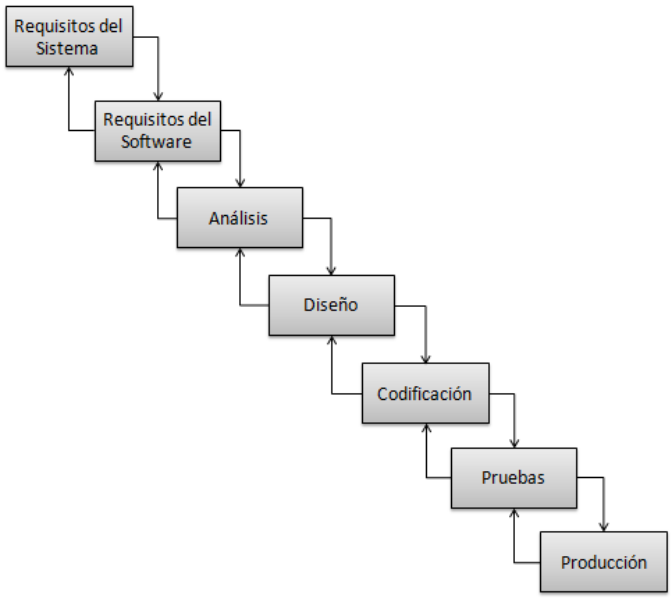


Figura 2-1: Ciclo de Vida en Cascada

El modelo en cascada es un proceso de desarrollo de siete etapas, donde cada etapa solo tiene una conexión hacia atrás con la etapa anterior. Y las pruebas sólo representan una etapa en el proceso, que además se corresponde con la aceptación final del producto y se encuentra al final del ciclo de vida.

Con el paso del tiempo y con la experiencia que se ha ido obteniendo, más en concreto en la calidad del software, los ciclos de vida de los productos han evolucionado otorgando el lugar que merecen a las fases de pruebas, ya que tienen una amplia repercusión en la obtención de un software de calidad.

2.2 Modelo Genérico en V

El modelo genérico en V es un modelo de procedimiento para el proceso de desarrollo de software. En él, el desarrollo y las pruebas se representan mediante dos ramas de la misma consideración y cada fase del desarrollo se contrapone a una fase diferente en las pruebas. Además, con el fin de ganar tiempo, en algunas ocasiones la preparación de las pruebas se puede llevar a cabo de manera paralela al desarrollo de software (siempre que se vaya realizando sobre bloques cerrados), anticipando así la especificación de casos de prueba o la planificación de las pruebas.

De gran relevancia es tener en cuenta que ambas partes del modelo han de ser consideradas de igual valor y que juntas contribuyen a la elaboración de software de calidad.

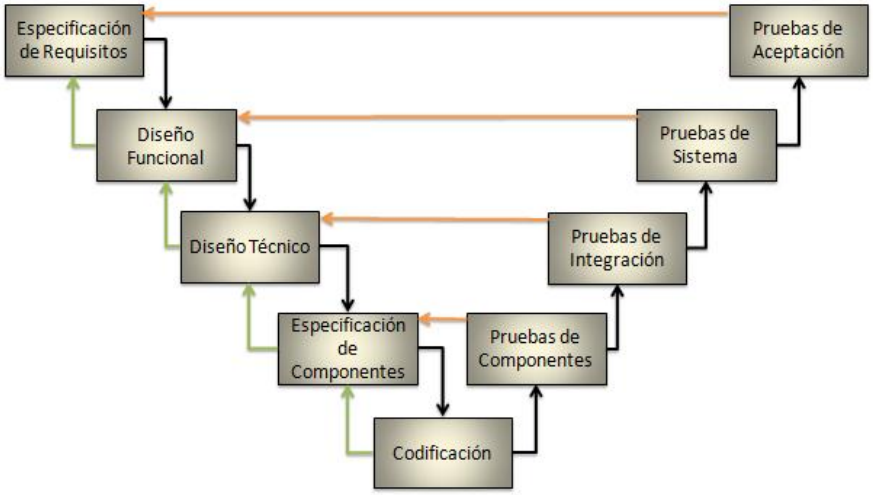


Figura 2-2: Modelo Genérico en V

La rama izquierda del modelo corresponde a las fases del desarrollo:

- **Especificación de requisitos**: es la definición de las características del software.
- **Diseño funcional**: es la conversión de los requisitos en funciones y procesos.
- **Diseño técnico**: en él se define el entorno del sistema, el diseño de intefaces, la arquitectura del sistema.
- **Especificación de componentes**: es la elaboración de los diferentes componentes que formarán el sistema.
- **Codificación**: es la transformación de los componentes en código ejecutable.

La rama derecha del modelo genérico en V corresponde a las ***fases de pruebas del software***, que paso a explicar de forma más detallada en los siguientes puntos [2]:

2.2.1 Pruebas de Componentes

Se prueba la funcionalidad de cada componente individual de manera separada y aislada, también se denominan pruebas **unitarias o de desarrollo**. Se trata de las primeras pruebas que se realizan sobre el software implementado y son las pruebas de cada uno de los componentes básicos del software (componente) respecto de su programación. Así podrían ser pruebas de módulo, de clases, de unidad, etc... Y su finalidad es asegurar la funcionalidad del componente dado intentando localizar errores en la elaboración o ausencia de funciones especificadas en los requisitos.

Normalmente, en este tipo de pruebas es el propio desarrollador quien ejecuta las pruebas unitarias y dado su conocimiento acerca de funciones, estructuras y variables, se emplean técnicas de caja blanca (que veremos en detalle más adelante) para el diseño de los casos de prueba. Además, aunque no es una buena costumbre, los defectos suelen corregirse según se prueba, sin llegar a abrir incidencias; lo que implica una falta de documentación sobre los primeros fallos localizados.

Para la ejecución de los casos de prueba, se emplean por regla general stubs (maquetas) y según el caso, controladores de pruebas que constituyen o utilizan la intefase abierta del objeto de prueba, que en este caso sería el componente. Los controladores de pruebas pueden simular entradas y recoger valores de salida.

Por otro lado, los stubs se necesitan cuando los componentes a probar requieren por su parte entradas o deben manipular salidas, ya que reemplazan o simulan componentes que aún no están disponibles. Y en ocasiones, pueden proporcionar datos de prueba (dummies). La diferencia entre los stubs y los controladores reside en que los stubs no contienen ninguna lógica de programación.

2.2.2 Pruebas de Integración

Estas pruebas valoran si los componentes individuales trabajan en conjunto tal y como se espera de ellos. O lo que es lo mismo, se prueba el funcionamiento de los diferentes módulos del sistema una vez unidos o agrupados en elementos mayores, verificando el comportamiento de los mismos frente a las comunicaciones que se produzcan entre ellos. El objetivo es la localización de errores de interfases y comprobar el correcto funcionamiento conjunto de los componentes.

Aún en los casos en los que los componentes básicos pasen sin error las pruebas de componentes, debe comprobarse su funcionalidad externa. Seguramente, con las pruebas de integración se localizarían todos los errores dados en las pruebas de componentes, pero sería una tarea más compleja, por ello, es aconsejable realizarlo en dos fases diferenciadas.

En las pruebas de integración debe tenerse en cuenta que existe la posibilidad de tener que reunir los resultados de diferentes desarrolladores y/o equipos de pruebas, pues cada componente puede tener un origen distinto. Y dado que no tiene por qué conocerse el funcionamiento intrínseco de cada módulo, es aconsejable que se utilicen técnicas de caja negra para el diseño de los casos de prueba y que además, sean equipos independientes del desarrollo quienes se encarguen de dichas pruebas.

Al igual que en las pruebas de componentes, pueden utilizarse stubs para sustituir a los componentes que falten por integrar. Además de controladores de pruebas que produzcan entradas y salidas para el sistema y registren datos. Pero ojo, la ejecución de pruebas con stubs/controladores y cuyos resultados sean OK, no asegura que después de realizar la integración completa de sistemas vaya a funcionar todo como se espera.

En las pruebas de integración, se diferencian varios *tipos de estrategia* a seguir:

- **Top-Down**: En esta estrategia la integración se lleva a cabo sistemáticamente de arriba a abajo. Se empieza por los componentes que no son llamados desde otros componentes del software y paso a paso se integran aquellos componentes que únicamente son llamados desde la parte ya integrada.

  Debido a la forma de proceder no son necesarios controladores de prueba, ya que todos los componentes subordinados deben ser sustituidos por stubs.

La implantación de la estrategia Top-Down solo es posible para software construido jerárquicamente de manera continuada, es por ello que en su forma pura no tiene apenas relevancia práctica.

- **Bottom-Up**: Se trata de la estrategia de pruebas inversa a Top-Down. En esta estrategia, se lleva a cabo una integración de abajo a arriba. Se empieza por aquellos componentes que no llaman a otras partes del programa y se continua por los componentes que solo llaman a la parte ya integrada. Se suele aplicar en desarrollos nuevos, pruebas de equipos grandes y distribuidos.

  Al contrario que en la estrategia Top-Down, en Bottom-Up no existen huecos que deban ser completados por stubs porque los componentes superiores deben ser simulados mediante controladores de pruebas.

  El punto común entre Top-Down y Botton-Up es que en ambos su forma pura no suele ser posible de aplicar.

- **End-to-End**: Es una estrategia de pruebas de integración orientada al proceso de negocio en el cual se integran en cada caso los componentes necesarios por parte de un proceso completo de negocio, o lo que es lo mismo, al emplear la estrategia se valida si el flujo de una aplicación está funcionando tal y como fue diseñado.

  En la estrategia End-to-End se pueden emplear controladores para sustituir los componentes de rango superior y stubs para los subordinados.

- **Funciones**: consiste en diseñar la integración orientada a una función del sistema, de manera que se integra cada uno de los componentes necesitados por la función correspondiente del sistema. En verdad, este tipo de estrategia por funciones podría decirse que es un subconjunto de la estrategia End-to-End centrada en unas funciones en concreto, con la diferencia de que no tienen por qué ser funciones que validen un comportamiento dado de principio a fin.

- **Ad-Hoc**: en esta estrategia se enfoca el software como módulos independientes y una vez se finaliza su implementación y validación exitosa por parte de desarrollo, se integran. La ventaja de la estrategia Ad-Hoc es que no existen retrasos en el comienzo de las pruebas,

por lo que ocasionalmente se puede lograr un acortamiento del proceso completo de desarrollo del software.

La utilización de controladores y stubs dependerá del tipo de componente terminado.

La estrategia Ad-Hoc en la práctica puede ser combinada con otras estrategias pero lo cierto es que puede ser empleada en cualquier situación en la que exista un esquema modular.

- **Big-Bang**: es la única estrategia de pruebas integradas que no es incremental. Esto es que en las estrategias vistas hasta el momento se añade un elemento por iteración mientras que en la estrategia Big-Bang se integra únicamente en el momento en el que se dispone de todos los componentes.

  Esta estrategia tiene dos grandes inconvenientes, que son: en primer lugar que el área de pruebas tendrá un tiempo de espera, en el que no se podrán ejecutar pruebas, ya que la integración no comienza hasta que no se finaliza el software completo. Y en segundo lugar, que las pruebas seguramente sean más complicadas y amplias y además los efectos de los errores que aparezcan serán difíciles de rastrear.

2.2.3 Pruebas de Sistema

Son las pruebas para validar que el sistema completo cumple los requisitos marcados. También podrían denominarse pruebas **funcionales**. A pesar que desde el punto de vista técnico, llegada esta fase de pruebas ya se habrán probado todos los componentes y su interrelación, sí faltarían las pruebas del sistema completo en condiciones de funcionamiento y desde el punto de vista del usuario, como serían: el entorno, las funciones, la carga, ...

En las pruebas de sistema, el entorno de pruebas debería corresponderse con el entorno de producción, sin llegar a ser el entorno de producción real. Para ello, deben omitirse los controladores de pruebas y stubs y las intefaces externas del sistema se probarán bajo condiciones de producción, porque la recreación debe ser lo más exacta posible al posterior

entorno de producción.

Las pruebas de sistema deben validar el cumplimiento de los requisitos establecidos, ya que en principio se prueba la calidad del software para el usuario. De acuerdo con la calidad del software según la ISO 9126 las pruebas de sistema diferencian entre ***requisitos***:

- **Funcionales**: Se comprueba ejecutado una batería de pruebas que las características quedan implementadas por las funciones disponibles, que son:
    - Idoneidad: Mide si son adecuadas las funciones disponibles para la utilización prevista.
    - Precisión: Comprueba si se ejecutan las funciones correctamente, en cuanto a cómo estaba acordado.
    - Interoperabilidad: Asegura que haya una interrelación libre de errores con el entorno del sistema.
    - Conformidad: Valora si se han cumplido las normas y preceptos establecidos.
    - Seguridad: Proporciona protección de los datos frente a accesos o pérdidas no esperadas.

      Las pruebas de los requisitos funcionales normalmente quedan contenidas en un plan de pruebas bien diseñado.
- **No funcionales:** el cumplimiento de estos requisitos es igual de importante pero a menudo difícil de probar y por ello están sometidos a un mayor riesgo. En la definición de requisitos no está siempre claro "cómo de bien" debe funcionar algo. Se dan a menudo de manera implícita, no tangible y por este motivo no se definen, el cliente no los define concretamente y suele ser difícil una cuantificación de los requisitos no funcionales, son los siguientes:
    - Fiabilidad: Es la cualidad que contiene el hecho de que un sistema haga lo que se espera de él.

- Usabilidad: Es la facilidad de utilización que ofrece el sistema, el cómo es visible y comprensible a ojos de un usuario.

- Eficiencia: Es la calidad en el tiempo de respuesta del sistema.

- Mantenibilidad: Es la habilidad que ofrece un sistema en cuanto a que el software pueda ser corregido o ampliado en el tiempo sin que esto sea un trabajo especialmente tedioso, complicado y agobiante.

- Portabilidad: Es la capacidad de un sistema para funcionar en diferentes plataformas, ya sean software o hardware.

  Las pruebas de los requisitos no funcionales se realizan completando pruebas de carga, de rendimiento, de volumen o masivas, de estrés, de seguridad de datos, de estabilidad, de robustez, de compatibilidad, de idoneidad de uso (usabilidad), comprobación de la documentación, de la mantenibilidad, ...las cuales explicaremos más adelante.

2.2.4 Pruebas de Aceptación

Son las pruebas finales que realiza el cliente con el fin de verificar por sí mismo el cumplimiento de los requisitos especificados, justo antes de su paso a producción. También son conocidas como **pruebas de usuario o UATs**. En dichas pruebas el usuario valida si el software cumple sus expectativas.

Las pruebas de usuario son pruebas funcionales, habitualmente de caja negra, ya que el usuario no tiene por qué tener conocimientos técnicos sobre el software. Al usuario, que es quien encargó el software inicialmente, sólo le interesará su estado final: la intefaz y la funcionalidad. Durante las pruebas de aceptación, el usuario realiza pruebas sobre un software bastante depurado, pues ha sufrido ya pruebas de desarrollo y pruebas integradas en las fases previas, por lo que lógicamente el número de fallos y sobretodo de una gravedad importante, queda reducido considerablemente.

En ocasiones, pueden realizarse pruebas de campo que consisten en una distribución anticipada de versiones estables del software, o lo que es lo mismo, versiones beta que se facilitan a una selección representativa del círculo de clientes, a las que someterán a una batería de pruebas en un entorno de usuario. Dicha versión beta, puede ser precedida por una versión alfa que se instalaría y probaría en el entorno de desarrollo.

2.2.5 Otros Tipos de Pruebas

Además de las pruebas de componentes, integración, sistema y aceptación, asociadas a etapas del ciclo de vida V del software, existen una serie de pruebas que no están ceñidas a una fase en sí, porque algunas de ellas pueden realizarse en cualquiera de las fases de pruebas existentes. Deben tenerse en cuenta, son las siguientes:

- **Pruebas de PostProducción**: Una vez que el cliente ha aceptado el producto y se ha instalado en el entorno de producción, se puede decir que las fases propias del desarrollo y las pruebas relacionadas con él han finalizado. Sin embargo, el software mismo está al comienzo de su vida real. A menudo se implanta para muchos años, mientras que se siguen desarrollando nuevos módulos para añadirle. Es muy posible que siga teniendo algún tipo de error y por ello se debe trabajar sobre él. Así, es conveniente la realización de pruebas que validen el comportamiento del software una vez instalado en el entorno de producción o lo que es lo mismo, en el entorno real sobre el que va a funcionar. Aunque ya con datos reales, se crea una serie de datos que utilicen solo para pruebas pero con los que se puedan comprobar funcionalidades esperadas del sistema en el entorno de producción.

- **Pruebas de Regresión**: Las pruebas de regresión son aquellas que se realizan sobre la funcionalidad que no se ha modificado, es decir, la que ya existía antes de instalar el software nuevo; con el fin de asegurar que ciertas operaciones "antiguas" sigan funcionando como se espera de ellas tras la instalación de nuevo software. Con estas pruebas se valida si debido a las modificaciones del software surgen nuevos errores y lo que se hace es probar de nuevo funcionalidades que ya se probaron en su momento, estudiando si se mantiene el funcionamiento esperado después de haber añadido

software.

Este tipo de pruebas se realizan durante las diferentes pruebas del proceso (desarrollo, integración, UATs...) y son de gran importancia, pues aunque desde luego, la finalidad es subir a producción nueva funcionalidad, no deja de ser de interés el que no se estropee la funcionalidad existente.

Las pruebas de regresión cobran cierto interés cuando se prueban justo antes de subir un software a producción o nada más subir un software a producción. Al ejecutarlas en estos puntos del ciclo de vida lo que se consigue es seguridad ante una instalación de software, ya que si no dan los resultados esperados o bien no se subiría el software a producción hasta que no se solucionasen los defectos localizados, o bien se podría dar marcha atrás a la subida de un software a producción.

- **Pruebas No Funcionales**: Pueden realizarse pruebas no funcionales en todos los niveles de prueba. El término prueba no funcional describe las pruebas necesarias para medir características de sistemas y de software que pueden cuantificarse según una escala variable, como pueden ser los tiempos de respuesta en las pruebas de prestaciones o rendimiento. Estas pruebas pueden hacer referencia a un modelo de calidad como el definido en el estándar de calidad ISO 9126.

    - *Pruebas de Carga*: Se realizan para conocer el umbral de funcionamiento del sistema con gran cantidad de usuarios/acciones a la vez y evaluar el dimensionamiento.

    - *Pruebas de Rendimiento*: Comprueban la velocidad del sistema en determinadas funciones o casos de uso.

    - *Pruebas de Volumen o Pruebas Masivas*: Se hacen para validar el comportamiento del sistema a la hora de procesar grandes cantidades de conjuntos de datos y/o archivos a tratar.

    - *Pruebas de Estrés*: Son para verificar el comportamiento del sistema en situación de sobrecarga de usuarios o de datos, llevando al sistema al extremo de su capacidad, por encima del umbral detectado en las pruebas de carga.

- *Pruebas de Seguridad*: Se trata de pruebas para comprobar si el sistema está protegido frente a accesos no autorizados o pérdidas por diversos motivos.

- *Pruebas de Estabilidad*: La finalidad de estas pruebas es la degradación del sistema ante una carga continuada. Así se estudia el comportamiento del sistema frente a un funcionamiento prologando y por ejemplo, se valida con qué frecuencia se cae el sistema por unidad de tiempo determinada.

- *Pruebas de Robustez*: Son las pruebas que se realizan para estudiar la reacción del sistema frente a una utilización incorrecta o frente a errores de entrada, o lo que es lo mismo, la reacción frente al manejo de excepciones.

- *Pruebas de Compatibilidad*: Se realizan para validar el trabajo del sistema junto con otros programas. Requiere una conversión de datos y unas interfaces hacia otros programas. Además, se puede comprobar la reacción del sistema en diferentes entornos hardware o de sistema operativo.

- *Pruebas de Idoneidad o de Uso o Usabilidad*: Son las pruebas que se realizan para validar cómo de complejo es el acceso, utilización, visibilidad y comprensible es el sistema. Además, se valida la facilidad de aprendizaje de su manejo para un usuario.

- *Pruebas de Documentación*: Son una serie de validaciones sobre la documentación que se entrega sobre un proyecto concreto. En este tipo de pruebas lo que se valida es si concuerda la documentación del programa con el sistema real, si la documentación se ha escrito de manera clara y comprensible, si se ha entregado toda la documentación prevista completa...

- *Pruebas de Mantenibilidad*: Se comprueba si se dispone de todos los documentos adecuados acerca del desarrollo del sistema, si se han cumplido los estándares de código preestablecidos o si el sistema dispone de una arquitectura estructura o modular. Se trata de una pruebas menos comunes pero de gran utilidad y todas estas validaciones son de cara a la facilidad de mantenimiento a largo plazo del sistema.

Se puede realizar otra clasificación en base al tipo de ejecución de la prueba:

- **Pruebas Manuales**: Son aquellas en las que se requiere la presencia de un ingeniero de pruebas para realizar las acciones descritas por un caso para llevar a cabo la ejecución de una prueba.

- **Pruebas Automatizadas**: Con el fin de evitar repetir una y otra vez la ejecución de las mismas pruebas, para ciertas pruebas es factible realizar una automatización de las mismas. Para ello se implementan los pasos de los casos de prueba de manera que puedan ser ejecutados en modo desatendido. Este tipo de pruebas tienen un uso cada vez más extendido en casos de pruebas de regresión, ya que se trata de un bloque de pruebas que se repite en determinadas ocasiones y aunque supone un coste adicional su automatización y mantenimiento, el ahorro en ejecución a largo plazo es importante.

  Actualmente se están realizando mediante Selenium y Rational y para su mantenimiento es necesario que haya un equipo que vaya actualizando los scripts. De hecho, para aplicaciones como una web, que cambian continuamente, el coste en mantenimiento de los scripts es alto.

  Para aclarar el término de automatización, podemos definir la automatización como la conversión de un proceso manual en automático donde:

  - Un "caso de prueba automatizado" es un programa.
  - Un proyecto de automatización es un proyecto de desarrollo y debe estar sujeto a los mismos estándares y normas.
  - Es necesario que los perfiles sean los apropiados: técnicos.
  - Debe existir un proceso de pruebas formalizado, con casos de prueba detallados y un entorno de pruebas.

2.3 Principios Fundamentales de las Pruebas de Software

El objetivo fundamental de este documento, es en su completitud describir lo mejor posible qué es una prueba, en qué consiste y cuál es su finalidad. Podemos reiterar que las pruebas de software tienen una importancia relevante dentro del proceso de calidad del software y que si buscamos con dedicación, podemos tener por seguro que vamos a localizar errores dentro de un software.

Deutsch nos va a ayudar citándole en el siguiente párrafo [1]: "El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca la falibilidad humana son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos... pueden estar especificados de forma errónea o imperfecta, así como los errores que aparecen en los posteriores pasos de diseño y desarrollo... Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad".

Visto una vez más la importancia de las pruebas de software, pasamos a exponer siete principios que sirven de cimientos a las pruebas y que por supuesto, nos ayudan a definirlas con mayor claridad y concisión, [2]:

- **Principio 1: Las pruebas muestran la presencia de defectos**: Gracias a las pruebas se puede demostrar la presencia de defectos. Las desviaciones respecto a los resultados previstos que se descubren durante las pruebas permiten localizar defectos existentes en el software.

  Las pruebas no pueden probar la ausencia de defectos, sin embargo, sí reducen la probabilidad de que queden defectos sin descubrir. Aunque se debe tener presente que la ausencia de fallos no prueba que el software sea correcto.

  Las pruebas tienen como objetivo detectar errores y por tanto, nunca deben planificarse bajo el supuesto de que no va a encontrarse ninguno.Hay que estar preparado para detectarlos, por ello, el enfoque correcto es esperar que aparezcan.

- **Principio 2: Las pruebas exhaustivas no son posibles**: Seguir una estrategia de pruebas exhaustiva supone abarcar todas las posibles combinaciones de valores de entrada y precondiciones. Para una parte considerable de proyectos, esto solo es una idea feliz imposible de cumplir, entre otros aspectos por temas de coste y tiempo y porque en muchos de los casos se llegaría a un plan de prueba infinito, lo cual es inviable desde todos los puntos de vista posibles.

  Una solución a esto puede ser el muestreo en las pruebas o seleccionar las pruebas, es decir, realizar una priorización de las pruebas.

  - Las pruebas deben incluir solo un subconjunto de todos los valores de entrada posibles. Las entradas seleccionadas pueden escogerse de forma sistemática o aleatoria.

  - En condiciones reales, se suele emplear el muestreo. Las pruebas de todas las combinaciones de entradas y precondiciones es viable desde el punto de vista económica sólo en casos triviales.

  - En lugar de llevar a cabo pruebas exhaustivas se debería utilizar el análisis de riesgo y el uso de priorizaciones para optimizarel esfuerzo en pruebas. No hay que probarlo todo y fijarse en los riesgos ayuda mucho a priorizar las pruebas a diseñar. Por ejemplo, se puede priorizar los casos para ejecutar en primer lugar los elementos más críticos basándose en las consecuencias de coste y/o tiempo si fallase x condición. Se trata de un orden de lanzamiento de las pruebas que va en base a los requisitos definidos por el usuario, marcados porel negocio.

- **Principio 3: Pruebas tempranas**: Las actividades de prueba deberían iniciarse tan pronto como sea posible en el ciclo de vida de desarrollo y deberían tener objetivos concretos.

  Cuanto antes se descubra un defecto, menos costosa será su corrección.

  - La máxima efectividad se alcanza cuando los errores se corrigen antes de ser implementados.

- Pueden probarse los documentos de requisitos conceptuales y las especificaciones.
- Los defectos que se descubren en la fase de concepción del proyecto se corrigen con mínimos esfuerzos y costes.

La preparación de una prueba consume tiempo. La prueba no es solo la ejecución.

- Pueden realizarse actividades de prueba, sobretodo el diseño, antes de que se complete el desarrollo del software.
- Las actividades de prueba, como pueden ser las revisiones de las mismas, deberían llevarse a cabo en paralelo a la especificación y el diseño de software.

- **Principio 4: Bloques de defectos**: Unos pocos módulos contienen la mayor parte de los defectos descubiertos durante la fase de pruebas o son responsables de la mayoría de los fallos en producción.

  El principio se basa en que al encontrar un defecto se encontrarán más en los alrededores. Los defectos aparecen frecuentemente en grupos, como las setas. Es una buena idea repasar el módulo en el que se ha encontrado un defecto porque la estadística nos dice que es muy posible que se localicen más defectos.

  Los probadores deben ser flexibles. Una vez detectado un defecto es una buena idea reconsiderar la dirección de las siguientes pruebas, por supuesto, dependiendo de la importancia del defecto localizado. La localización de un defecto debe ser revisada con un mayor nivel de detalle, ya sea incorporando nuevas pruebas o modificando las existentes.

  Además, detectar cuáles son los puntos débiles de la compañía o del equipo de desarrollo puede dar pistas de donde se van a localizar defectos. Esto lo da la experiencia. Cuando se conoce al desarrollo, se sabe dónde suelen tener más dificultades y el tester conocedor de ello, deberá buscar fallos donde considere que residen los problemas de desarrollo centrando el esfuerzo en ellos y ajustando la estrategia.

- **Principio 5: La paradoja del pesticida (Bezier)**: Si se repiten una y otra vez los casos de prueba, se llegará a que el mismo conjunto de casos de prueba no sirva para localizar nuevos defectos.

  Para superar esta "paradoja del pesticida", los casos de prueba necesitan ser revisados de manera regular y se necesita escribir casos nuevos y diferentes para ejercitar diferentes partes del software o sistema para localizar más defectos, especialmente si se han producido cambios en el código. Esto es, evitar pruebas innecesarias y redundantes.

- **Principio 6: Las pruebas dependen del contexto**: Las pruebas se llevan a cabo de manera diferente en contextos diferentes ya que pueden tener distintos resultados en distintos contextos.

  Un ejemplo de esto serían las pruebas a realizar en el entorno de integración frente a las que se realizan en el entorno de producción, pues aunque los entornos serán muy parecidos, no serán iguales y los ejecutores de las pruebas fijarán sus objetivos en puntos diferentes de los resultados que obtengan y por tanto, el enfoque de las pruebas así como de los resultados esperados deberán ser sutilmente diferentes.

- **Principio 7: La falacia de la ausencia de errores**: Localizar y corregir defectos no sirve de nada si el sistema construido no cubre las necesidades y expectativas marcadas por el cliente en los requisitos.

  Realmente, solo con las pruebas no se verifica la calidad del software en su totalidad, ya que es posible que se estén obviando los requisitos especificados. Esto es, que un software puede estar prácticamente libre de fallos pero puede no ajustarse en nada a lo solicitado por el cliente si el software no se adapta a la perfección a los requisitos acordados. Además, el propio procedimiento de pruebas puede contener errores y las condiciones de prueba pueden no estar bien preparadas para la localización de errores. Por consiguiente, la calidad de un producto debe construirse desde el principio de la creación del software y mantenerse a lo largo de todo el ciclo de vida del software.

Para fortalecer aún más los principios fundamentales de las pruebas, añado una serie de puntos que no pueden obviarse en el proceso de creación de casos de prueba [3]:

- No pueden faltar los casos de prueba tanto para las condiciones de entrada válidas como para aquellas que no lo sean. Es primordial para comprobar la robustez del sistema fijarse bien en cómo responde ante datos que espera y sobre todo ante los datos que no espera.

- Realizar una descripción detallada de todos los componentes de un caso de prueba es un aspecto clave en el diseño. Todo caso de prueba debe constar de:

    - Precondiciones

    - Datos de entrada

    - Resultados esperados

    La idea parte de la realización de un diseño completo que facilite la etapa de la ejecución en sí y que no lleve a confusión. Debe ser explícito en cuanto a las condiciones del entorno previas a la ejecución de la prueba, a los datos de entrada que deben emplearse y al resultado que se espera que se produzca.

- Revisar cuidadosamente el resultado obtenido de cada prueba, comparándolo con el resultado esperado.

    Por muy lógico que pueda parecer el resultado obtenido a los ojos del tester, éste debe coincidir con el resultado esperado definido en el diseño de la prueba para poder dar por buena la ejecución. Ya que el resultado esperado estará definido en los requisitos del usuario.

- No olvidar nunca el negocio sobre el que se trabaja. Es mejor detectar una incidencia grave que pueda impedir que el negocio salga adelante, que diez mejoras que no afectan al mismo.

    Poner especial foco en las funcionalidades más impactantes a nivel de alcance funcional y que más afecten al negocio.

- Un programador debe evitar probar su propio programa. Como mucho debería realizar las pruebas unitarias.

  Hay que definir a los probadores y al equipo al que deban pertenecer. Ya que un factor importante es el humano, ¿a quién le gusta localizar errores sobre su propio trabajo? Esta es la razón de que a día de hoy se piense que lo óptimo es que el equipo de pruebas sea totalmente independiente del equipo de desarrollo.

- Las pruebas son una tarea para la que hace falta creatividad e imaginación. Suponen un desafío intelectual, como mínimo, igual al de programar.Localizar problemas allá donde los desarrolladores no los han detectado, requiere de un gran ingenio y de máxima capacidad de concentración para no dejar escapar los fallos creados.

2.4 ¿Cuándo podemos decir que las pruebas realizadas han sido suficientes?

No solo en la ingeniería de software se plantea la duda de la cantidad de esfuerzo suficiente a realizar sobre una tarea para que ésta quede finalizada [2]. Es cierto que en algunos sectores es algo más complejo delimitar los esfuerzos a invertir, pero por supuesto, y más tratándose de ingeniería, no es más que un reto más (valga la redundancia) que afrontar y superar buscando una solución. Podría decirse que en el caso de las pruebas se puede aplicar lo que de alguna forma conocemos como "la ley del mínimo esfuerzo", pero eso sí, con el máximo rendimiento.

Por más pruebas que se realicen sobre un software dado, nunca se podrá asegurar al 100% que dicho software esté libre de fallos completamente. Hay que ajustar el **ámbito** de las pruebas realizando una batería de pruebas finita que garantice que se encontrarán errores para que sean tratados a tiempo. En la práctica, unas pruebas exhaustivas son rara vez posibles, y no se puede garantizar para las partes probadas la ausencia total de errores. Esto es porque no se puede probar absolutamente todo. Sí se podrá garantizar que sobre las pruebas realizadas, el software está "limpio", pero no que no tenga errores.

Ya que no se puede realizar un plan de pruebas infinito, hay técnicas que ayudan al diseñador a definir las pruebas de manera que sean lo más completas posibles. Para ello, es importante realizar pruebas sobre lo más *prioritario* en base a lo que conlleve un mayor *riesgo* y lo que haya solicitado el cliente, o lo que es lo mismo, definir un **alcance** de las pruebas que se deben diseñar en base al *impacto* que tendrían en un futuro si fallasen. Por ejemplo, no es igual el riesgo en un error que pueda producirse en el sistema del detector de combustible en un avión comercial que en el de un coche.

Dado que cada mercado es un mundo diferente, en este aspecto cobra mucha importancia el cliente, pues es quien verdaderamente conoce el *negocio* sobre el que se ha desarrollado el software y por tanto, es quien debe indicar la priorización a seguir. O en su caso, las consultoras específicas que conocen muy bien el negocio en el que se mueven, motivo por el cual las grandes empresas externalizan algunos de sus servicios hoy en día.

Como hemos visto el esfuerzo no puede ser infinito por cuestiones claras de tiempo y por supuesto, de **coste**. Con el paso del tiempo, debido a la experiencia, se ha establecido la afirmación de que los errores que se localizan en un software son menos costosos de solucionar cuanto antes se localicen en el ciclo de vida del producto. Uno de los objetivos es reducir la posibilidad de que ocurra una incidencia grave en producción, donde el impacto, sin duda, será mucho mayor. La idea está en que los costes de las pruebas deberían ser menores que los posibles costes de los errores. Además, una resolución temprana de errores mejora el producto a largo plazo. En la figura 2-3 se puede observar una estimación de la relación del coste asociado dependiendo de la fase en la que se detecta cierto error.

Fase en la que se detecta el error	Coste
Requisitos	1
Diseño	3-6
Codificación	10
Pruebas de Sistema	15-40
Pruebas de Aceptación	30-70
Producción	40-1000

Figura 2-3: Coste relativo por etapa en la que se detecta un error

Si en un futuro se descubren fallos, para asegurar que las pruebas realizadas en su momento fueron correctas y suficientes, habrá que demostrar que dicho fallo no se reprodujo durante la realización de las pruebas o que la prueba que produce el fallo no estaba contenida en el plan de pruebas inicial. Lo que apuntará a que el fallo del software sea debido a otras causas, como pueden ser una actualización del software o un fallo en el entorno.

En un proceso de pruebas deben ser probadas todas las funcionalidades de un software de manera sistemática. Si un software es nuevo se comprueba el correcto funcionamiento del software por completo. Cuando se trata de una actualización o un módulo nuevo, el foco sobre el que se debe centrar el mayor esfuerzo a realizar en pruebas es el software de nueva creación. Pero no menos importante es asegurar que el software que ya existía sigue funcionando de la misma forma. Esto se soluciona diseñando un plan de *pruebas de regresión* que es una batería de pruebas con las que se certifica si el software ya existente sigue funcionando de la misma forma que antes de instalar el software nuevo. Además, dicho plan de pruebas de regresión debe ejecutarse con cada nueva incorporación de software al software inicial.

Para finalizar el tema del esfuerzo necesario a emplear en pruebas, diremos que teniendo en cuenta los problemas generales que hemos identificado (ámbito, alcance y costes) utilizaremos los conocimientos teóricos sobre el diseño de casos de prueba para poder definir un conjunto finito y viable y que por supuesto, aporte fiabilidad y calidad, ya sea a un software nuevo o a uno ya existente.

2.5 Criterios de Finalización

Conocer cuando se deben finalizar las pruebas a realizar sobre un software es imprescindible para aportar calidad a un plan de pruebas. Algunas de las pautas o herramientas que ayudan a detectar los criterios de finalización de pruebas son [2]:

- **Ratio de localización de errores**. Consiste en finalizar las pruebas cuando el número de errores encontrados por hora de esfuerzo de pruebas desciende de un valor establecido. Aquí se tiene en cuenta que las pruebas dejan de ser rentables a partir de un determinado

ratio. Dicha rentabilidad reside en la localización de errores, y una vez fuera del ratio de errores, las pruebas se convierten únicamente en un gasto sin obtener resultados, por ello sería aconsejable finalizarlas. Se trata de una métrica simple pero no es suficiente. Eso sí, la fiabilidad del sistema dependerá de la calidad de las pruebas realizadas, además, la posibilidad de que exista un error crítico podría forzar la continuación de las pruebas.

- **Finalización de las pruebas por motivos de coste o tiempo**. Es un motivo de finalización aplastante pero conlleva un riesgo importante a tener en cuenta, ya que tiende a originar costes adicionales por la aparición posterior de errores, asimismo, no es un criterio que pueda asegurar la fiabilidad del sistema. Puede venir dado por ejemplo, a causa de una planificación escasa de recursos. En definitiva, este criterio de finalización denota una mala planificación de las pruebas.

- **Cobertura de código**. Es la finalización de las pruebas en el momento en el que se completa X% de código de programa ejecutado con resultado OK. Al tratarse de código, este criterio de finalización estaría ligado a las pruebas de sistema o unitarias. Y se podría extrapolar a las pruebas funcionales si en lugar de código se hablase de funcionalidad cubierta.

- **Cobertura de riesgos**. Una vez que los casos de prueba están ordenados en base a una priorización de ejecución debido al riesgo que llevan asociado. La cobertura de riesgos lo que pretende es que los casos de prueba de una clase de riesgo definida se hayan completado. Entendiéndose por completado el que se hayan ejecutado y en caso de haber localizado errores, estén corregidos y por tanto, el caso de prueba finalmente esté OK. Por ejemplo, un criterio de finalización marcado podría ser que se concluya el 100% de los casos de prioridad alta, el 80% de los casos de prueba de prioridad media y el 60% de los casos de prioridad baja. Al terminar dicho porcentaje probado se podría asegurar una fiabilidad del sistema bastante alta, impidiendo su subida a producción en caso de no cumplir los criterios de finalización establecidos. De esta manera, se certifica que de existir algún error en las pruebas que quedasen por ejecutar, al ser su prioridad menor, el impacto también lo sería.

Los criterios de finalización no aseguran en ningún momento una fiabilidad del 100% en la no existencia de errores. Es importante resaltar que la fiabilidad del sistema reside en la calidad de

las pruebas realizadas. Y que concatenadas a un buen criterio de finalización, sí pueden aportar una fiabilidad estable del sistema, al menos en cuanto a la depuración de errores de un gran impacto.

2.6 Diferencias entre Diseñar, Desarrollar y Probar

Las distinciones entre diseñar, desarrollar y probar software no son lo suficientemente evidentes en las teorías clásicas de la ingeniería de la informática. Por suerte con el tiempo y la experiencia, se va hilando más fino y a día de hoy podríamos afirmar que se trata de tareas perfectamente separadas y establecidas en su punto correspondiente cronológicamente en un ciclo de vida [2].

Previo a la discusión sobre las diferencias entre diseñar, desarrollar y probar software, partiremos de una definición específica de cada tarea:

- **Diseñar**: Es una de las fases tempranas en un ciclo de vida de software. Se trata de definir el cómo se cumplirá la especificación de requisitos definida por el cliente y previamente estudiada en la fase de análisis. El diseño se puede realizar a diferentes niveles, esto es:
    - *Diseño a alto nivel*: Es la definición del sistema como estructura, a nivel de módulos software y las relaciones que mantienen entre ellos.
    - *Diseño a nivel intermedio*: Es una especificación algo más detallada del diseño a alto nivel como puede ser la especificación de los módulos definidos, por ejemplo, indicando el tipo de lenguaje que va a utilizarse o incluso la definición de las clases a implementar.
    - *Diseño a bajo nivel*: Es la descripción de los módulos con un detalle mucho más cercano a lo que sería el código, pinceladas de los atributos y métodos que se crearán en cada módulo.

- **Desarrollar**: Es la fase de codificación del código que compondrá el software final. Se realiza tras completar las fases previas de análisis y diseño, y por supuesto, siguiendo las pautas indicadas por las mismas. El desarrollo de un software engloba a su vez otras fases:

    - *Diseño*: definición de las especificaciones técnicas que van a componer el código.

    - *Pruebas unitarias*: Son las pruebas del sistema o módulo como entidad única, sin llegar a integrarlo con el resto de sistemas o módulos.

- **Probar**: Se somete al software a una batería de pruebas para verificar que su comportamiento es correcto y para ello, el objetivo es la localización de errores sobre las funcionalidades dadas. Se trata de una fase igual de importante que el resto de fases en un ciclo de vida de software y cuyo objetivo final es colaborar en la calidad del software creado, depurando su funcionamiento.

Las pruebas requieren una forma diferente de pensar de las que aquellos que diseñan o desarrollan sistemas. Mientras que la misión del *diseñador* es ayudar al cliente suministrándole los requisitos correctos, la misión del *desarrollador* es convertir los requisitos en funciones. Y la misión del *probador* es examinar la correcta implementación de los requisitos del usuario. Lo que hay que tener presente es que sin duda, el objetivo común a los tres roles es proporcionar un buen software de calidad.

En principio una persona podría asumir los tres roles. Pero si verdaderamente se quiere obtener un software de calidad, lo más conveniente es que se trate de personas diferentes, que ocupen un rol determinado en el proceso para producir mejores resultados. Podemos comparar un poco las diferentes posiciones:

- El **probador** mantiene una distancia suficiente respecto del objeto de la prueba. Cuanto más alejado esté el probador del desarrollo más independientemente y más objetivamente podrá llevar a cabo su trabajo. Sin embargo, una distancia demasiado grande respecto del objeto de prueba, puede provocar que se necesite más tiempo para las pruebas.

- El **desarrollador** es complicado que se mantenga neutral respecto a sus "creaciones". Es quien mayor conocimiento tiene sobre el software creado. Esto puede suponer una

ventaja respecto a las pruebas porque no será necesario un coste adicional para ponerle al corriente sobre el dominio del software a probar. Sin embargo, es una gran desventaja pues por mucho que lo intente no aplicará la suficiente objetividad en la realización de las pruebas. Hay que tener en cuenta que el hombre tiende a ser ciego respecto a sus propios errores y por tanto, el software no quedará lo suficientemente depurado. No obstante, lo más importante es que los errores que sean consecuencia de una mala interpretación de los requisitos quedarían sin descubrir.

Como se debe realizar una batería de pruebas para depurar el software, se podría jugar con dos supuestos:

- **El probador forma parte de desarrollo**: Si por temas de coste, de psicología o cualquier otro motivo, las pruebas a realizar se ejecutarán desde desarrollo, se plantean dos marcos:
    - *Desarrollo prueba sus propias creaciones*. Lo primero que surge es que no se localizarán muchos de los fallos existentes ya sean de software o de requisitos, consecuencia directa de no admitir errores sobre nuestras propias creaciones o interpretaciones. Al conocer perfectamente el software creado es muy complejo realizar una batería de pruebas robusta que ataque directamente a los puntos débiles del software, pues al ser propio será muy complicado identificarlos correctamente.
    - *Desarrollo prueba otras creaciones*: Bajo la idea de evitar al máximo posible los errores humanos, una opción es que los diferentes equipos de desarrollo se prueben el software unos a otros. De esta forma, una misma persona ejecutará el rol de desarrollador y de probador, pero no se dará el caso de que una persona pruebe un software que haya desarrollado. Es importante, que si se determina esta forma de trabajo los equipos puedan trabajar de la manera más independiente posible.
- **El probador sea externo a desarrollo**: Es lo que se conoce como **Outsourcing de pruebas** y sin duda alguna es la mejor opción posible. El coste tanto en tiempo como económico será mayor pues, en primer lugar, requiere un gran esfuerzo de puesta al

corriente, conocimiento del dominio del software y preparación previa a la ejecución de las pruebas por parte de los testers o equipo de pruebas. Pero asegura que haya un equipo especializado en pruebas preparado para depurar un software que no han desarrollado y sobre el que localizarán errores cometidos en las fases previas. La total separación de los roles de desarrollador y probador, tal y como la experiencia ha demostrado a lo largo de los años, certificará que su salida a producción sea más fructífera.

Diseño, desarrollo y pruebas son tres fases fundamentales en un ciclo de vida de software. Son fases que se complementan y cuyo objetivo final es común: contribuir a la elaboración del producto con la calidad suficiente y requerida por el cliente. Por tanto, de manera global se trata de fases constructivas y aunque puedan cuestionar el trabajo realizado en las fases anteriores o incluso considerarse la fase de pruebas en concreto, destructiva (término que no me gusta aplicar), siempre será con el objetivo de corregir lo antes posible los fallos localizados.

2.7 Personalidad de un Buen Probador

Un probador debe ser una persona cualificada, en principio un ingeniero de software para que tenga la formación adecuada al trabajo que va a realizar. Por ello, es de considerable importancia que un tester tenga las siguientes cualidades [2]:

- Ser **curioso**, atento a los detalles. Para comprender los escenarios prácticos en los que se mueve el cliente, para ser capaz de analizar la estructura de la prueba y para descubrir detalles que pueden identificar fallos.

- Tener **escepticismo** y **ojo crítico**. Los objetos de prueba contienen defectos, se trata de encontrarlos, por lo que deberá tener una buena actitud de trabajo para ser capaz de localizarlos cuando aparezcan ante él. No se debe ver limitado por el hecho de que un error grave pueda afectar a, por ejemplo, las fechas del proyecto, su misión es encontrarlos y ante temas que requieran una mayor responsabilidad tendrá que tener el suficiente criterio para escalarlo a sus superiores y que ellos se encarguen de resolver el problema.

- Buenas **capacidades de comunicación**. Para entre otras muchas tareas informar a los desarrolladores de los errores localizados, para no perder la perspectiva ante situaciones frustrantes, para establecer rápidamente una relación de trabajo con los desarrolladores porque trabajar en un ambiente cordial de trabajo, con una comunicación positiva puede ayudar a evitar o al menos facilitar situaciones complicadas. Para ello es importante el empleo de mano izquierda siempre que haga falta y de una expresión clara, concisa, con determinación pero sobre todo con respeto hacia las personas con las que se tenga relación laboral.

  Durante la fase de pruebas se tiene contacto directo y continuo con los desarrolladores, hay que evitar todo lo posible los problemas de comunicación:

  - Especialmente en fases del proyecto estresantes, la notificación de los errores puede generar problemas, sobre todo si los probadores son vistos como portadores de malas noticias. El trato debe ser positivo y constructivo en ambas direcciones: tester - desarrollador y desarrollador - tester.

  - La manera de notificar el error y la descripción del mismo son decisivos. Para ello se pueden establecer plantillas de comunicación de errores donde cierto tipo de información sea imprescindible rellenar para que los tester planteen con el detalle adecuado el problema localizado y los desarrolladores puedan entender e investigar sobre el error del que se les informa.

  - No criticar a las personas sino mostrar el error de forma objetiva. Puede parecer una obviedad pero es muy importante el enfoque que se da al reporte sobre un error. Por ello se hablará de un "posible error localizado" hasta que se confirme su verdadera existencia y jamás se afirmará, por ejemplo, que se trata de un error cometido por desarrollo dando al fallo localizado un enfoque impersonal.

  - Facilitar la solución del error al desarrollador mediante una notificación. Esto es un valor añadido que puede ser de gran utilidad para el desarrollador y sobretodo una ayuda interesante para conseguir una rápida solución. El tester al localizar un error y ser conocedor de la funcionalidad a esperar del software, debe aportar una

idea sobre qué puede estar sucediendo y si es posible, una idea de cómo solucionarlo.

- El objetivo común debe estar siempre en primer plano. El objetivo tanto de desarrollador como de probador es la depuración del software.
- Una comunicación insuficiente o la ausencia de ésta entre probadores y desarrolladores puede impedir el buen trabajo en equipo.

Hay varias formas de mejorar la comunicación y las relaciones entre los probadores y el resto:

- Empezar colaborando en lugar de entrar en refriegas que no llevan a un buen final. Recordar a todos que el objetivo común es obtener sistemas de mejor calidad.
- Comunicar lo que se ha encontrado en el producto de forma neutra, enfocada a los hechos, sin criticar a la persona que lo ha creado, por ejemplo, redactando informes de incidencias y revisiones de forma objetiva y ateniéndose a los hechos.
- Intentar comprender cómo se siente la otra persona y por qué reacciona como lo hace.
- Confirmar que la otra persona ha entendido lo que has dicho y viceversa.

- **Experiencia**. Ayuda a identificar donde se pueden acumular los errores. Como en cualquier función de desempeño de trabajo la experiencia es un valor añadido que aporta una mayor confianza sobre un determinado trabajador. Pero aún más importante que la experiencia es la actitud, la predisposición a realizar un buen trabajo. La experiencia se gana con el tiempo y se puede formar a alguien cualificado para que vaya ganando experiencia poco a poco, pero la actitud es una forma de ser que difícilmente se puede aprender.

2.8 Importancia de los Requisitos

Más de la mitad de los proyectos software que se desarrollan en la actualidad se alejan sustancialmente de los objetivos de presupuesto y calendario. Y muchos proyectos se abandonan o los productos resultantes no se usan. En casi todos estos casos los errores en requisitos tienen una importancia vital. Por ello, es muy importante entender bien su definición para realizar con fundamento cualquier tipo de proyecto desde el principio.

Tal y como Pressman nos indica [1] "Para que un esfuerzo de desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, decepcionará al usuario y desprestigiará al que lo ha desarrollado. [...] Tanto el desarrollador como el cliente juegan un papel activo en la especificación y el análisis de requisitos. El cliente intenta volver a plantear detalladamente el concepto, algo nebuloso, de la función y del comportamiento del software. El desarrollador actúa como interrogador, como consultor y como persona que resuelve problemas. El análisis y la especificación de los requisitos puede parecer una tarea relativamente sencilla, pero las apariencias engañan. El contenido de la comunicación es muy denso. Abundan los casos en que se puede llegar a malas interpretaciones o falta de información. Es probable que se llegue a ambigüedades..."

Para seguir argumentando la importancia de los requisitos y de manera más concreta, cómo afectan los requisitos a las pruebas, paso a la exposición de varios puntos [3]:

- No se prueba a partir de los requisitos, se prueban los requisitos. Probar un sistema es comparar lo que hace realmente con lo que debería hacer. Esto es que con las pruebas se debe validar si el comportamiento real del sistema cumple con los requisitos establecidos..

- Una buena ingeniería de pruebas produce buenos requisitos y una buena ingeniería de requisitos produce buenas pruebas. Si los requisitos son pobres o inexistentes se deben hacer las pruebas que sean posibles, ya que ayudarán a definir los requisitos. Las pruebas pasarían a ser una descripción de lo que el sistema hace. No es la manera óptima de proceder, pero siempre pueden existir excepciones.

- La actividad de análisis y de diseño de casos permite detectar errores en los requisitos. Un buen plan de pruebas ponen de manifiesto requisitos ambiguos, incompletos, vagos, etc.

- Un pequeño cambio en requisitos puede provocar grandes cambios en el desarrollo.Es necesario probar todos los cambios para comprobar que el sistema hace lo correcto en las situaciones afectadas por el cambio y hacer pruebas de regresión.Ayudar al usuario a que prepare sus casos de prueba de aceptación es un modo de evitar cambios de última hora en los requisitos.

- No es imposible probar sin requisitos, simplemente es más difícil. Si no se conociesen los requisitos, se puede probar, pero desde una postura de "oráculo", en la que se asume que el ingeniero conoce las respuestas correctas. Pero ojo, si se considera cierto que los ingenieros de pruebas no necesitan los requisitos, se puede argumentar que lo que hacen no son exactamente pruebas, que no se prueba sino que se "explora" el sistema. Esto se ha formalizado en las "Pruebas de Exploración".

- En casos excepcionales en los que haya una especial prisa en la finalización de un proyecto, en cuanto haya unos requisitos mínimos definidos, sería posible comenzar con el diseño de los casos de prueba. Eso sí, cualquier modificación de los requisitos implicaría la revisión y modificación de los casos de prueba diseñados, pudiendo ampliar considerablemente el tiempo total.

Desde las primeras clases que recibí en la Universidad, hasta en el día a día en la oficina, siempre está muy presente (y de vez en cuando alguien lo utiliza como ejemplo) un dibujo muy explicativo sobre lo que supone los requisitos y su interpretación y codificación. Se trata de la figura 2-4 en la que podemos observar con claridad que tanto el cliente como el desarrollador están muy implicados en la fase de requisitos y de cuyo perfecto entendimiento depende el éxito del proyecto, pues al fin y al cabo, la calidad de un proyecto se mide en gran parte en base a la satisfacción del cliente.

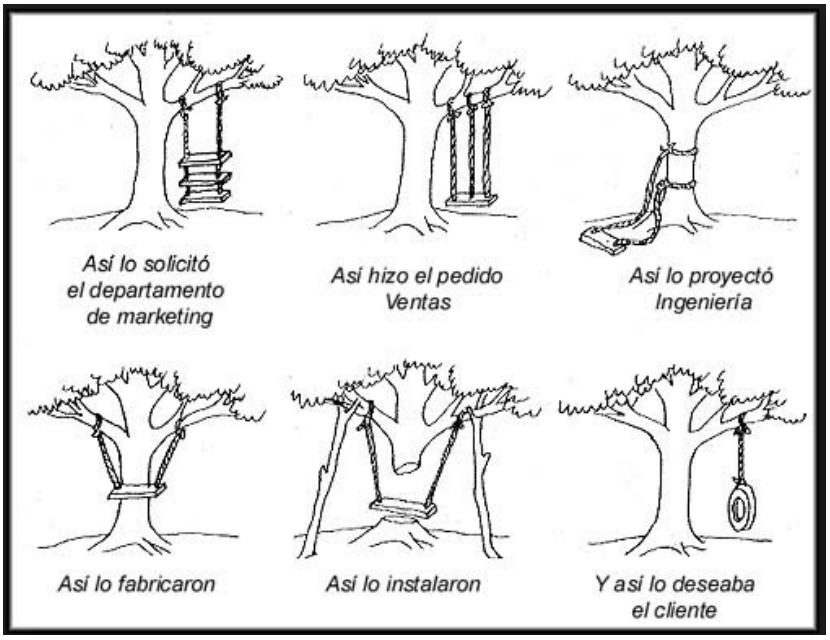


Figura 2-4: Requisitos

Siempre se pueden afianzar conocimientos exponiendo varias definiciones, en este caso de requisitos:

- Un requisito es la circunstancia o condición necesaria para algo.
- IEEE Standard Glossary of software Engineering Terminology:
    - Condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.
    - Condición o capacidad que debe estar presente en un sistema o componentes de sistema que satisfaga un contrato, estándar, especificación u otro documento formal.
    - Una representación documentada de una condición o facilidad.
- Una descripción de lo que el sistema debería hacer.
- En la ingeniería clásica, los requisitos son los datos de entrada en la fase de diseño del producto, determinando qué debe hacer el sistema, pero no cómo.

Existen una serie de características que los requisitos deben cumplir:

- **Necesario**: un requisito debe ser imprescindible para que el producto final sea como se espera.
- **No ambiguo**: debe expresarse con la suficiente claridad como para que su significado sea unívoco con el fin de evitar malinterpretaciones.
- **Conciso**: se refiere a la claridad con la que un requisito debe estar definido. Consiste en utilizar lenguajes no técnicos de manera que cualquiera pueda entender su significado.
- **Consistente**: un requisito debe ser único y no entrar en conflicto con el significado de otro requisito.
- **Completo**: un requisito debe contener toda la información necesaria para dar explicación al mismo. Esto es, que no se debe completar con información externa al requisito en sí.
- **Alcanzable**: un requisito debe ser viable en cuanto a la posibilidad de realización en base al tiempo, coste y recursos disponibles.
- **Verificable**: es la capacidad a que un requisito sea verificable, es decir, a comprobar a que ha sido satisfecho o no.

Hay que tener en cuenta que estas características no se pueden medir por ningún sistema y que existen métricas que ayudan a verificar si los requisitos de un determinado proyecto cumplen las características expuestas.

A modo de conclusión sobre los requisitos diré que se trata de la etapa más crucial dentro de un proyecto, pues de una buena definición de requisitos depende una buena interpretación de los mismos y por tanto el éxito.

2.9 SQA

En todo proyecto software siempre se habla de que se quiere que sea de calidad, pero ¿qué es exactamente la calidad en el software? Se trata de un concepto algo abstracto y por ello no muy fácil de concretar, de hecho, es algo subjetivo, lo cual lo complica aún más. A lo largo de este apartado, iremos añadiendo información sobre lo que supone la calidad de un software para definir SQA. Para empezar podemos quedarnos con una definición de calidad de entre todas las que existen, en este caso, escojo la dada por Pressman [1] "La calidad del software se define como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente".

Tratándose de ingeniería lo que se ha hecho es elaborar una serie de procedimientos y métodos que ayudan a definir bajo los mismos parámetros el concepto de la calidad del software. Podemos decir que la *garantía de calidad del software* o lo que es lo mismo *Software Quality Assurance* (**SQA**) es un conjunto de actividades planificadas y sistemáticas que ayudan a que el producto satisfaga los requisitos establecidos. Para ello se emplean métodos, procedimientos y herramientas mediante las cuales se aporta fiabilidad al producto cuyo objetivo final es que sea de calidad.

Aún a pesar de lo que hemos explicado hasta el momento sobre la calidad, quizá esta frase extraída de ISTQB [2] puede ayudarnos a aclarar ideas: "Que la calidad del software sea un factor decisivo de éxito es difícil de ver, pero que la falta de calidad es un factor decisivo de fracaso, es un tema bastante claro".

Hay tres aspectos muy importantes con relación al aseguramiento de la calidad del software, tal y como dijo Wiegers 1990 [1]:

- La calidad no se puede probar, se construye.
- El aseguramiento de la calidad del software no es una tarea que se realiza en una fase particular del ciclo de vida de desarrollo.
- Las actividades asociadas con el aseguramiento de la calidad del software deben ser realizadas por personas que no estén directamente involucradas en el esfuerzo de

desarrollo.

Todo equipo de trabajo de pruebas que forme parte de SQA debe tener como objetivo la depuración del software creado para obtener la calidad que el cliente espera del producto. O lo que es lo mismo:

- Se trabaja en la localización de errores en las fases tempranas del software para que su corrección sea lo antes posible.
- Todos los equipos se rigen bajo unos estándares de trabajo que consolidan el producto final.

Podemos concluir que la calidad se planea. Está claro que el concepto de calidad no es algo que se obtenga de una forma automática o espontánea. Es algo que se busca y es por ello que se debe planificar, construir e implantar en el producto a lo largo de todo el ciclo de vida.

2.10 Estándares de Calidad de Software

Los estándares de calidad de software son valores empleados dentro de una técnica que se utilizan para la medición de la calidad del software. Gracias a los estándares se puede medir la calidad de un determinado software bajo los mismos criterios en diferentes empresas o diferentes lugares del mundo. Así la calidad pasa a ser un parámetro más objetivo y más fácilmente medible [2].

En primer lugar, nombramos la norma **ISO / IEC 9126** ya que principalmente nos basaremos en ella. Según **ISO / IEC 9126** la calidad de software es la totalidad de propiedades y características de un producto de software referidas a su aptitud para satisfacer necesidades explícitas o implícitas. Este estándar se centra en la calidad del software en sí y sus características como programa.

El aseguramiento de la calidad (QS / QA) diferencia entre:

- *Medidas constructivas*: para evitar errores durante la construcción del software.
- *Métodos analíticos*: para la detección de errores después de construir el software.

La calidad del software según la ISO / IEC 9126 contiene un conjunto estructurado de características, que son:

- *Funcionales*: Idoneidad, Precisión, Conformidad, Interoperatividad, Seguridad.
- *No Funcionales*: Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad.

Dichas características han quedado detalladas páginas atrás.

En segundo lugar, nombramos el estándar **IEEE Std 610**. Para el estándar **IEEE Std 610**la calidad es el grado en el que un componente, sistema o proceso alcanza los requisitos especificados y/o las necesidades y expectativas del usuario o cliente. Esta norma se centra más en que el software se ajuste a los requisitos especificados.

3 Paradigmas de las Pruebas

En este capítulo se exponen las diferentes técnicas de diseño de casos de prueba. A continuación, se detalla el ciclo de vida al completo de las pruebas de integración. Y por último, se realiza una comparativa entre las dos fuentes de información de la teoría de este proyecto, que son Pressman e ISTQB.

3.1 Técnicas Dinámicas de Diseño de Pruebas

En primer lugar, quiero dejar constancia de que existe otro tipo de diseño de casos de prueba al que se denomina **técnicas estáticas** o **pruebas estáticas** o **análisis estático**, el cual engloba procedimientos que requieren un análisis manual del objeto de prueba, como puede ser la lectura, respaldado por herramientas pero siempre sin llegar a ejecutar el objeto en sí de prueba.

Las diferencias con las técnicas dinámicas es que en las técnicas estáticas se prescinde de los datos de prueba y de la ejecución. Y lo que se localizan son defectos y errores pero no fallos, debido a la ausencia de ejecución. He aquí el motivo por el que no las detallaré en el proyecto, y procedo directamente a exponer con el mayor detalle posible las técnicas dinámicas, puesto que en las pruebas de integración va intrínseca la ejecución.

En las **técnicas dinámicas** o **pruebas dinámicas** o **análisis dinámico** de casos de prueba [2] se compara el resultado esperado y el resultado obtenido, donde los resultados obtenidos se obtienen mediante la ejecución de los casos de prueba sobre el objeto de prueba.

Las pruebas dinámicas son complementarias a las pruebas estáticas. La principal diferencia entre ambas técnicas reside en la ejecución, es por ello que para las pruebas estáticas no son necesarios datos de prueba al no existir ejecución, mientras que para las pruebas dinámicas los datos son imprescindibles, pues no se conciben sin ejecución del objeto de prueba. Por otro lado, a priori las pruebas dinámicas se ejecutarán más tarde en el tiempo que las estáticas.

3.1.1 Pruebas de Caja Blanca

En las pruebas de caja blanca se prueba la estructura del programa. El probador conoce la estructura del programa y su código. Es por ello que los casos de prueba se especifican en base a la estructura del programa / código del programa. Este tipo de pruebas también son conocidas por **técnicas intrusivas**, **de cobertura** o **estructurales**.

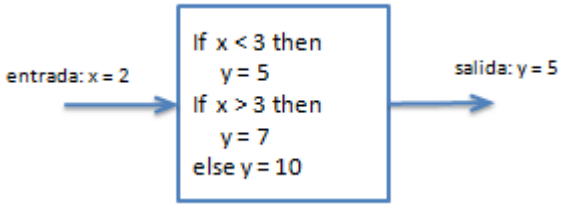


Figura 3-1: Técnicas de Caja Blanca

Mediante los métodos de prueba de la caja blanca, se pueden obtener casos de prueba que garanticen que se ejercitan:

- Por lo menos una vez todos los caminos independientes de cada módulo.
- Todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Todos los bucles en sus límites y con sus límites operacionales.
- Las estructuras internas de datos para asegurar su validez.

En las técnicas de caja blanca se deben ejecutar todas las partes del programa, al menos una vez durante las pruebas, de ahí lo de intrusivas, pues teóricamente deben pasar como mínimo una vez por cada una de las estructuras del sistema.

Este tipo de pruebas son las que realiza un desarrollador ya que es quien verdaderamente conoce la estructura interna del sistema y por descontado se deben realizar en fases tempranas del ciclo de vida.

3.1.1.1 Cobertura de Sentencias o Técnica del Camino Básico

Cada sentencia del programa y su ejecución respectiva constituyen el centro de la observación. La base de esta técnica reside en un gráfico de flujo de control. Cada sentencia se representa mediante un nodo y el flujo de datos entre sentencias mediante una arista. El objetivo de las pruebas consiste en ejecutar un número predeterminado de sentencias.

*Medida C_0 (Cobertura de sentencias) = (nº sentencias ejecutadas / nº total sentencias) *100*

De esta forma mientras para un if sería suficiente con un único caso de prueba para conseguir una cobertura de sentencia del 100%. Si se tratase de una estructura if-else harían falta dos casos de prueba para obtener la cobertura del 100%.

Si se utiliza esta técnica habría que considerar dos puntos:

- Si existen sentencias muertas (sentencias que no pueden ser alcanzadas) no será posible alcanzar una cobertura de sentencias del 100%.
- Las sentencias no existentes, no podrán probarse. Esto es que si faltase alguna sentencia por incluir, dicho fallo no podrá ser detectado en este tipo de pruebas.

3.1.1.2 Cobertura de Decisiones o Ramas

En la cobertura de decisiones o ramas lo que se tiene en cuenta es el flujo de control, esto es que no solo los nodos, sino también las aristas. Por tanto, todas las aristas del gráfico de flujo de control deben ser recorridas al menos una vez.

*Medida C_1 (Cobertura de ramas) = (nº ramas recorridas / nº total ramas) *100*

En esta ocasión, para recorrer una estructura if habría que recorrer dos ramas y por tanto se necesitarían dos casos de prueba para conseguir una cobertura de ramas del 100%. Mientras que para recorrer una estructura if-else, al ser también dos caminos y por tanto dos ramas, la cobertura de ramas del 100% se obtendría también con dos casos de prueba.

Algunos de los inconvenientes de esta técnica serían:

- No será posible localizar si faltan sentencias, pues solo se trabaja sobre lo existente.
- Es un proceso muy costoso de aplicar en condiciones complejas.
- Se trata de un método insuficiente para la comprobación de bucles.
- No se tienen en cuenta las dependencias entre bucles.

Para comparar ambas métricas, expongo un ejemplo a través de la siguiente figura:

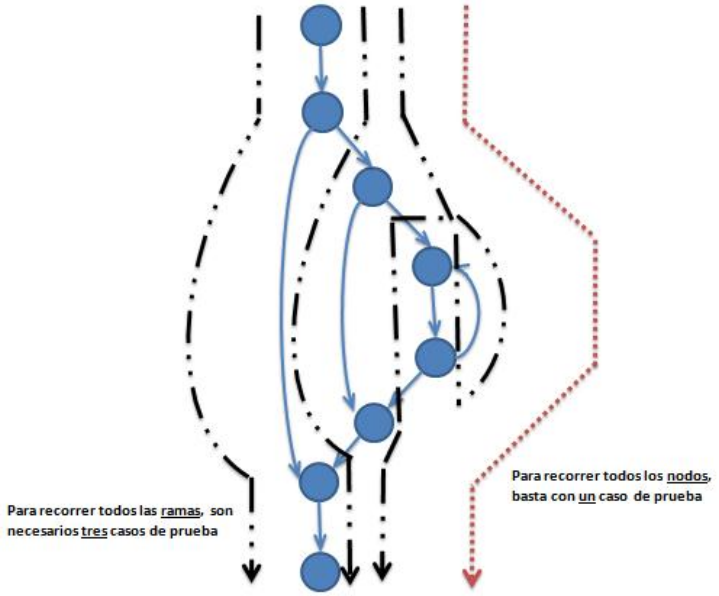


Figura 3-2: Cobertura Sentencias VS Cobertura Ramas

La cobertura por ramas por lo general, suele necesitar un mayor número de casos de prueba que con cobertura de sentencias para alcanzar una cobertura del 100%. Para recorrer todas y cada una de las ramas es inevitable que algunas de las aristas se recorran en más de una ocasión.

3.1.2 Pruebas de Caja Negra

En este tipo de pruebas la estructura del programa es irrelevante o desconocida, o dicho de otra forma, es transparente para el probador ya que la importancia de este tipo de pruebas reside en la funcionalidad como tal. Es por ello, que en muchas ocasiones se las conoce como **pruebas funcionales**. Las pruebas se reducen en su totalidad a las especificaciones / requisitos, se trata de pruebas en las que se valida el comportamiento de entrada y de salida de un cierto elemento. En ellas se intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Figura 3-3: Técnicas de Caja Negra

Las técnicas de caja negra se tienden a aplicar durante fases posteriores a donde se aplican las técnicas de caja blanca, ya que en las de caja negra se ignora intencionadamente la estructura de control, centrando la atención en el campo de información.

Es importante conocer que las pruebas de caja negra no son una alternativa a las técnicas de prueba de caja blanca, sino que son un enfoque complementario para que aplicando ambas técnicas se puedan descubrir diferentes tipos de errores y de diferente índole.

Este tipo de pruebas están enfocadas al tipo de pruebas que realizaría un probador independiente de desarrollo, el cual procuraría depurar la funcionalidad del sistema y su comportamiento sin tener conocimiento alguno sobre la parte interna del sistema.

Cabe destacar que las pruebas funcionales son las de mayor peso dentro de las pruebas, por ello, los procedimientos de caja negra son siempre utilizados.

3.1.2.1 Clases o Particiones de Equivalencia

Las particiones de equivalencia son un subconjunto de datos de entrada al sistema, establecido según las especificaciones, para los que el sistema tiene el mismo comportamiento. Evitan que haya que probar todos los posibles datos de entrada al sistema, puesto que el sistema debe reaccionar de igual modo ante todos los datos de un mismo tipo y bastaría probar con uno de ellos para asegurar que el sistema va a funcionar como se espera para todos los valores que forman parten de la partición de equivalencia.

Las particiones se definen teniendo en cuenta que si una condición de entrada especifica un rango de entrada, se define una partición de entrada válida y dos no válidas. Esto serían tres casos de prueba: un caso de prueba para un valor que esté dentro del rango, otro caso para un valor inferior al rango y un tercer caso de prueba para un valor de entrada mayor que el rango dado. De esta forma, el caso creado para probar el resultado que se obtiene con un dato de entrada dentro del rango sería la entrada válida, pues es la esperada para producir un caso de prueba OK. Y los casos con los datos de entrada menor y mayor que el rango serían los casos KO, pues están fuera de los valores esperados.

Para el supuesto en el que no se trate de un rango de valores como dato de entrada, sino un valor en concreto, se define una partición de equivalencia válida y otra no válida. O lo que es lo mismo, una partición con el valor válido sería un caso de prueba OK, y el caso KO sería con cualquier otro valor que no sea el especificado y que por tanto se espera que produzca un error controlado en el sistema.

Una vez establecidas las clases de equivalencia, el siguiente paso es crear casos de prueba a partir de ellas, para ello, básicamente se cumplirá la norma de que habrá que crear al menos un caso de prueba por cada partición definida. Además, será necesario combinar las clases de equivalencia para generar distintos casos de prueba y así cubrir toda la casuística del proyecto.

La utilización de clases de equivalencia tiene una serie de ventajas claras:

- Se realiza una determinación sistemática de casos de prueba, es decir, con un número mínimo de casos de prueba se maximiza la cobertura de pruebas.
- La descomposición en clases de equivalencia en base a las definiciones o especificaciones abarca de muy buena manera los requisitos funcionales.
- La priorización de las clases de equivalencia puede servir para priorizar los casos de prueba.
- Las pruebas de las condiciones de excepción definidas quedan aseguradas por los casos de prueba negativos o KOs.

3.1.2.2 Valores Límite

La experiencia en pruebas ha permitido reconocer zonas en las que es muy posible que se produzcan fallos, y éstas precisamente son las zonas donde se encuentran los valores límite de los rangos de valores. Mediante esta técnica se consigue comprobar el funcionamiento del sistema ante valores límite de rangos de entrada.

Dado un rango de valores delimitado por dos valores, los valores límite son precisamente los valores que delimitan el rango, y los casos de prueba se definirían para los valores que marcan los dos límites del rango y además, casos de prueba para los valores inmediatamente mayores y menores que los límites. Lo que serían seis casos de prueba o lo que es lo mismo, para un rango delimitado por x e y, los casos serían los siguientes:

- Casos OK: Valor x; Valor x+1; Valor y; Valor y-1
- Casos KO: Valor x-1; Valor y+1

Las mismas reglas deben aplicarse para las especificaciones de salida. Si las estructuras internas tienen límites preestablecido hay que asegurarse de diseñar casos que ejerciten la estructura de datos en sus límites. En esta técnica no se seleccionan elementos representativos como en las particiones de equivalencia, sino que el criterio es únicamente la condición de límite de los valores.

Si se trata de un conjunto de valores en lugar de un rango, no se podrá aplicar valores límite, pues con dos clases de equivalencia (pertenece al grupo y no pertenece al grupo) estaría cubierta la casuística a probar.

3.1.2.3 Combinar Particiones de Equivalencia con Valores Límite

Sin duda alguna, una buena técnica para realizar casos de prueba sería combinando las técnicas de particiones de equivalencia con los valores límites. Esto es, obtener todos los casos de prueba en base a las particiones de equivalencia y añadir a este plan de pruebas, todos los casos que se obtienen aplicando al mismo enunciado la técnica de los valores límite. O lo que es lo mismo, el rango de valores contenidos en una clase de equivalencia más los valores que se pueden identificar en los límites de dicho rango de valores definirían la casuística de pruebas a realizar. El resultado sería un plan de pruebas completo, sin repeticiones y robusto.

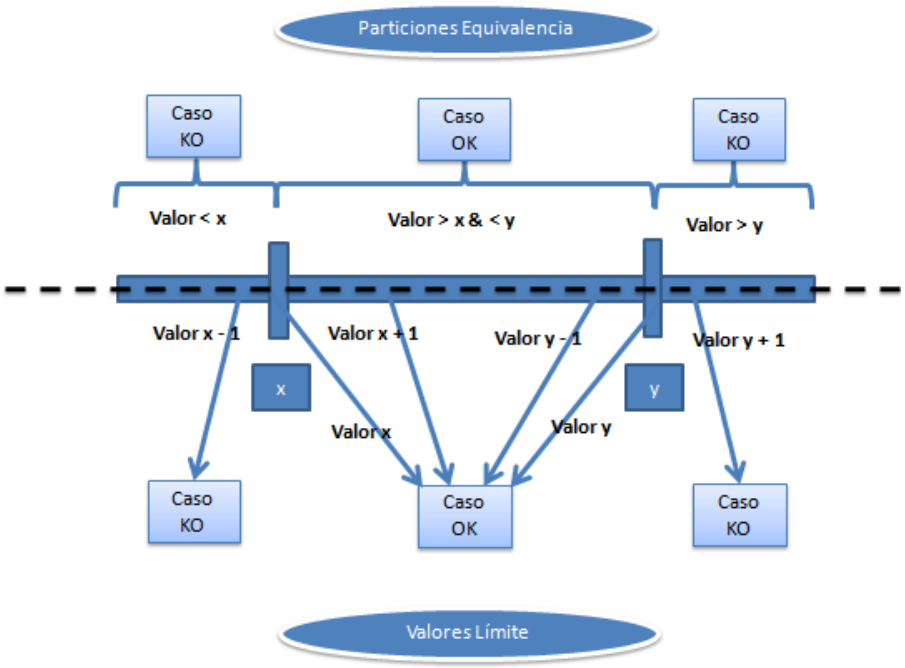


Figura 3-4: Clases de Equivalencia VS Valores Límite

3.1.3 Técnicas Basadas en la Experiencia

Las técnicas basadas en la experiencia, son un tipo más de técnica de prueba dinámica, que consisten en la especificación de casos de prueba de manera intuitiva. Consiste en un procedimiento en el que los casos de prueba son elaborados sin un modo de proceder claramente metódico, en base a la intuición y la experiencia del probador. La especificación intuitiva de casos de prueba se denomina también como pruebas orientadas a **puntos débiles** o **error guessing**.

Para realizar un buen diseño con técnicas basadas en la experiencia, cobra una importancia relevante la formación y actitud del probador, pues de alguna forma la técnica residirá en su cabeza más que en ningún manual. Por ello, es necesario que el probador tenga una amplia experiencia y conocimientos para poder aplicar una intuición coherente en base a su sabiduría en el diseño de los casos de prueba. Al tratarse de experiencia, las bases de esta técnica se centran

en diseñar casos de prueba en los puntos donde hayan aparecido ya errores frecuentemente en el pasado, además de intuir donde podrían aparecer errores en un futuro.

Lo cierto es que en la práctica sólo se utiliza para completar un conjunto de casos de prueba disponibles y diseñados con otros tipos de metodologías. Ya que se trata de una técnica que no es suficiente para procedimientos de pruebas sistemáticos. Sin embargo, a menudo se obtienen casos de prueba basándose en la experiencia que no es posible especificar mediante otros procedimientos.

Las técnicas basadas en la experiencia son una modalidad nueva que con los años se ha establecido como una técnica más. De hecho, podemos ver cómo se trata de una técnica inexistente en Pressman mientras que ISTQB la introduce con total normalidad.

3.2 Ciclo de Vida de las Pruebas de Integración

De entre los modelos existentes, vamos a centrarnos en el que paso a detallar a continuación. Se trata de la Fase de Pruebas de integración, que está incluida en la rama derecha del modelo genérico en V. Sirve de referencia la figura 3-5, para comprender que la fase de las pruebas de integración, a su vez, dispone de diferentes etapas dentro de su propio ciclo de vida.

El ciclo de vida de las pruebas de integración se compone por cinco fases, las cuales son más o menos comunes a todos los ciclos de vida: planificación, análisis y diseño, implementación y ejecución, evaluación y cierre. Y llaman quizá, la atención dos fases más en las que se trabaja en paralelo desde la primera etapa hasta la última, se trata de las fases de monitorización y control y mejora continua. Ayudan, sin duda a la optimización del resto de fases y permiten documentar las fases de un proyecto desde su inicio hasta el final.

Veremos el detalle de cada una en las siguientes líneas [3]:

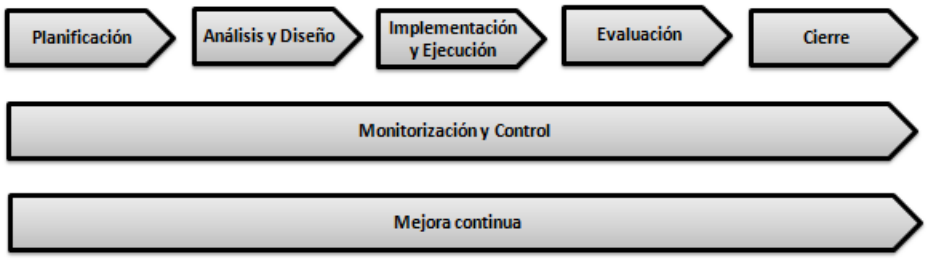


Figura 3-5: Fases del Ciclo de Vida de las Pruebas de Integración

3.2.1 Planificación

La fase de Planificación tiene como objetivo principal determinar la viabilidad, el alcance, los riesgos y el objetivo de las pruebas, para realizar una previsión de los recursos necesarios para la realización de las pruebas, así como establecer un calendario de las tareas a realizar. Se compone de las siguientes actividades:

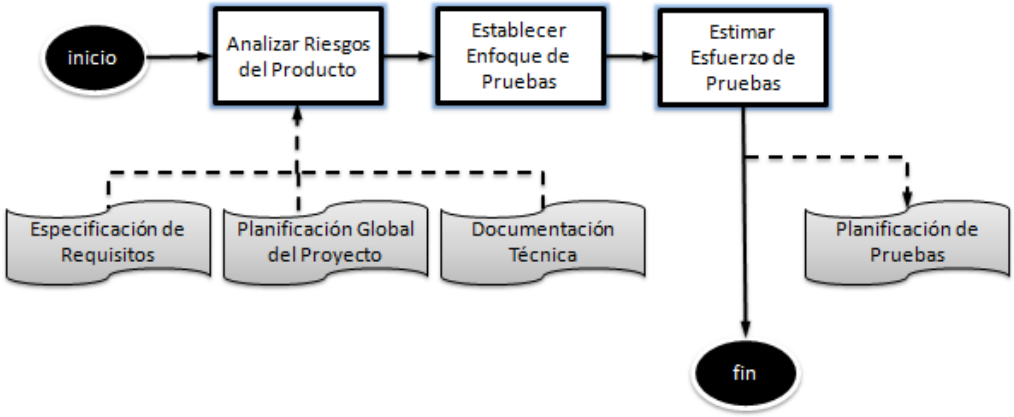


Figura 3-6: Diagrama de Actividades en la Planificación de Pruebas

- **Analizar los riesgos de producto:** Se trata de analizar el producto que va a ser probado para identificar los riesgos asociados al mismo. Se entiende como riesgos de producto aquellos aspectos del software o en el sistema que suponen un riesgo para la calidad del producto final, la posibilidad de que ocurra un evento con consecuencias no deseadas,

un problema potencial. El nivel de riesgo vendrá determinado por la probabilidad de que ocurra el evento adverso y por el impacto o daño resultante que pudiese ocasionar.

Los riesgos asociados a un proyecto que puedan influir en su éxito deben ser gestionados. En primer lugar, habrá que estimar la probabilidad de que ocurra y el daño potencial. En segundo lugar, se implementarán medidas para tratar los riesgos identificados, preparando de forma activa medidas para reducir la probabilidad y/o el daño potencial. Existen diferentes tipos de riesgos, los más típicos son:

- *Factores organizativos*: Pueden ser por falta de recursos o de capacidad, insuficiente cooperación entre departamentos o estimaciones nada realistas, ...
- *Factores técnicos*: Se producen por disponer de unos requisitos incorrectos, incompletos o no realistas, por una escasa calidad del diseño, código o pruebas, por disponer de herramientas, metodologías o procedimientos nuevos que entorpezcan el día a día del personal, ...
- *Riesgos asociados al entorno*: Son debidos a cambios debidos a requisitos legales, problemas de soporte con el suministrador, mala calidad en general del entorno, ...

Un análisis de los riesgos desde fases tempranas del ciclo de vida del producto supone una gestión continuada de los mismos, con continuas actualizaciones de los riesgos del momento para tratarlos según su prioridad.

- **Establecer el enfoque de las pruebas**: A partir del análisis de riesgos del producto se debe delimitar el alcance de las pruebas y una estrategia inicial de ejecución. Asimismo es necesario establecer los criterios de entrada a ejecución de pruebas, los criterios de aceptación o rechazo de producto y los de suspensión y reactivación de las pruebas.

- **Estimar esfuerzo de pruebas**: Es una aproximación en coste, tiempo y funcionalidad a probar sobre un proyecto. Con el input recibido de las actividades anteriores podremos desarrollar la planificación de las pruebas a realizar, que contenga el calendario, las necesidades de entorno y formación, los roles y responsabilidades y los riesgos del proyecto.

3.2.2 Análisis y Diseño

La fase de Análisis y Diseño tiene como objetivo principal analizar la documentación en base a la cual se van a generar los casos de prueba, aplicar las técnicas de diseño de casos de prueba adecuados, crear los casos de prueba trazándolos con requisitos y riesgos y revisión de los mismos. Las actividades que lo componen son:

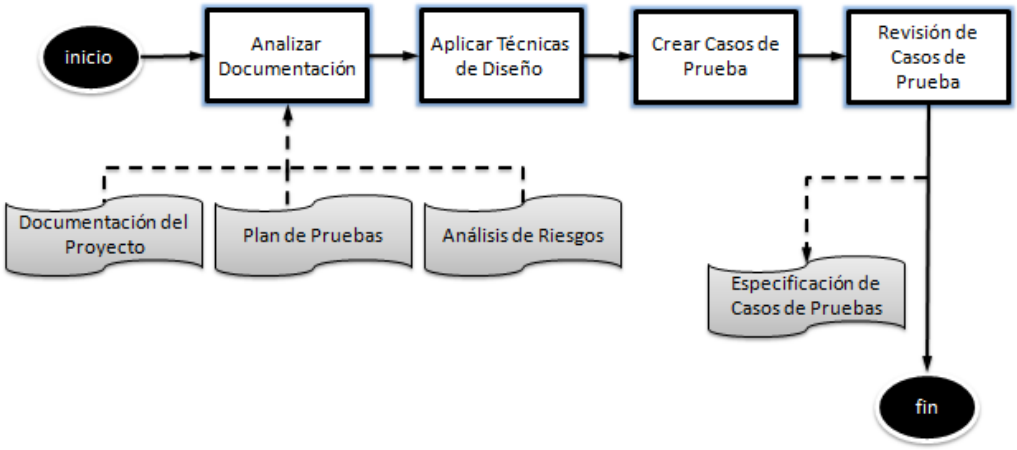


Figura 3-7: Diagrama de actividades en la fase de Análisis y Diseño

- **Analizar documentación**: El objeto de esta tarea es, valga la redundancia, analizar la documentación de entrada que está disponible para las pruebas y que al mismo tiempo sea considerada necesaria según el nivel de las pruebas que se tenga que realizar. En esta actividad se analiza la base de pruebas (documentación) para determinar lo que hay que probar, es decir, las condiciones de prueba (ej.: una función, transacción, elemento estructural, característica de calidad). Engloba las siguientes tareas:
    - Recopilación de la información útil del proyecto.
    - Catalogación y estructuración de la información.
    - Identificación de carencias y defectos documentales.
    - Búsqueda de soluciones a las carencias y defectos documentales.

- **Aplicar técnicas de diseño**: La finalidad de esta actividad es identificar y aplicar las técnicas de diseño estructurado de casos de prueba que sean más apropiadas. Engloba las siguientes tareas:
    - Identificar las técnicas aplicables a cada bloque de pruebas.
    - Elaborar la casuística a alto nivel que da cobertura a cada bloque.
- **Crear casos de prueba**: Creación de una casuística detallada de pruebas con trazabilidad con requisitos y riesgos:
    - Crear la especificación de los casos de prueba.
    - Trazar los casos de prueba con requisitos.
    - Trazar los casos de prueba con los riesgos identificados.
- **Revisión de casos de prueba**: Creación de una casuística detallada de pruebas con trazabilidad con requisitos y riesgos.
    - Detección de defectos en la especificación de casos de prueba
    - Corrección de los defectos encontrados.

3.2.3 Implementación y Ejecución

La fase de Implementación y Ejecución tiene como objetivo principal ejecutar los casos de prueba, registrar y analizar resultados, comparar resultados obtenidos con esperados y repetir las actividades de prueba según la acción que se lleve a cabo para cada discrepancia. Las actividades que lo componen son:

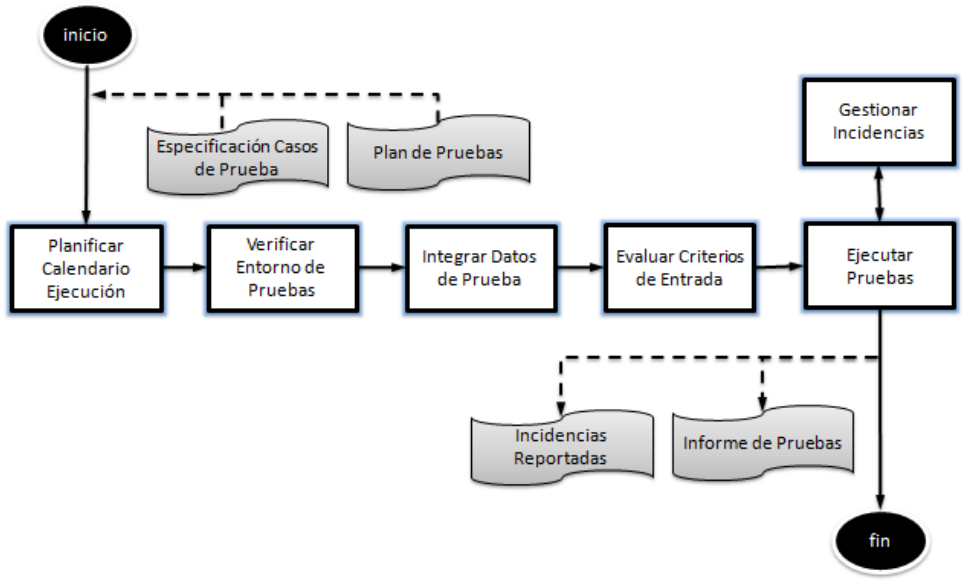


Figura 3-8: Diagrama de Actividades en la Fase de Implementación y Ejecución

- **Planificar calendario detallado de ejecución**: Definir un enfoque detallado de pruebas, que puede ser en base a una priorización de casos y establecer el calendario de ejecución de los casos de prueba.

- **Verificar entorno de pruebas**: Validar el entorno previsto de ejecución de pruebas para comprobar su correcto funcionamiento.

- **Integrar los datos de pruebas**: Hacer provisión de los datos de prueba que se van a utilizar en la ejecución de las pruebas.

- **Evaluar criterios de entrada**: El objetivo de esta actividad es analizar los criterios de entrada a la ejecución de pruebas definidos en el Plan de Pruebas para validar que se pueden comenzar la ejecución de las mismas.

- **Ejecutar las pruebas**: Engloba las siguientes tareas:

    - Ejecutar casos de prueba

    - Registrar y analizar los resultados de ejecución.

    - Reportar incidencias encontradas.

- **Gestionar incidencias reportadas**: El objetivo de la gestión de incidencias es tener una subproceso para gestionar adecuadamente de forma ordenada y sistemática las incidencias de producto encontradas durante la ejecución de casos de prueba. Engloba las siguientes tareas:
    - Revisar las incidencias con las áreas implicadas.
    - Actualizar estado de las incidencias.
    - Volver a ejecutar los casos de prueba necesarios (repruebas y regresión).

3.2.4 Evaluación

La fase de Evaluación tiene como objetivo principal evaluar los criterios de finalización la ejecución de las pruebas respecto de los objetivos definidos y elaborar el Informe final de pruebas. Las actividades que lo componen son:

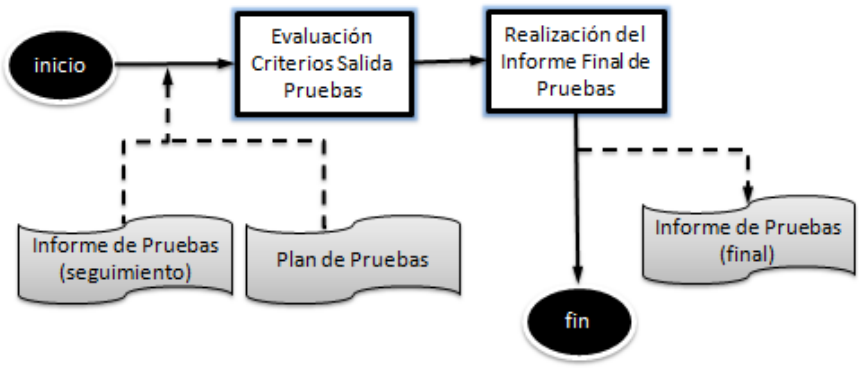


Figura 3-9: Diagrama Actividades en la Fase Evaluación

- **Evaluación criterios de salida de pruebas**: Medir la completitud de las pruebas con respecto a los criterios de salida de las pruebas.

- **Realización del Informe final de pruebas**: Obtención de un informe resumen del desempeño de las pruebas.

3.2.5 Cierre

La fase de Cierre tiene como objetivo principal recoger los datos de las actividades de prueba, cierre de los informes de incidencias, comprobación de que entregables han sido entregados, documento de aceptación, archivado de las pruebas, análisis de las lecciones aprendidas. Las actividades que lo componen son:

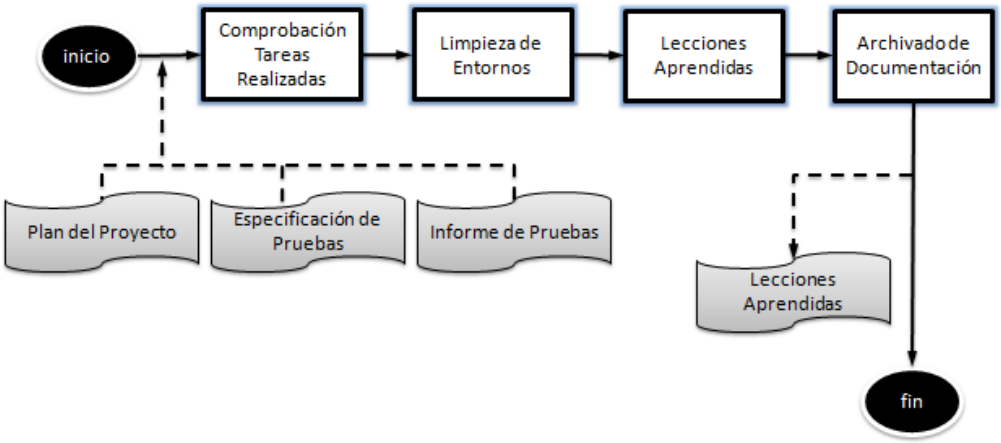


Figura 3-10: Diagrama Actividades Fase de Cierre

- **Comprobación de tareas finalizadas**: Verificar que todas las tareas planificadas de las distintas fases del proyecto se han realizado.

- **Limpieza de entornos**: El grupo de pruebas debe dejar los entornos de pruebas en situación lo más parecida posible a antes de comenzar el proyecto de pruebas.

- **Lecciones aprendidas**: Se debe documentar las lecciones aprendidas durante el proyecto de pruebas.

- **Archivado de información**: Se debe salvaguardar los productos de trabajo del proceso de pruebas convenientemente.

3.2.6 Monitorización y Control

La fase de Monitorización y Control tiene como objetivo principal monitorizar y controlar los aspectos planificados en el Plan de Pruebas. Las actividades que lo componen son:

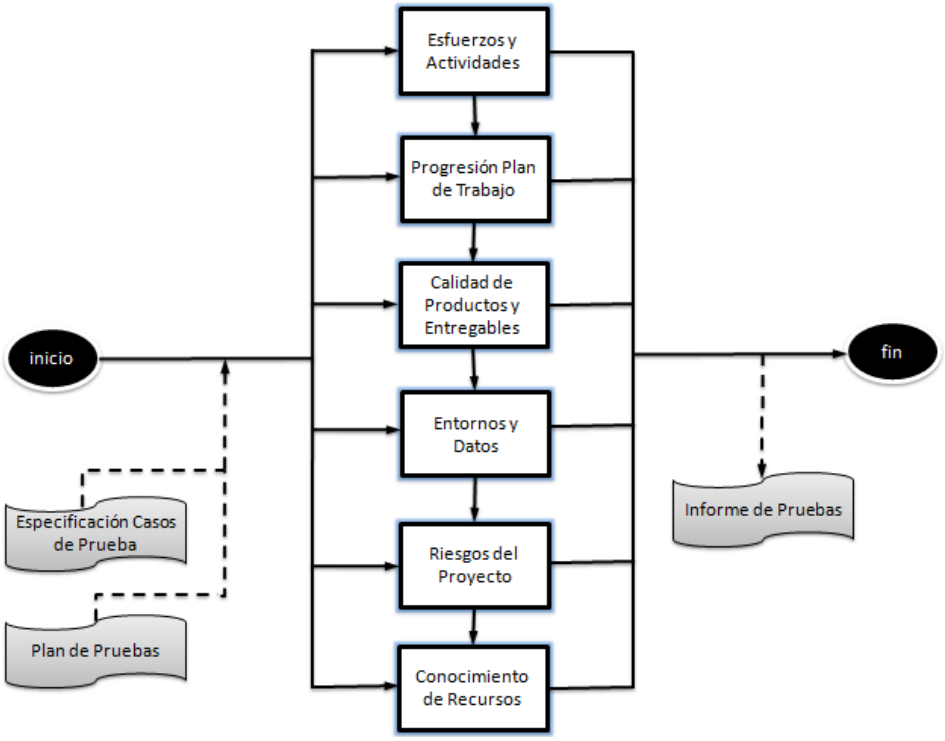


Figura 3-11: Diagrama Actividades de la Fase Monitorización y Control

- **Esfuerzos y Actividades**: Se trata de monitorizar y controlar los esfuerzos estimados frente a los planificados de cada una de las fases del ciclo de vida del proyecto.

- **Progresión Plan de trabajo**: Se trata de monitorizar y controlar la progresión del plan de trabajo tanto de diseño como de ejecución de casos de prueba.

- **Calidad de productos y entregables**: Se trata de monitorizar y controlar la calidad de los productos entregados durante el diseño y ejecución.

- **Entornos y Datos**: Se trata de monitorizar y controlar la disponibilidad de entornos antes y durante el proyecto de pruebas.

- **Riesgos del proyecto**: Se trata de monitorizar y controlar los riesgos del proyecto a partir de los identificados durante la planificación.

- **Conocimiento de Recursos**: Se trata de monitorizar y controlar el conocimiento que los recursos tienen de los distintos aspectos del proyecto (tecnología, metodología, etc.)

- **Acciones Correctivas**: Se trata de monitorizar y controlar las acciones correctivas que surjan durante la vida del proyecto.

3.2.7 Mejora Continua

La fase de mejora continua es una fase paralela al resto de fases, de inicio a fin, cuyo objetivo es mejorar los procesos y procedimientos de dicha metodología. Las actividades que lo componen son:

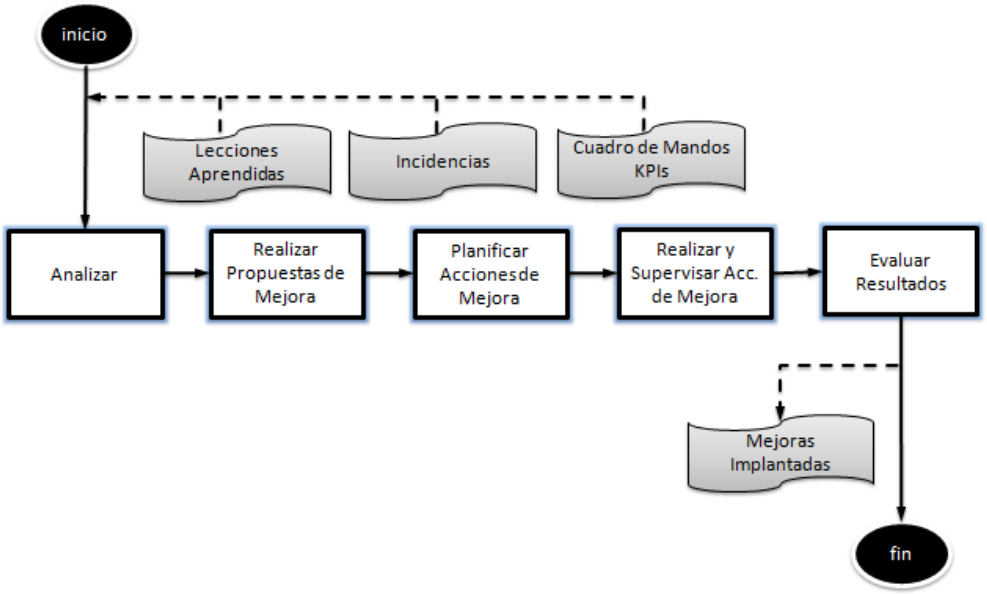


Figura 3-12: Diagrama de Actividades de la Fase de Mejora Continua

- **Analizar.** Se analizarán las distintas entradas a Mejora Continua con el fin de evidenciar defectos, problemas y oportunidades de mejoras dentro de la Metodología de Pruebas tanto en la parte metodológica como en su implementación práctica.

- **Realizar propuestas de mejora.** Una vez detectado y delimitados los aspectos del proceso de pruebas que deben ser mejorados, y asignadas la responsabilidad sobre quienes deben realizar las propuestas de mejora, las personas responsables asociadas a tales propuestas diseñaran las acciones de mejora.

- **Planificar acciones de mejora.** Una vez desarrollada cada propuesta de mejora, esta se implementará a través de acciones de mejora que deben ser planificadas detalladamente.

- **Realizar y supervisar acciones de mejora.** En base a la planificación establecida se ejecutarán las acciones de mejora. Durante toda la implantación de la mejora se realizará un seguimiento del proyecto en base al esquema establecido en la planificación.

- **Evaluar resultados:** Una vez desplegada totalmente la propuesta de mejora debe realizarse una evaluación de los resultados obtenidos. La referencia fundamental debe ser si los objetivos perseguidos con la mejora desplegada se han alcanzado o no.

3.3 Roger S. Pressman VS ISTQB

Uno de los objetivos del estudio de este proyecto consiste en averiguar las diferencias que pueden existir entre los conceptos que plantea el clásico Pressman sobre la calidad del software y más en concreto sobre las pruebas integradas, frente a lo que se indica en una certificación mucho más actual como es ISTQB.

En rasgos generales, ambos hablan mucho sobre la importancia de la calidad del software. La gran diferencia entre los dos es que Pressman obvia totalmente que las pruebas tienen su propio ciclo de vida, el cual se detalla al milímetro en ISTQB. Pressman expone cómo se debería crear un software nuevo para que sea de calidad. Mientras que ISTQB se centra en la realización de pruebas como parte fundamental en la elaboración de un software de calidad. Este punto evidencia que ISTQB trata una pequeña parte del conjunto del contenido de Pressman.

Sí coinciden en las técnicas de diseño de casos de pruebas, excepto en que, ISTQB añade una técnica más que Pressman no consideró en su momento, se trata de la experiencia. Además de las técnicas de caja blanca y caja negra, en ISTQB consideran que la experiencia en sí es una técnica más a tener en cuenta para poner en práctica. A título personal añadiría que desde luego la experiencia no sirve de nada si no está plena de conocimiento en pruebas y en ejecución de las mismas, y sobretodo que se haya formado teniendo en conocimiento las técnicas de diseño de casos de prueba de los que hablan tanto Pressman como ISTQB, por lo que a mi parecer considerar la experiencia como una técnica más, no es más que una acción de la pura lógica, un paso más en la evolución de la elaboración de software de calidad a lo largo del tiempo.

Otro punto en el que sí se localizan pequeñas diferencias respecto a lo que se muestra en Pressman frente a lo expuesto por ISTQB, es que en Pressman aunque con el lenguaje directo no se habla de las pruebas como una fase destructiva, puede verse que de alguna forma se trata de los comienzos de las pruebas como parte del proceso de calidad y entre líneas, al final, la

realización de las pruebas deja una sensación negativa. Mientras que en ISTQB en algún momento admite que las pruebas se pueden considerar destructivas pero se esfuerzan en poner todo el foco en aclarar que el objetivo final es común al del resto de las fases del ciclo del vida en V del software y se trata de un objetivo positivo, para finalizar con que las pruebas en su conjunto son constructivas.

Un ejemplo de la psicología negativa que ve Pressman sobre las pruebas es el siguiente párrafo [1]: "La prueba presenta una interesante anomalía para el ingeniero de software. Durante las fases anteriores de definición y de desarrollo, el ingeniero intenta construir el software partiendo de un concepto abstracto y llegando a una implementación tangible. A continuación, llega la prueba. El ingeniero crea una serie de casos de prueba que intentan "demoler" el software que ha sido construido. De hecho, la prueba es uno de los pasos de la ingeniería del software que se puede ver (por lo menos psicológicamente) como destructivo en lugar de constructivo. ¿Debe infundir culpabilidad la prueba? ¿Es realmente destructiva? La respuesta a estas preguntas es no! Sin embargo, los objetivos de la prueba son algo diferentes de lo que podríamos esperar."

Con este párrafo de Pressman la conclusión es rápida de obtener: las pruebas no son destructivas. Pero utilizando palabras como demoler aporta esa sombra de negatividad sobre las pruebas que ISTQB se esfuerza en hacer desaparecer, es aquí donde observo la diferencia más importante entre ambos. Mi conclusión más rotunda es que el objetivo de la prueba es común al de la creación del software, pues juntos pretenden conseguir una calidad digna de entregar a un cliente. El desarrollo no se puede concebir sin la prueba y viceversa, por esto mismo, en absoluto la prueba es destructiva, sino que es constructiva de un software de calidad.

Como punto final a este apartado diré que los resultados de la comparación no podrían agradarme más. Al parecer las bases de la ingeniería del software están muy bien asentadas y los valores aportados por Pressman siguen muy presentes a día de hoy y más en concreto en ISTQB. Realmente, todo el conocimiento de Pressman está incluido en ISTQB, vamos que perfectamente se podría decir que es ISTQB quien se basa en las definiciones dadas por Pressman. Desde luego, se sigue trabajando en ellas para mejorarlas, pero realmente los pilares son los mismos y esto da mucha estabilidad al mundo de la calidad del software. Dado que se trata de una ingeniería joven, ver que lo escrito hace unos años sigue siendo parte fundamental, aporta mayor fiabilidad a los cimientos sobre los que trabajamos los informáticos.

4 Caso de Estudio

En Reingeniería Web se mejoran las prestaciones de la aplicación Web de la empresa de telecomunicaciones para la que trabajamos y a la que tiene acceso cualquier internauta, ya sea cliente o no de la teleoperadora. En primer lugar, se crea una nueva forma de logado a la Web, totalmente transparente para el usuario de la Web pero que optimizará los tiempos de respuesta mejorando el servicio que se ofrece a los clientes. Y en segundo lugar, se modifican los procesos de envío de peticiones para que en las ocasiones en las que se produzca una indisponibilidad del sistema no tenga que ser el propio cliente quien reintente la operación, si no que sea el sistema, una vez más de manera transparente para el cliente, quien relance la operación pasados unos minutos con la intención de sortear la indisponibilidad que fuese.

En el caso práctico que presento en el proyecto final de carrera, participé activamente en el ciclo de vida del proyecto desde su planificación hasta las pruebas integradas y soporte a las pruebas de usuario. Desempeñé diferentes roles que me permitieron formar parte de la planificación de la release a la que pertenece el proyecto. Y un tiempo más tarde, realicé desde el diseño de los casos de prueba de integración hasta su ejecución y soporte a las pruebas de aceptación. Por eso no podía dejar pasar la oportunidad de exponer el detalle de todo el trabajo que refleja el ciclo de vida de las pruebas integradas de Reingeniería Web.

4.1 Gestión de la Configuración

Antes de empezar a entrar en materia sobre el caso de estudio, me parece de gran importancia situar el marco en el que se produce todo el trabajo. Se trata de exponer los pilares básicos antes de ocuparse de un proyecto y es la materia que cualquier tester debe conocer con el mayor detalle posible para trabajar con eficacia. Esto es conocer qué es middleware, cuáles son los diferentes entornos y cómo se procede con las instalaciones.

4.1.1 Middleware

Middleware es posiblemente el componente más importante de todos los que forman una integración de sistemas, ya que es el sistema que conecta unos sistemas con otros y además, consigue que se entiendan entre ellos, pues como es normal, cada sistema es independiente y utiliza su propio "lenguaje". Por tanto, middleware es el encargado de la transmisión de datos entre las diferentes aplicaciones, esto es, que es traductor y comunicador de datos mediante la invocación de servicios estándar o servicios middleware. En él se publicarán los mensajes entre los diferentes sistemas y se encarga de distribuir tanto la petición como la respuesta.

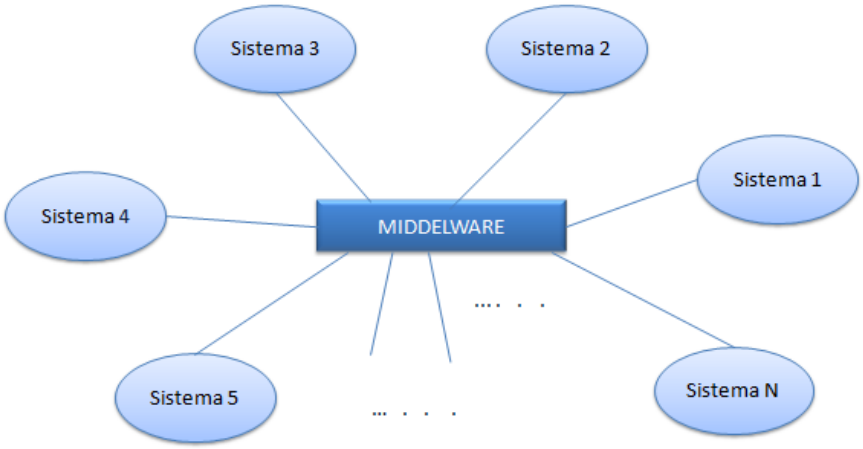


Figura 4-1: Middleware

Los objetivos que persigue middleware al establecer este tipo de comunicaciones entre varias aplicaciones son:

- Independizar las aplicaciones de la capa de comunicaciones.
- Mantener la integridad de datos entre los distintos sistemas.
- Ofrecer un acceso a datos y servicios de unas aplicaciones a otras de modo independiente de la plataforma en la que estén.

El middleware que se emplea está basado en el producto TIBCO (Integración basada en el intercambio de mensajes), hay otros productos que implementan un middleware, como WebMethods.

Una operación middelware siempre se invoca desde un sistema origen hacia uno o varios sistemas destino. Al finalizar la acción en el destino, se monta la respuesta al origen con el resultado de la operación realizada. Dependiendo del número de sistemas destino, existen dos tipos de operaciones middelware:

- **Operación Punto a Punto**: Interviene un único sistema de origen y un único sistema destino, que por supuesto, se comunican a través de middelware.

Figura 4-2: Mdw - Punto a Punto

- **Operación Multipunto**: También son conocidas como **Flujo de operaciones o Servicio** y es aquella operación en la que un único sistema origen se comunica con varios sistemas destino. Los flujos tienen suboperaciones y cada una va dirigida a un sistema destino.

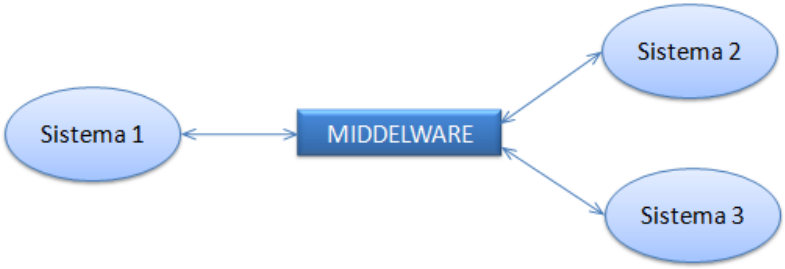


Figura 4-3: Mdw - Multipunto

Como componentes de middleware, existen una serie de *elementos de la arquitectura* que son necesarios conocer [3]:

- **Aplicación (API)**: Las aplicaciones son sistemas software que usan o ponen a disposición de middleware sus servicios o datos. Además, pueden ser tanto cliente como servidor. La API es un elemento software que permite comunicarse a la aplicación con el adaptador de middleware. Generalmente son funciones implementadas en el código de la

aplicación o en el API propio del paquete sobre el que se desarrolla. Por último, se comunican con middleware a través de adaptadores ligeros o pesados según sean y demonios.

- **Adaptador ligero**: Es la forma de comunicación entre el sistema origen y middleware. Los adaptadores ligeros actúan de traductores y adaptan el sistema con el que hacen de interfaz a middleware. Hacen que cualquier tipo de aplicación (cliente) pueda acoplarse a middleware e intercambiar información. Son utilizados por las aplicaciones origen para solicitar servicios middleware. Asimismo, en los adaptadores no se implementa ninguna lógica, su única función es transmitir correctamente los datos de las peticiones hacia su destino. Hay distintos tipos:

    - *Librerías de acceso a middleware*: Para ser utilizada los sistemas deben invocarla y pasarle los parámetros relacionados con el servicio que se quiere invocar, básicamente el código del servicio middleware y el xml con los datos que van a viajar. Las librerías generan un log donde se puede ver la petición y la respuesta del servicio middleware. Como su nombre indica la librería no es un proceso unix que se levanta y se para, es una librería puesta en una ruta. Es por ello que para que la petición se publique solo hace falta que esté arriba la máquina del aplicativo y por supuesto, con la librería cargada.

    - *Adaptador de Eventos*: Este tipo de adaptador funciona mediante un proceso que debe estar corriendo (además de la máquina, por supuesto) para que puedan publicarse peticiones de forma asíncrona. El proceso principal se conecta a la base de datos de la aplicación y va leyendo cada x segundos una tabla donde busca eventos o registros con estado 0 o pendiente de procesar. Por otro lado, el sistema origen es el encargado de ir insertando en la base de datos los registros con estado *pendiente de procesar* para solicitar un servicio middleware. Las peticiones se van encolando en la base de datos del adaptador de eventos. Se pueden producir las siguientes situaciones:

        - El estado de la petición es "pendiente de procesar" (0):

- ✓ Y el adaptador de eventos está levantado, por lo que la petición se publica, y pueden darse los casos:
  - La petición va OK y cambia al estado "procesado OK" (4).
  - Sucede algún tipo de error en la petición y cambia al estado "KO" (6).
  - La máquina destino no contesta provocando un timeout, por lo que el evento cambia al estado "timeout" (7).
- ✓ Y el adaptador de eventos está caído: el evento no se publica y mantiene el estado.

- *Http Agent, Web Service*: Son servicios que proporciona un servidor web. Este interfaz recibe desde el cliente web un código de la operación middleware solicitada y un xml de petición. Es un adaptador síncrono, así que permanece a la espera de una respuesta hasta que se cumpla un timeout que tiene fijado.

- **Demonio**: Los demonios son software de TIBCO y son los encargados de publicar y recoger las peticiones de los adaptadores ligeros y adaptadores pesados. La consecuencia directa, es que para que se publique cualquier operación correctamente, los demonios deben estar levantados.

- **Repositorios**: Son procesos unix intermedios que acceden a un repositorio de base de datos donde se almacena la lógica de cada servicio. Por ejemplo, es el repositorio el que al recibir una petición de un sistema origen, determina a qué sistema o sistemas destino debe enviar dicha petición. Dependiendo de si gestionan un servicio punto a punto (un origen y un solo destino) o un servicio multipunto (un origen y varios destinos); tendrán implementada lógica o serán meros transformadores del mensaje. Transformándolo en un xml hacia el sistema origen o en un mensaje entendible por el adaptador pesado.

- **Adaptador pesado**: Es el opuesto al ligero y por tanto, es la forma de comunicación entre middleware y el sistema destino. Se trata del último proceso que transforma la información para que el sistema destino la entienda, ejecute las tareas correspondientes y genere una respuesta. Los adaptadores pesados actúan de traductores y adaptan el sistema

con el que hacen de interfaz a middleware. Hacen que cualquier tipo de aplicación (servidor) pueda acoplarse a middlewaree intercambiar información, para ello son utilizados por las aplicaciones para recibir solicitudes de servicios middleware. Hay que tener en cuenta que en los adaptadores no se implementa ninguna lógica, su única función es transmitir correctamente los datos de las respuestas hacia su destino.

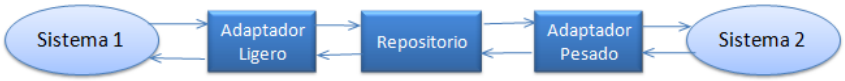


Figura 4-4: Mdw - Adaptadores

Una vez vistos los pilares básicos sobre los que se asienta middleware, podemos afirmar lo siguiente: la dificultad del tester que busca los logs que se producen al ejecutar una prueba reside en saber en qué lugar sucede el problema e identificar la máquina donde buscarlo. Damos un paso más adelante sobre el conocimiento del funcionamiento de middleware y es necesario destacar la importancia de los logs. Durante la fase de pruebas, lo que se hace es monitorizar la actividad que se produce en middleware. Aunque lo cierto es que la monitorización y almacenaje del tráfico (logs) que se genera en middleware se realiza en todos los entornos del software, vamos a centrarnos en los entornos no productivos, y más en concreto en el entorno de integración.

El producto TIBCO dispone de una utilidad para poder realizar escuchas en el bus de middleware, esta escucha se puede redireccionar hacia un fichero de log donde se almacena o bien podemos visualizar el tráfico online. Para mantener un orden, lo que se hace es generar un fichero por día, de manera que existe un fichero que contiene toda la actividad que transcurre por middleware cada día.

Una vez dentro de los logs, la primera idea que se le ocurre a cualquiera es el desorden. Pero no es así en absoluto, los logs están llenos de operaciones, ya sean punto a punto o flujos de operaciones. Una operación ok siempre se compone de un número par de segmentos, pues en ella habrá al menos dos peticiones y dos respuestas, en caso de tratarse de una punto a punto, que

sería el caso más sencillo. Sería así:

- Petición 1: La monta el sistema origen y va a middleware a través de un adaptador ligero.
- Petición 2: Middleware interpreta la petición que ha recibido, la traduce y se la envía al sistema destino, gracias al repositorio. Lo hace a través de un adaptador pesado.
- Respuesta 1: El sistema destino responde a la petición recibida y se lo envía a middleware.
- Respuesta 2: Middleware interpreta la respuesta, la traduce y se la envía al sistema origen.

En el caso de producirse un timeout y tratarse de una punto a punto, no va a llegar una respuesta al origen con el fallo puesto que estaría claro que el único sistema destino implicado no estaría contestando, es por ello, que la operación quedaría incompleta registrándose en los logs, dos peticiones pero sin respuesta.

Entre todo el tráfico existente en los logs, una de las formas de identificación de un servicio u operación lanzada, es por el tracking_id, que es un campo cuya numeración es única por operación y que se repite en los logs de petición y de respuesta que componen la operación.

Cada operación realiza una funcionalidad diferente y cada funcionalidad tiene asociada una numeración, de manera que las operaciones se publican con dicha numeración establecida previamente y se cumple que cada operación tiene siempre el mismo número asociado que además, es independiente del sistema origen que lance la operación. Así, por ejemplo, un alta de una promoción se ejecuta mediante una operación 09030 y un alta de promoción es siempre una 09030 independientemente desde qué sistema haya sido ejecutada.

Para tener un control absoluto sobre las operaciones, existe una excel a la que denominamos *Canon de operaciones* en la que se almacenan absolutamente todas las operaciones existentes con los datos que las componen, tales como: sistema origen, descripción, numeración que la identifica, adaptadores que necesita levantados para poder publicarse, suboperaciones asociadas para los casos de los flujos de operaciones,...

Sistema Origen	Adaptador Ligero	Repositorio	Servicio	Operación	Adaptador Pesado	Descripción	Repositorio
Sistema 1	Ad. Eventos	Repositorio 1	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 2	HTTP Agent	Repositorio 2	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 3	Ad. Eventos	Repositorio 3	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 4	LibAccMdw Java	Repositorio 4	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 5	Ad. Eventos	Repositorio 5	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 6	LibAccMdw Java	Repositorio 6	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 7	Ad. Eventos	Repositorio 7	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
Sistema 7	LibAccMdw Java	Repositorio 7	09030	09030	Adaptador 1	Alta, Baja y Modificación de Promoción	Repositorio 8
			09030	26069	Adaptador 2	Alta, Baja y Modificación de Promoción	Repositorio 9
			09030	26069	Adaptador 2	Alta, Baja y Modificación de Promoción (Cobro OCC)	Repositorio 9
			09030	03018	Adaptador 2	Consulta Tipo Cliente	Repositorio 10
			09030	03030	Adaptador 2	Consulta de Oferta Comercial	Repositorio 10
			09030	13002	Adaptador 4	Envío de SMS (informa del alta o baja de la promoción)	Repositorio 11
			09030	13004	Adaptador 4	Envío de SMS Campañas	Repositorio 11
			09030	74001	Adaptador 5	Envío de MMS	Repositorio 12

Figura 4-5: Canon de Operaciones

En la figura 4-5 podemos observar un filtro en la excel del *Canon de operaciones* por servicio. En primer lugar se puede ver que se trata de un flujo de operaciones porque tiene operaciones asociadas al servicio 09030, que tienen otra numeración al ser suboperaciones y que se publicarán dentro del servicio principal. Todas las suboperaciones tienen en común que no disponen de sistema origen especificado, eso es porque todas estas suboperaciones se publican igualmente dentro del servicio, independientemente de quien haya sido el sistema invocador. Al filtrar por servicio, además, se pueden consultar todos los sistemas origen desde los que es posible ejecutar este tipo de servicio.

Para finalizar este importante apartado, apuntaremos que para que haya una prueba de integración, middleware tiene que estar implicado, pues es un indicador inequívoco de que se está produciendo una comunicación entre más de un sistema, es decir, que se está produciendo integración entre diferentes sistemas. De hecho, en aquellos casos en los que un sistema no realice comunicación con otro sistema, porque por ejemplo, directamente actualice sus bases de datos sin necesidad de indicárselo a otro sistema, no se hablará de integración como tal, sino que oficialmente se trataría de una prueba de desarrollo o componentes.

4.1.2 Entornos de pruebas

Las pruebas de calidad de software están compuestas por un escrupuloso orden que permite tener un control sobre lo que esté sucediendo en cada momento en cada entorno de pruebas. De manera que en este caso el cliente, pues es a quien pertenecen los entornos en los que

trabajamos, dispone de unos entornos independientes entre sí. El orden de utilización de los entornos que se sigue es:

- **System Test**: Los equipos de desarrollo elaboran un software que posteriormente instalan en el entorno de system test para ejecutar sus pruebas de sistema. En cada entorno de system test solo existe un sistema, por lo que solo puede probarse la funcionalidad interna y correspondiente a un único sistema. Al realizar las pruebas, localizarán errores que lógicamente tendrán que corregir modificado el software inicial en el entorno de system test.

- **Entorno Integrado**: Una vez realizadas las pruebas de cada sistema, se van realizando las instalaciones en el entorno de integración de los diferentes sistemas, por supuesto, con sus correcciones incluidas. Aquí entra en juego la interacción entre los sistemas y los diferentes equipos a través de la herramienta PVCS que veremos en detalle más adelante. Gracias a PVCS se tiene un control absoluto sobre las instalaciones que se van realizando en el entorno de integración. Es un entorno en el que conviven por primera vez todos los sistemas, con la finalidad de que se realicen comprobaciones sobre las conexiones entre ellos, es decir, las pruebas integradas.

- **Entorno UAT**: Tras la ejecución de las pruebas integradas y la corrección de todos los errores localizados, el siguiente paso es instalar el software ya depurado en el entorno de aceptación, convirtiendo así al entorno de UAT en un espejo del entorno integrado. Este entorno está destinado para que sobre él se realicen las pruebas de usuario.

- **Entorno de Producción**: Es el entorno real, el entorno que se utiliza para ofrecer los servicios a los clientes finales. Una vez concluidas todas las pruebas sobre los entornos no productivos, el último paso es instalar el software junto con todas las correcciones a los errores localizados durante las diferentes fases de pruebas en el entorno de producción. Al tratarse del entorno real, las instalaciones ya no se tratan con PVCS. Tras realizarse una subida a producción, se realizan pruebas de postproducción para asegurar que el software que se acaba de instalar funciona sin problemas en el entorno final, ya que en este entorno, cualquier tipo de fallo puede suponer un error con cliente final y por tanto surge la posibilidad de pérdida económica. Además, para asegurar el correcto funcionamiento del entorno de producción, es a este entorno al que se le somete a

pruebas de pruebas de carga, de rendimiento, de estrés, ... para verificar que soporta los usos extremos que pueden llegar a realizar los clientes. Por descontado, este tipo de pruebas extremas se realizan a horas en las que se conoce que el entorno de producción no está siendo prácticamente utilizado, para evitar en lo máximo posible cualquier molestia que pueda ocasionarse a los clientes finales.

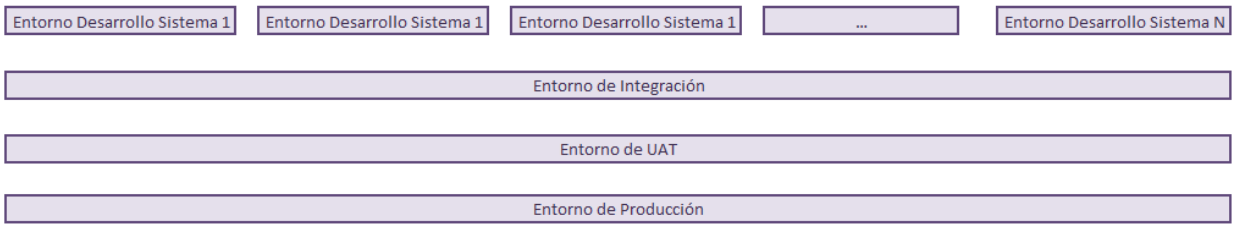


Figura 4-6: Entornos de Pruebas

En la figura 4-6, se representa una simulación de lo que podría entenderse cómo son los diferentes entornos. De manera que cada sistema inicialmente se encuentra instalado solo en su propio entorno, y después va a estar instalado en el resto de entornos. De hecho, se puede afirmar, que a pesar de que hay ciertas diferencias, como algunas simulaciones y diferencias de rendimiento; los entornos de integración, UAT y producción, son prácticamente iguales en cuanto a instalaciones de software.

4.1.3 Instalaciones

Como venimos viendo a lo largo del documento, una de las partes primordiales del ciclo de vida del software de las pruebas integradas es la generación de software y consecuentemente, sus instalaciones. Existe un equipo especializado que se encarga de las instalaciones del software, al que denominamos SCM y con el que interactuamos continuamente para solicitarles instalaciones, las cuales realizan a través de la potente aplicación PVCS.

Con el fin de evitar problemas de índole técnica durante de la ejecución tanto de las pruebas integradas como de las pruebas de usuario mientras se realizan instalaciones, las cuales en la mayoría de los casos requieren de un reinicio de la máquina del sistema donde se esté instalando. Se han creado unos horarios determinados de instalación durante los cuales se conoce que seguramente el entorno no esté perfecto en su totalidad y por tanto, no deben ejecutarse pruebas contra los sistemas donde se esté instalando. A estos horarios de instalación los denominamos **ventanas de instalación** y suelen determinarse a primera hora, a medio día y al final de la jornada.

En ocasiones puntuales, sí se permite la instalación de software que no requiera el reinicio de los sistemas y que por tanto, no vaya a ralentizar el avance de las pruebas durante la jornada laboral. Dicha excepción no tendrá permiso para realizarse si el pico de trabajo en pruebas es muy alto en un momento dado, ya que si sucede algún tipo de error en la instalación, aunque inicialmente no iba a suponer un reinicio de la máquina, sí lo suponga finalmente. Es por ello, que la mejor opción en momentos de picos altos de trabajo, es esperar a que cualquier tipo de instalación de software se realice únicamente durante las ventanas de instalación.

Otro punto a tener en cuenta sobre las instalaciones son los **pasos a producción**. Los pasos a producción van ligados a la finalización de una release, siendo una release un conjunto de proyectos cuya fecha de subida a producción es común. En una situación óptima en la que todos los proyectos hayan sido finalizados y todos puedan subirse a producción, todos formarán parte del **kit de producción** que se realiza para el seguimiento de las entregas que deberán instalarse en el entorno de producción.

Las instalaciones en producción procuran realizarse durante las noches de los fines de semana, con el fin de interferir lo menos posible en las necesidades de los clientes reales.

En las ocasiones en las que no se han finalizado las pruebas de todos los proyectos que componen una release, pueden suceder los siguientes supuestos:

- En el kit de producción solo se incluyen las entregas de los proyectos finalizados.
- En el kit de producción se incluyen las entregas de los proyectos finalizados y algunas entregas en particular de otros proyectos, aunque estén por finalizar. Se pueden dar las siguientes situaciones:

- Existencia de **dependencias** entre las entregas: Se tiende a que esto suceda cada vez en menos ocasiones por los problemas que supone llegar a esta situación, pero la realidad es que siguen realizándose entregas dependientes entre sí, lo que puede obligar a instalar en producción entregas cuyas pruebas no hayan finalizado porque una entrega no pueda instalarse si no se instala otra. En este caso, si finalmente se opta por instalar en producción, se corre un riesgo al instalar un software que no se haya finalizado de probar. Aquí corresponde al equipo de pruebas determinar los riesgos que conllevaría dicha instalación, que depende directamente de las pruebas que se hayan ejecutado. Por ejemplo, depende de la prioridad de las pruebas ejecutadas para las que ya se tenga ok. En el caso de que el riesgo sea muy grande porque por ejemplo, no se hayan iniciado las pruebas sobre el software dependiente, sería el cliente quien decide si corre el riesgo de subir un software sin probar al entorno de producción, o si bien, no sube ninguna entrega dependiente hasta que se pruebe todo el software.

  También sucede el caso en el que desde casi el principio de las pruebas se conozca que hay cierto software dependiente, lo que supone la puntualización de un riesgo y al tratarlo como tal, se pondrá más foco por parte de todos los equipos (tanto pruebas, como desarrollo para solucionar los defectos que se localicen cuanto antes) en intentar sacar adelante las pruebas sobre este software dependiente antes que otras pruebas. Es decir, se otorgaría prioridad de ejecución al software dependiente para intentar evitar problemas a la hora de la subida a producción.

- Subida de un **proyecto con incidencias sin resolver**: Puede darse el caso en el que se haya detectado una incidencia en un proyecto durante las pruebas de integración o durante las pruebas de usuario, la cual esté sin resolver y por tanto, sin software de corrección subido al entorno correspondiente de pruebas. Y que además, por cuestiones de tiempo, de software y/o económicas sea imprescindible subir dicho proyecto a producción con la incidencia sin resolver. En estos casos, la incidencia pasará a ser inmediatamente una incidencia de producción ocasionando las siguientes consecuencias:

- Una vez abierta la incidencia de producción, se podrá anular la incidencia de pruebas correspondiente.

- La incidencia se resolverá a través de la entrega de la incidencia en el entorno de producción, pero quedará sin solución en el entorno no productivo donde se descubrió. De cara a futuras pruebas, esto no puede permitirse y es por ello, que una vez entregada, instalada y probada la PR en el entorno de producción, se procede a la entrega, instalación y prueba de la incidencia en los entornos no productivos: integración y aceptación.

- Las incidencias en entornos no productivos que se anulen por una incidencia en producción no afectarán negativamente a los ANS, ya que aunque se anulen, realmente se hará una entrega que solucione el problema localizado inicialmente. Es por ello, que en el motivo de anulación de la incidencia de entornos no productivos se especifica que se anula por la incidencia de producción correspondiente, realizando así una trazabilidad de la incidencia en cuestión.

- Proyectos sin dependencias de software: Aunque las pruebas de un proyecto no estén al 100%, puede ser importante que suba a producción, y si se confirma que las pruebas realizadas son suficientes para asegurar el funcionamiento del software, dicho proyecto subirá también a producción. En estos casos lo que entra en juego es la prioridad asignada a los casos de prueba y se toma la decisión en base a los casos ejecutados de prioridad alta, de manera que se asegura que la funcionalidad más relevante está depurada. En alguna ocasión excepcional, lo que se acuerda es finalizar las pruebas integradas como pruebas de postproducción, asegurando así que lo que no se ha probado en el entorno de integración, se verifique en el entorno de producción.

- **Ocultaciones** en producción: Esto consiste en subir el proyecto al entorno de producción pero se ocultando su funcionalidad de manera que no sea visible ni para el cliente final ni para los operarios de la empresa, es decir, quedaría instalado en producción, pero no sería posible su ejecución en dicho entorno. Normalmente, las ocultaciones están muy ligadas a las dependencias entre las

entregas, pues lo más lógico sería no subir proyectos para ocultarlos. Pero seguramente, las circunstancias del momento no den lugar a otras opciones, por lo que se escoge subir el proyecto, ocultarlo, seguir realizando las pruebas como si nada hubiese sucedido en los entornos no productivos. Y una vez finalizadas dichas pruebas, se procedería a la desocultación del proyecto en producción, o lo que es lo mismo, hacer visible la funcionalidad ya subida a clientes y operadores.

4.2 Herramientas de Prueba

Una vez que el proceso de pruebas está definido, las herramientas ayudan que se lleve a cabo de manera efectiva.

Las tareas de pruebas pueden tanto apoyarse en herramientas (Tools) como automatizarse mediante ellas. El apoyo en herramientas tiene sentido en aquellas áreas en las que se trabaja de forma "creativa". La pura ejecución de las pruebas (el doing) en ocasiones puede automatizarse mediante herramientas.

La totalidad de las herramientas se denominan en relación al concepto CASE (Computer Aided Software Engineering) que son utilizadas en torno a la ingeniería del software o CAST-Tools (Computer Aided Software Testing) empleadas de manera concreta en pruebas.

Las herramientas se diferencian según su clasificación, ya que para cada etapa del proceso de pruebas se dispone de diferentes herramientas. Y además algunas herramientas dan soporte a un tipo de actividad. De hecho, existen herramientas llamadas **suites** que proporcionan soporte a la mayoría o a todas las actividades asociadas al proceso de pruebas, como pueden ser HP-Mercury, IBM-Rational, Borland, ...

Las herramientas son particularmente útiles cuando hay tareas repetitivas proporcionando mejoras en la eficiencia y en la fiabilidad.

Por último, una característica común de las herramientas y que las dota de un mayor valor, es que en ellas se puede realizar un **registro de la actividad** que se realiza en ellas mismas.

Componiendo así un histórico de actividades donde poder consultar en un futuro todo tipo de detalles sobre las mismas, como son: usuario que realiza algún tipo de modificación, fecha y hora de la actividad, recurso implicado en la actividad, estado al que cambia un registro,...

A continuación, pasamos a detallar las herramientas más importantes que utilizamos a diario en el equipo de pruebas integradas.

4.2.1 Quality Center

Se trata de una herramienta para la gestión de los casos de prueba y el proceso de prueba que sirve de apoyo al tester en la administración y seguimiento de los diversos planes de pruebas [4]. Gracias a Quality Center se puede realizar un seguimiento de la ejecución de las pruebas y ver en tiempo real el estado de ejecución de los casos de prueba. Para ello, tiene una interfaz a través de la cual se muestra: el diseño de cada caso de prueba organizado por proyectos, la ejecución de los casos, una gestión de las incidencias, gestión de los requisitos, y gestión de la configuración.

Además, permite tener un soporte a la trazabilidad de requisitos - casos - resultados - incidencias. También se realiza un registro y una administración de los casos, detalle de los mismos, datos y resultados de las ejecuciones. Esto facilita la última cualidad a destacar, que es la generación automática de documentación de las pruebas como son: el plan de pruebas o los informes de pruebas. En general, al tener un registro permite la obtención de documentación de las pruebas ya sea directa, resumida o procesada.

Más en concreto, podemos decir que Quality Center es la suite de HP para la gestión de pruebas. Consta de cinco módulos [3]:

- **Release**: Los proyectos se agrupan por la fecha de subida a producción o PAP que tengan, siendo un conjunto de proyectos con común PAP, una Release. Por lo tanto, en Release lo que se introducen son los proyectos que la componen.
- **Requeriments**: Es un listado donde se reflejan todas las entregas de software, a las que llamamos RU y es el último lugar donde deben figurar los casos creados en Test Plan e incluídos en Test Lab, porque en Requeriments es donde se asocian al requisito o entrega dela que dependen.

Se estructura en tres niveles:

- El primero agrupa las entregas (RU) en grupos de 100.
- El segundo sería la entrega (RU) asociada a un proyecto en concreto.
- El tercer nivel es el que se añade con las funciones de prueba definidas para esa RU y sistema.

- **Test Plan**: Es donde se inserta el plan de pruebas del proyecto. Aquí se diseñan los casos de prueba y se definen los campos de cada prueba según la configuración definida. Se incluye información como: descripción del caso de prueba, entorno en el que tiene que probarse, proyecto al que pertenece, usuario del diseñador que lo ha creado, prioridad de ejecución, pasos a seguir para su realización, precondiciones y resultado esperado.
  Se estructura en cuatro niveles:

  - *Release*: Dentro de este nivel se incluirán todos los proyectos que se han entregado para el sistema y para esta Release o conjunto de proyectos con la misma fecha de subida a producción.
  - *Proyecto*: Dentro de este nivel se incluirán todas las entregas (RU) que se han abierto para el sistema y el proyecto.
  - *RU o entrega*: Dentro de este nivel se incluirán todas las funciones que se tienen que probar debido a la entrega de una RU, deben incluirse funciones de Regresión.
  - *Funcionalidad*: En este nivel se incluirán directamente los casos de prueba para la funcionalidad definida en este directorio.
- **Test Lab**: Una vez se han diseñado los casos en Test Plan, se añaden a Test Lab que es donde se crea el test set para ejecutar los casos de prueba. Es decir, donde se controla el estado de ejecución de los casos de prueba y se organizan las ejecuciones. Además, incluye información específica del caso como: hora y fecha última ejecución, ejecutor, si el caso es de ejecución manual o automático, también se puede escoger mostrar los campos que se quieran y que figuran en Test Plan. Test Lab permite visualizar los casos de prueba con el estado en que se encuentran, que son:
  - *No ejecutado*: El caso de prueba está creado en Test Plan y cargado en Test Lab pero aún no se ha lanzado la ejecución de esta prueba.
  - *Bloqueado*: Indica que el caso de prueba no se puede ejecutar en estos momentos

por el motivo que sea, que además es imprescindible que se indique el mismo explícitamente en el campo dedicado a este fin. El motivo de bloqueo puede ser una entrega que no se haya entregado e instalado, un fallo en el entorno de ejecución, un problema con el recurso o dato que se necesita para su ejecución, un fallo localizado en otra prueba que pueda afectar también a ésta, etc... El caso de prueba puede haberse ejecutado o no, aunque lo normal es que no se haya ejecutado para este estado.

- *Pasado*: El caso de prueba ha sido ejecutado y el resultado ha sido satisfactorio. Hay que añadir al caso de prueba la evidencia de la prueba ejecutada que demuestra que el resultado esperado coincide con el obtenido. Documentar los resultados es de suma importancia, ya que si en un futuro surgen problemas que ronden a alguna de las pruebas ejecutadas, gracias a la evidencia se podrá demostrar que en el transcurso de las pruebas funcionaba correctamente. Es por ello, que la evidencia debe plasmar claramente que el resultado obtenido coincide con el resultado esperado. Debe incluir los logs de la ejecución y/o pantallazos sobre las diferentes pantallas que apareciesen durante la ejecución.
- *Fallado*: El caso de prueba se ha ejecutado y se ha producido un defecto, lo que indica que el resultado esperado no se ha conseguido. Hay que abrir una incidencia y reportarla con todo nivel de detalle al equipo de desarrollo para que revise lo sucedido y el problema se solucione lo antes posible. Al cambio del estado del caso a fallado, se le añade un motivo descriptivo que indique de qué fallo se trata.
- *Pendiente*: Consiste en un caso de prueba que puede ejecutarse, esto es, que no hay nada que no permita su ejecución y por tanto, está a la espera de que se lance la correspondiente prueba.
- *Cancelado*: Se trata de un caso de prueba que a pesar de haber sido diseñado, finalmente no aplica. El que no aplique puede deberse a diversos motivos, algunos de ellos podrían ser: un cambio de alcance, un fallo en el diseño, un caso redundante, ...

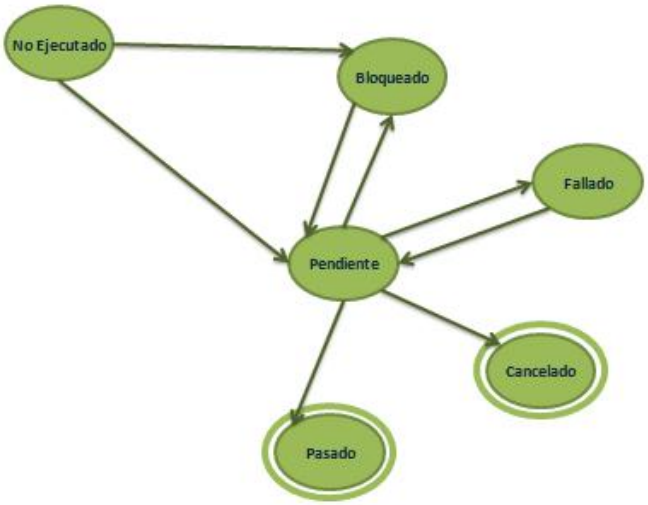


Figura 4-7: Estados Casos de Prueba

- **Defects**: A través de esta opción se reportan las incidencias detectadas en el proyecto en el que se está trabajando. Llevado a nuestro caso particular, somos integración y los usuarios quienes trabajamos con QC. Desde integración no se abren defectos en QC pues desarrollo no tiene visibilidad sobre dicha herramienta. Los que sí abren defectos para registrar los problemas que se encuentran durante la ejecución de las pruebas de un proyecto son los usuarios. Como desde el equipo de integración se da soporte a las pruebas del usuario, desde integración se revisan los defectos y se tratan a través de QC. Y somos el equipo de integración el enlace entre el equipo de aceptación y desarrollo, o lo que es lo mismo, los usuarios no tienen comunicación directa con desarrollo, cualquier problema entre aceptación y desarrollo se trata a través del equipo de integración. El principal motivo es la falta de conocimiento técnico por parte del usuario.

  Por otro lado, la existencia de los defectos es de gran utilidad pues entre otras cosas, obliga a que el usuario aporte un detalle sobre el problema localizado, facilitando así al equipo de integración el estudio del defecto localizado.Si bien se trata realmente de un error el equipo de integración es el encargado de reportarlo a desarrollo, o si bien no es más que algo puntual o fallo en un recurso, se le indica al usuario que repita la prueba. La comunicación entre integración-usuario se produce mediante los diferentes estados de las incidencias que abren los usuarios, que son:

  - *Abierto*: El usuario localiza un defecto y lo abre asociándolo al caso de prueba

que ha ejecutado. En el defecto debe especificar los datos de prueba que ha utilizado, el entorno de ejecución, qué esperaba obtener y qué resultado ha obtenido para que desde el equipo de integración se puedan buscar los logs referentes a la ejecución lanzada por el usuario.

- *En Análisis*: A este estado lo transita el equipo de integración en cuanto está con el estudio del defecto y lo mantendrá así hasta que determine si se trata realmente de un error o no.
- *Más-Datos*: Si el equipo de integración necesita más datos sobre la ejecución de la prueba que resultó en el fallo obtenido por el usuario, transitándolo a Más-Datos el usuario tendrá que proporcionar un mayor detalle sobre la prueba y su ejecución.
- *En-Resolución*: Un defecto estará en este estado cuando tenga una incidencia asociada a él. Si tras el análisis realizado por el equipo de integración se determina que hay una incidencia, que en este caso se tratará de una SM al estar en pruebas de aceptación, dicha incidencia tendrá que asociarse al defecto que quedará en estado En-Resolución hasta que la incidencia se resuelva, se entregue y se instale.
- *Asignado*: A este estado se transita el defecto cuando el usuario puede repetir la ejecución de la prueba, ya sea porque se haya solucionado un problema puntual que puede haber ocasionado el fallo o bien porque se haya entregado la incidencia asociada al defecto.
- *Rechazado*: Es el estado al que el usuario transita un defecto tras haber reejecutado una prueba y seguir obteniendo el error que obtuvo inicialmente cuando abrió el defecto. Es importante que se trate del mismo error, si fuese diferente, tendría que tratarse en otro defecto.
- *Resuelto*: Una vez reejecutada una prueba por parte del usuario, si obtiene el resultado esperado que se indica en el caso, transitará el defecto a Resuelto para darlo por solucionado.
- *Petición-Anulado*: Cuando se considera que un defecto no es tal, ya sea porque se tratase de algo puntual y no se pueda reproducir el fallo o porque se pueda considerar que esté fuera del alcance establecido inicialmente, el equipo de integración transita el defecto a Petición-Anulado a la espera de que el usuario

esté de acuerdo en su anulación.

- *Anulado*: Se trata de un estado final al que se transita un defecto cuando tanto el usuario como integración están de acuerdo en su anulación.

  Cabe indicar, que los defectos que se anulen, van en contra de las estadísticas o ANS del equipo de los usuarios, ya que estarían abriendo defectos que no aplican con su consiguiente pérdida de tiempo y sobretodo de un mal análisis inicial.
- *Cerrado*: Es el estado final al que transita los defectos el equipo de integración una vez que el usuario los resuelve.

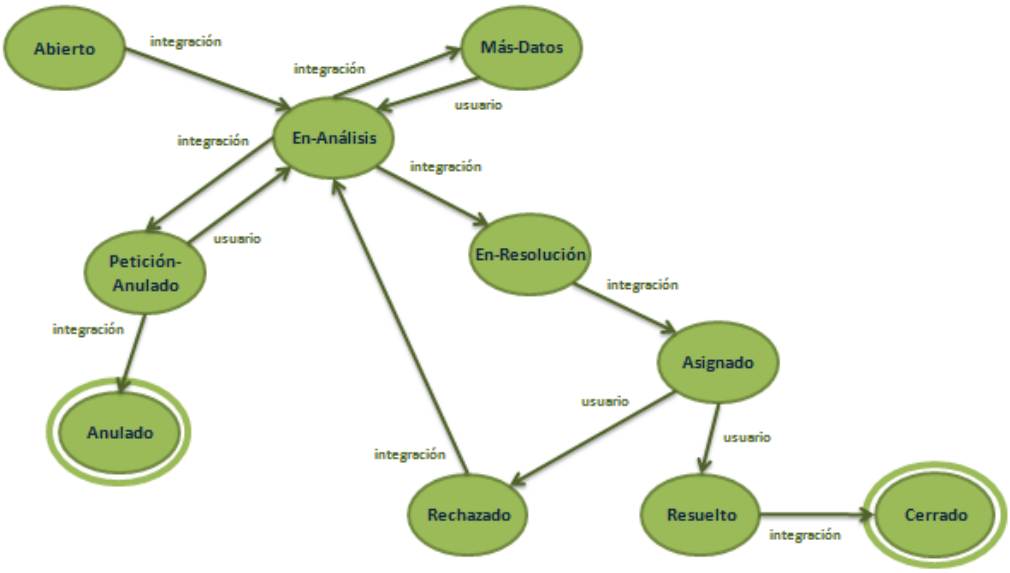


Figura 4-8: Estados de los Defectos

4.2.2 PVCS

Es la herramienta a través de la cual se controla todo lo concerniente a las entregas e instalaciones de software. Mediante esta herramienta se realizan las entregas de software, se solicita permiso para las instalaciones, se aprueban las instalaciones, se puede consultar el estado de una entrega, como por ejemplo si ya está instalada, o si ha subido a producción [5]. Todo este tipo de acciones pueden realizarse gracias a que cada equipo que tiene acceso a ella, que son desarrollo, el equipo encargado de las instalaciones al que denominamos SCM y pruebas, tiene unos permisos para transitar unos estados u otros. Así, solo puede realizar una entrega desarrollo, solo SCM puede solicitar permiso para instalar, solo el equipo de pruebas puede dar permiso para la instalación, solo SCM puede transitar el estado de la entrega a En Test para indicar que se ha realizado la instalación correctamente. Por supuesto, todo queda registrado con hora y fecha, usuario y modificación que ha realizado.

Hay una conexión entre QC y PVCS, y es que si una entrega de PVCS bloquea casos de prueba de QC, se especifica en el motivo de bloqueo de los casos afectados, añadiendo así una trazabilidad entre las entregas pendientes y los casos de prueba.

A cada entrega de software que se realiza se la denomina request. Y existen dos tipos de requests, los que son entregas en sí de software, que serían las entregas de un proyecto y los que son incidencias asociadas a ese software inicial.

- **RU**: Software de un nuevo requisito solicitado por el cliente.
- **ST**: Incidencia localizada en el entorno de desarrollo durante las pruebas de System Test que realiza el equipo de desarrollo.
- **TI**: Incidencia detectada en el entorno de integración mientras se realizan las pruebas integradas.
- **SM**: Incidencia detectada en el entorno de UAT durante las pruebas de usuario.
- **PR**: Error localizado en el entorno de producción, bien durante las pruebas de postproducción o bien durante cualquier transacción realizada para clientes reales.

Dado que el proceso es: pruebas de desarrollo, pruebas de integración, pruebas de usuario. Cada incidencia localizada, puede significar que dicho problema ha pasado por alto en la fase anterior. Debido a ello, existen unos ANS que penalizan al equipo anterior por cada incidencia localizada

por el siguiente equipo en el proceso. Así una TI penaliza directamente sobre desarrollo y una SM sobre el equipo de integración.

Dependiendo del momento en el que se encuentre la entrega de un request y por tanto de un software dado, estará "en manos" de un equipo u otro. Así una entrega se puede estar desarrollando, puede estar instalándose o puede estar en pruebas. Para especificar esta situación, hay una serie de estados por los que todo request va transitando desde que se crea hasta que se cierra, son los siguientes:

- **Análisis Desarrollo**: Es el estado inicial de un request. La entrega de software se encuentra en pleno desarrollo, ya ha sido nombrada para que todos los participantes en las pruebas sepan de su existencia.

- **Resuelta Proveedores**: El desarrollo del software asociado a la entrega ha finalizado y el equipo de desarrollo lo indica transitando el request a Resuelta Proveedores mediante lo cual además debe realizar la entrega del software creado.

- **Validado SCM**: El equipo encargado de realizar la instalación, en primer lugar tiene que comprobar si está todo correcto y listo para instalarse. Con este estado, SCM aprueba la entrega.

- **Rechazado SCM**: El equipo de SCM, al realizar la comprobación sobre la entrega realizada, comprueba que faltan datos o hay problemas previos que le impiden iniciar la instalación correctamente, por lo que rechaza la entrega para que desarrollo la corrija y la entregue de nuevo.

- **Petición a Integración**: Una vez que SCM ha aprobado la entrega, solicita permiso al equipo de pruebas de integración para instalar en el entorno de integración.

- **Aprobada Integración**: El equipo de pruebas de integración da permiso para que comience la instalación por parte del equipo de SCM.

- **En Test**: SCM ha realizado la instalación de la entrega correctamente, lo indica transitando el request al estado En Test y por tanto pueden comenzar las pruebas de integración.

- **Resuelta Integración**: Las pruebas de integración se han finalizado. Se trata de un estado final, lo que indica que esta entrega podría subirse a producción en el momento en que se decida.

- **Cerrada**: La entrega ya se encuentra instalada en el entorno de producción.

Lo habitual es que los proyectos se prueben primero en el entorno de integración y después en el entorno de UAT, para ello, existen cuatro estados más para controlar el estado del request:

- **Petición a UAT**: El equipo de SCM solicita permiso para instalar el request en el entorno de UAT. Tal y como está dimensionado el equipo de los usuarios solo tienen acceso a PVCS para realizar consultas y es el equipo de integración quien controla las instalaciones tanto en el entorno de integración como en el de aceptación, debido sobre todo al conocimiento técnico que tienen los tester de integración y del que carecen los usuarios.

- **Aprobada UAT**: Se aprueba la instalación.

- **En UAT**: La entrega está instalada y lista para que se ejecuten las pruebas.

- **Resuelta UAT**: Una vez terminadas las pruebas de aceptación, se resuelve la entrega. Al igual que Resuelta Integración se trata de un estado final con lo que la entrega puede cerrarse y subirse a producción.

Por último, añadimos dos estados más en los que se trata la anulación de las incidencias que no apliquen y que aplican a las TIs y SMs:

- **Petición Anulación**: Tanto para el caso de las TIs como de las SMs, el equipo de desarrollo y el equipo de integración pueden solicitar tal anulación. Por descontado, las anulaciones se solicitan porque se considera que el código no necesita ningún tipo de corrección, por tanto, la anulación es previa a una posible creación de código correctivo.

- **Anulada**: Al igual que el estado anterior, son el equipo de desarrollo y el equipo de integradas quienes tienen los permisos para poder anular una incidencia.

Tal y como sucedía con el caso de los defectos, la anulación de las incidencias incide negativamente sobre los ANS del equipo que abrió la incidencia. Si es SM afecta al equipo de aceptación y si es TI afecta al equipo de integradas.

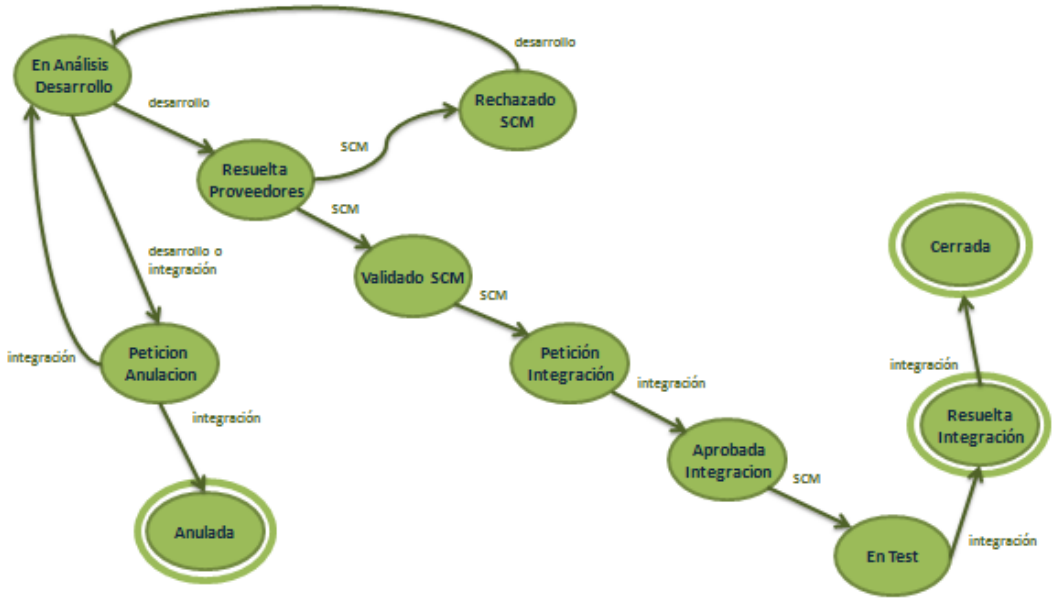


Figura 4-9: Diagrama de Estados en PVCS

Cabe destacar que la existencia de los requests y de sus diferentes estados, aporta al proyecto en general, un perfecto y absoluto control sobre los entornos y sus instalaciones, por no decir que consultando PVCS se puede saber en todo momento el estado actual de las entregas. Tener este privilegio en un proyecto otorga una mayor fiabilidad sobre el software pues permite tener una visión muy real sobre el estado del mismo y lo que es más importante, sobre cómo se va a comportar tras la subida a producción. Es del tipo de herramienta del que no te das cuenta de su importancia hasta que careces de ella y cualquier desarrollo puede instalar lo que quiera, cuando quiera. Haciendo imposible que las pruebas realizadas tengan valor real sobre un software que cambia sin control.

4.2.3 Jtrac Estimaciones

Es una herramienta libre mediante la cual se realiza el control de los proyectos y sus costes. Así en una jtrac [6] de un proyecto lo primero que se incluye es toda la documentación por parte del cliente referente al proyecto. A continuación, dicha documentación se utiliza para trabajar en la estimación de pruebas a realizar junto con el coste que supondría realizarlas. En cuanto se dispone de ella, se sube la estimación obtenida a jtrac, donde el cliente la revisa y o bien la aprueba indicando el mes en el que está previsto que se instale para que se incluya como un proyecto más en la planificación o bien propone algunos cambios, de manera que habría que reestimar.

Al igual que sucede con las herramientas que hemos visto anteriormente, con la jtrac de las estimaciones también se utilizan diferentes estados que permiten saber a quién le pertenece dar el siguiente paso a lo largo de todo el trabajo. Son unos estados acordados previamente y sobre los que solo tiene el poder de cambio de estado un equipo en cada momento, lo que delimita perfectamente a quien le corresponde realizar la siguiente acción en el ciclo de vida de la estimación. En este caso, la interacción se produce entre el cliente y el equipo de integración, dado que es el cliente quien solicita realizar pruebas y el equipo de integración quienindica qué pruebas va a realizar y qué coste conllevan. Por otro lado, aunque las pruebas de aceptación corren a cargo del propio cliente, que es a quien pertenecen los usuarios, el equipo de integración da soporte técnico a los usuarios para que puedan ejecutar sus pruebas, es por ello, que todo proyecto con pruebas de aceptación supone un coste que ha de abonarse al equipo de integración por los servicios prestados. De forma que en la estimación se incluye el alcance funcional de las pruebas a realizar por parte del equipo de integración si tiene pruebas integradas y/o el alcance funcional de las pruebas que sabemos que ejecutarán los usuarios, para cubrir con ellas el soporte técnico que les ofreceremos durante sus pruebas de aceptación, en caso de tenerlas.

Los estados de las estimaciones de jtrac son los siguientes:

- **Abierta**: Cuando el cliente dispone de un proyecto sobre el que quiere que se realicen pruebas, abre una jtrac e introduce toda la documentación del proyecto e información sobre si requiere pruebas integradas y/o de aceptación y fecha estimada para la realización de las pruebas.

- **En-Estudio**: El equipo de integración la transita a este estado para indicar que se encuentra en el análisis de la documentación recibida.

- **Más-Datos**: El equipo de integración transita la jtrac a Más-Datos cuando considera que falta algún tipo de documentación para poder realizar la estimación, además, debe indicar explícitamente qué necesita.

- **Estimada**: Una vez que el equipo ha finalizado la estimación, sube dicha estimación a jtrac y transita el estado a Estimada, indicando que ha finalizado una etapa más.

- **Petición-Cancelada**: Si el equipo de integración, tras haber estudiado la documentación concluye que el proyecto no requiere pruebas de integración o de usuario, transita la jtrac a Petición-Cancelada.

- **Cancelada**: Si entre el cliente y el equipo de integración se llega al acuerdo de no realizar pruebas ni de integración ni de aceptación sobre un determinado proyecto, finalmente se cancela la jtrac.

- **Rechazada**: El jefe de proyecto debe revisar la estimación realizada por el equipo de integración y puede rechazarla en caso de no estar de acuerdo con la estimación aportada. Puede ser un desacuerdo en cuanto a la funcionalidad o al coste. Aunque es evidente que a menor coste, menor funcionalidad a probar habrá.

- **Aprobada**: Si al jefe de proyecto le parece bien la estimación, la aprueba.

- **En-Int**: Este estado indica que la estimación aprobada ya está en pruebas de integración. Cabe destacar que un proyecto puede tener pruebas integradas y/o pruebas de aceptación.

- **En-UAT**: Este estado indica que la estimación aprobada ya está en pruebas de aceptación.

- **Cerrada**: Una vez finalizadas las pruebas, se cierra la jtrac.

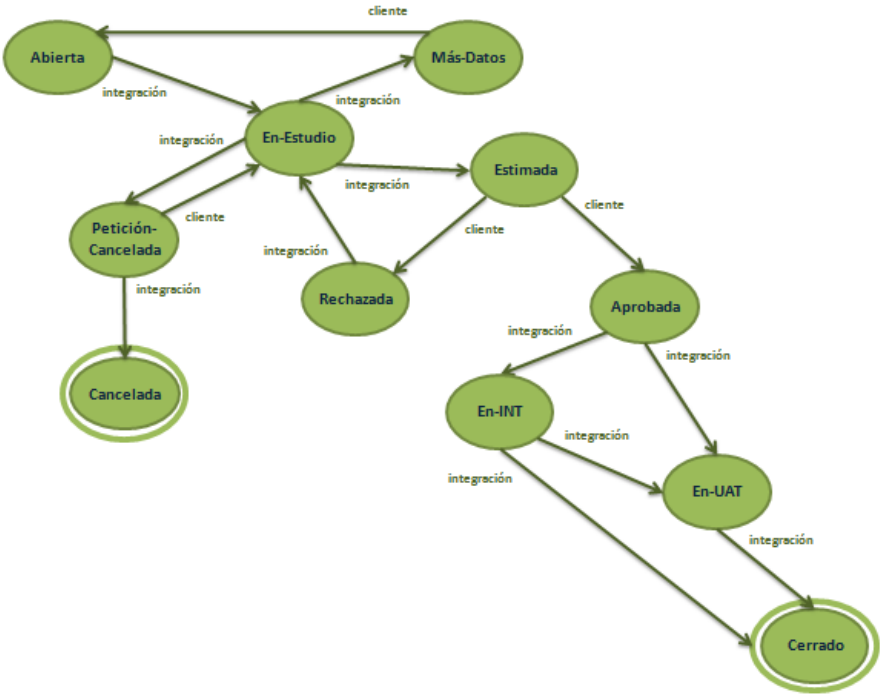


Figura 4-10: Estados Jtrac Estimaciones

4.2.4 Jtrac Gestión Entornos

De nuevo utilizamos una herramienta libre, en este caso para tener un control sobre las incidencias que se producen en los entornos y poder tratarlas. En cuanto se detecta un problema en un entorno, ya sea por parte del equipo de pruebas, el propio desarrollo o los usuarios; lo que se hace es abrir una incidencia jtrac [6] para comunicarles el detalle del problema localizado al equipo de Gestión de Entornos, que es quien se encargará de solucionarlo. La incidencia puede abrirse porque sea un problema que requiera un reinicio, una configuración de alguna variable de entorno, el levantamiento de una máquina, etc. Al tratar entre equipos diferentes, es de gran utilidad el que esta herramienta disponga de estados de transición. Una vez solucionado el problema, Gestión de Entornos transita el estado de la jtrac para que desde pruebas se verifique si ya está OK la incidencia para proceder a su cierre o a su rechazo.

Los estados son los siguientes:

- **Abierta**: Cuando se localiza una incidencia en un entorno se abre una jtrac especificando los datos que se requieren en la jtrac en cuanto a la incidencia, como son: entorno, sistema y problema localizado.

- **En-Tratamiento**: El equipo de Gestión de Entornos recoge los datos sobre la incidencia y comienza su tratamiento para solucionarla.

- **Tratada**: Una vez solucionada la incidencia, el equipo de Gestión de Entornos transita la jtrac al estado Tratada para que el equipo de integración compruebe el estado de la misma.

- **Resuelta**: Una vez comprobado que se ha solucionado el problema de entorno por el que se abrió la jtrac, el equipo de integradas transita la jtrac al estado Resuelta.

- **Rechazada**: Cuando la incidencia ya haya sido tratada por el equipo de Gestión de Entornos, si al realizar la validación de la misma por parte del equipo de integración, se comprueba que el fallo en el entorno se sigue produciendo, se rechaza la jtrac, especificando el problema de nuevo para que el equipo de Gestión de Entornos pueda tratarla otra vez.

- **Cerrada**: Cuando una incidencia de entorno se haya resuelto, el equipo de Gestión de Entornos procede a cerrarla, que es el estado final.

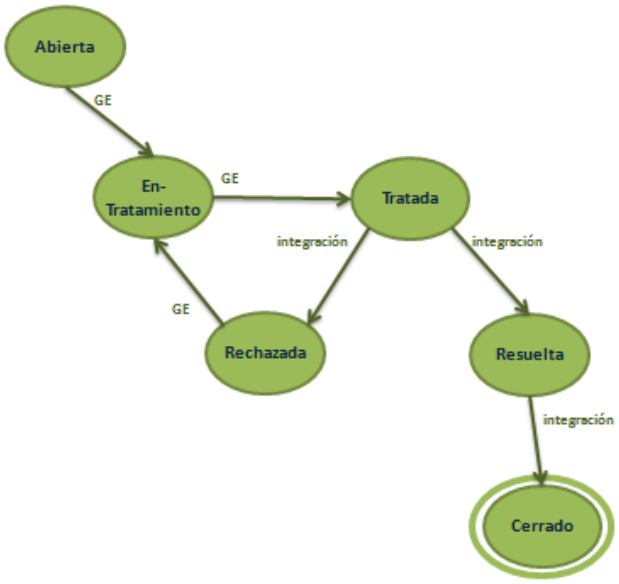


Figura 4-11: Estados Jtrac GE

4.2.5 Bases de Datos

Desde el equipo de integración el trabajo de tester es realmente muy técnico, y por suerte se puede acceder a las bases de datos que el cliente tiene preparadas en los entornos no productivos, lo que hace aún más interesante este trabajo.

Con el acceso a bases de datos e incluso con permisos de escritura en la mayoría de ellas, las pruebas de integración se convierten en mucho más que en meras pruebas funcionales de caja negra. Además de que facilita muchísimo algunas tareas, normalmente es mucho más rápido y eficaz para los tester la preparación de datos directamente sobre las bases de datos, evitando así cualquier problema de comunicación entre algún sistema o incluso entre un sistema y su propia base de datos al crear recursos sobre las aplicaciones.Por ejemplo, es muy útil para la preparación de recursos que hacemos para el usuario como parte del soporte que el equipo de integradas realiza al equipo de aceptación. Debido a que los usuarios no son técnicos, entre otras cosas, cualquier tema relacionado con bases de datos es inmediatamente una tarea del equipo de integración.

Otro ejemplo que puedo resaltar en cuanto a la utilización de bases de datos, es que modificando ciertos datos podemos lanzar una prueba de manera que simulamos que el origen sea otro

sistema desde el que realmente nosotros no podemos ejecutar, lo que nos permite ejecutar pruebas que se lanzarían de manera síncrona en un entorno real, pero que nosotros podemos ejecutar de manera asíncrona en el entorno de pruebas.

El acceso a las bases de datos también nos permite ir analizando on-line mientras se ejecuta una prueba cómo se comporta la misma en cuanto a la utilización de la base de datos. Permitiendo incluso una mejora en la localización de posibles errores que puedan tener su origen en la propia base de datos y por tanto, al fin y al cabo, lo que nos ofrece es una mayor visibilidad de lo que sucede en cada prueba, lo que sin duda ayudará a sumar calidad al resultado final del software probado.

4.2.6 SecureCRT

SecureCRT [7] es una herramienta mediante la cual nos conectamos a las máquinas de los diferentes sistemas, principalmente para visualizar logs. Se trata de un excelente emulador que tiene todas las funciones de un cliente Telnet y soporta el protocolo SSH (Secure Shell). Permite realizar logins automáticos, nombrar sesiones para los diferentes hosts, tiene funciones de impresión, selección de los colores de visualización de los logs, limitar el número de líneas a mostrar en la scrollbar, ...

La gran ventaja de utilizar SecureCRT reside en que no solo ejecutamos una prueba funcional, si no que podemos estudiar su comportamiento técnicamente al poder analizar los logs y así su comportamiento a un nivel inferior. Asimismo, existen dos formas a la hora de ver los logs:

- **On-line**: Permite ver el tráfico de datos que genera la ejecución de una prueba a la vez que se está produciendo. Lo que hacemos en estos casos es lanzar una prueba desde una ventana, mientras que desde otra ventana tenemos el tráfico de datos desde SecureCRT abierto para ver los logs que va generando la ejecución de la prueba.

- **A posteriori**: Los logs diarios que se generan en cada máquina son almacenados continuamente, de manera que consultando los ficheros donde se hayan almacenado los logs que hayan generado una prueba ejecutada, podremos estudiar el tráfico correspondiente a dicha prueba.

Posiblemente el inconveniente de ver tráfico de datos es que hay que formar a los testers para que sepan leer los logs, lo cual no es sencillo. Una de las dificultades reside en que al consultar el tráfico que genera una prueba (a menos que se apliquen cierto tipo de filtros si se sabe exactamente qué se espera visualizar y no siempre es así, por ejemplo, si se trata de una nueva funcionalidad), lo que en realidad veremos en los logs será todo el tráfico que se haya generado en esa máquina, por lo que seguramente no solo estarán los logs de la prueba que buscamos, si no que habrá muchísimos más, que por supuesto, dependen de la cantidad de personas que haya ejecutando pruebas sobre esa máquina. Pero una vez se aprende, sin duda alguna aporta mucho interés a las evidencias de las pruebas que se ejecutan, pues no solo se demuestra el resultado que se obtiene con por ejemplo, pantallazos, si no que con los logs se registra el comportamiento exacto de la prueba ejecutada.

Además, los logs aportan un valor incondicional a los tester, que de nuevo, hace denotar que las pruebas integradas, no son únicamente pruebas funcionales de caja negra. Los logs aportan a las pruebas integradas un conocimiento técnico a bajo nivel sobre el funcionamiento de los sistemas sobre los que se esté trabajando. Por ejemplo, una prueba funcional podría ser el pago de un artículo con tarjeta de crédito, la prueba funcional sería OK si la compra se realiza de manera satisfactoria. Tal y como se ejecutaría la prueba, al disponer de los logs que se producen al ejecutar esta prueba, la prueba en sí consistiría además de asegurar de que la compra se produce de manera satisfactoria, principalmente en validar que también se realizan unas operaciones internas y ajenas al cliente final como pueden ser: la validación de la numeración de la tarjeta de crédito, la comprobación de que el cliente existe y efectivamente es el asociado a ese número de tarjeta, la respuesta afirmativa indicando que dicho cliente puede realizar compras validando si dicha numeración se encuentra en alguna lista negra, la justificación de que se realiza el cargo de la compra al cliente correspondiente y por último, la ratificación de que el stock del producto comprado se reduce en uno.

4.2.7 Email

Así es, el correo electrónico es una herramienta fundamental en el equipo de pruebas de integración. Esla herramienta principal mediante la cual nos comunicamos con el resto de equipos implicados en el proceso, que son muchos. Se intercambian correos a diario con desarrollo, con el cliente, con los jefes de proyecto, con el equipo encargado de las instalaciones (SCM), con Gestión de Entornos, con los propios compañeros de pruebas... Desde luego su finalidad es agilizar la comunicación lo máximo posible mientras se deja constancia por escrito de todo lo dicho para evitar confusiones.

Lo cierto es que un mail puede dar mucho juego, en el sentido en el que se puede aportar todo tipo de detalles mediante texto, imágenes, documentación adjunta... todo lo que queramos explicar. El correo electrónico es una fuente potentísima de información que se intercambia continuamente entre todos los equipos que forman parte de las pruebas.

4.3 Ciclo de Vida del Caso de Estudio

El modelo de ciclo de vida que hemos aplicado es el que vamos a ver a continuación. Iré detallando las actividades que realizamos en el ejemplo práctico, ordenadas en las diferentes fases del ciclo de vida de las pruebas integradas.

En este punto, me gustaría añadir alguna explicación en cuanto a equipos que formen parte del equipo de integradas se refiere. En la figura 4-12 muestro esos equipos y las interacciones entre ellos.

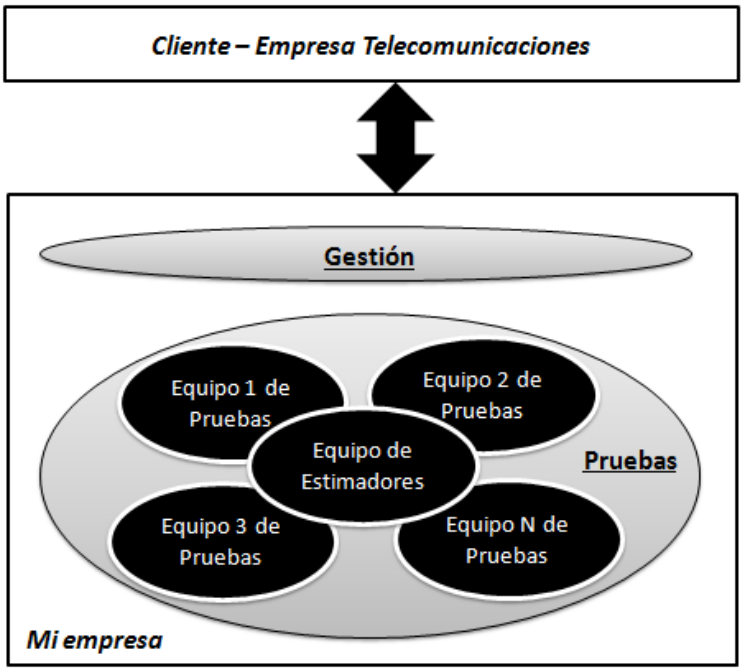


Figura 4-12: Organigrama

Por aclarar, existe una empresa de telecomunicaciones. Por otro lado, dentro del proyecto de telecomunicaciones se han dividido en primer lugar dos grandes grupos: gestión, que se encarga de las fases más tempranas de los proyectos, como es la planificación, así como de la monitorización y control. Y pruebas, que lo componen varios equipos, teniendo cada equipo un líder. Además, una serie de personas de cada equipo de pruebas forman parte del equipo de estimadores, que suele coincidir con los líderes de cada equipo, pues son los más cualificados.

4.3.1 Planificación del Caso de Estudio

El primer paso para un proyecto comienza en el momento en que el cliente nos indica que dispone de un nuevo proyecto, en este caso de Reingeniería Web. Para poder comenzar a trabajar, es imprescindible que el cliente abra una jtrac de estimaciones, que en este caso será la TESTEL-1213, que estará en estado Abierta, donde sea el propio cliente quien incluya la documentación de la que disponga del proyecto, la cual suele ser, como mínimo el SO. Y además necesitamos que incluya las fechas aproximadas de entrega de software, para las que se está

barajando el 25 de agosto. Pero sobretodo, la fecha de subida a producción o el PAP, que en un 95% de las veces se efectúa en domingo y para la release a la que pertenece Reingeniería Web se fija en el domingo 21 de septiembre de 2014.

El siguiente paso consiste en realizar la estimación por parte de un equipo de estimadores perteneciente al equipo de integradas. Para ello, el estado de la jtrac TESTEL-1213 del proyecto se transitará al estado En-Estudio.

El equipo de estimadores debe ser personal cualificado, con experiencia en pruebas y que conozca muy bien el negocio pues es quien determina qué tipo de pruebas son las necesarias para cubrir la funcionalidad que se entregue con cada nuevo proyecto. Para realizar las estimaciones en sí, el equipo de estimadores dispone de una excel donde se van introduciendo las funcionalidades sobre las que posteriormente se realizará un plan de pruebas. La excel está automatizada, de manera que dependiendo del tipo de funcionalidades que se vayan añadiendo, calcula automáticamente el coste y tiempo estimado que conllevará la realización de pruebas para cubrir dicha funcionalidad. Además, en nuestro caso particular, las estimaciones también contienen la funcionalidad que suponemos va a ejecutar el usuario en sus pruebas de aceptación, ya que es el equipo de integración el encargado de dar soporte técnico al usuario y por tanto, debe añadirse para contemplarse como coste asociado al proyecto.

Si durante la realización de la estimación se tienen dudas sobre el contenido del proyecto, en todo momento se podrá hablar bien vía email o bien vía teléfonica (preferiblemente vía email para dejar constancia de todo lo dicho) con el jefe de proyecto que pertenece al cliente y es quien lidera al completo el proyecto. En el caso particular de Reingeniería Web, en la documentación no se especifican tres aspectos:

- El tipo de recursos que podrán beneficiarse de la nueva promoción.
- El tipo de recursos para los que va a aplicar la nueva forma de logado.
- Las operaciones a las que aplicará el encolado.

Por supuesto, estas dudas son remitidas al jefe de proyecto vía email y nos remite la siguiente contestación:

- Cualquier tipo de recurso puede beneficiarse de la nueva promoción, eso sí, tened en cuenta que la nueva promoción se activará en FACTPRE, siendo éste el maestro de la promoción.

- Incluye el logado de los clientes monolíneas residenciales tanto pospago como prepago y empresa pospago.
- Se realizará el desarrollo del encolado para las operaciones:
    - Recarga con tarjeta de crédito o 06001.
    - Alta, baja o modificación de una promoción o 09030.
    - Cambio de tarifa o 26032.
    - Actualización de servicios o 26033.

Una vez resueltas las dudas sobre la funcionalidad que se va a implementar, se finaliza la excel de estimaciones, se adjunta a la jtrac TESTEL-1213 y se transita la jtrac TESTEL-1213 al estado Estimada.

Identificación de los Casos de Prueba Funcionales

Funcionalidad o UC	Nombre de la transacción	Sistemas	Complejidades	Fase de pruebas
Logado en WEB	Consulta de Servicios para un Residencial Pospago desde WEB	WEB	Medio	Integración y UAT
	Consulta de Servicios para un Residencial Prepago desde WEB	MDW		
	Consulta de Servicios para un Empresa Pospago desde WEB	FACTPOS		
		FACTPRE		
Nº de transacciones	3			
Encolado	Recarga por Tarjeta de Crédito 06001 desde WEB	WEB	Complejo	Integración y UAT
	Verificar que se tras encolarse, se reenvía la petición	MDW		
	Alta promoción 09030 desde WEB	WEB		
	Verificar que se tras encolarse, se reenvía la petición	MDW		
	Cambio de Tarifa 26032 desde WEB	WEB		
	Verificar que se tras encolarse, se reenvía la petición	MDW		
	Actualización de Servicios 26033 desde WEB	WEB		
	Verificar que se tras encolarse, se reenvía la petición	MDW		
Nº de transacciones	8			
Alta Promoción y Re-Alta - 09030	Alta Promo desde WEB	WEB	Medio	Integración y UAT
	Comprobar logs de FACTPOS	FACTPOS		
	Intentar realizar un Alta Promo cuando la Promo ya esté dada de alta previamente desde WEB	WEB		
	Comprobar logs de FACTPOS	FACTPOS		
Nº de transacciones	4			
Consulta de Promociones - 09020	Consulta de Promociones desde WEB	WEB	Simple	Integración y UAT
	Comprobar el resultado de FACTPOS, FACTPRE, RESER			
Nº de transacciones	2			
Baja Promoción y Re-Baja - 09030	Baja Promo desde WEB	WEB	Medio	Integración y UAT
	Comprobar logs de FACTPOS	WEB		
	Intentar realizar una Baja Promo cuando la Promo ya esté dada de baja previamente desde	WEB		
	Comprobar logs de FACTPOS	FACTPOS		
Nº de transacciones	4			

Figura 4-13: Funcionalidad incluida en la Estimación

En este punto, la estimación al completo pasa a manos del jefe de proyecto, quien se encarga de revisar que la funcionalidad que se ha introducido en la estimación se ajusta a lo que requiere el proyecto y decide si aprueba la estimación tal y como está, o si requiere algún tipo de modificación, para lo que se pondrá en contacto con el equipo de estimadores y juntos llegarán a

un acuerdo. Por otra parte, un equipo de costes del cliente estudia la aprobación del coste asociado a la realización de las pruebas que se estimen.

En el caso de Reingeniería Web no se producen problemas en este aspecto y se aprueba el alcance funcional y el coste asociado. Esto se refleja cuando el cliente transita la jtrac de estimaciones al estado Aprobada. Es en este momento en el que se puede decir que se ha estimado que las pruebas de integración tengan una duración de ocho jornadas y las pruebas de usuario de diez jornadas. Además, en la figura 4-14 puede verse que se indica también que se dará soporte durante cinco jornadas a las pruebas de postproducción. Las pruebas de postproducción las realiza el usuario y sobre el entorno real del cliente al que llamamos producción. El soporte a las pruebas de postproducción por parte del equipo de integradas consiste en que cuando se localice un posible error, se solicita al equipo de integradas que justifique con una evidencia cómo fue esa ejecución en el entorno de integración, en caso de que lanzase esa prueba en concreto. Lo cual demuestra si esa prueba se realizó correctamente por parte del equipo de integración o si por el contrario, sucedió algún tipo de error durante las pruebas de integración que se pasó por alto, y en este caso, acarrearía algún que otro problema al equipo por un trabajo mal realizado. El resto de tareas sobre la postproducción son cosa del cliente.

Una vez aprobada la estimación, se efectúa un estudio de los riesgos en el que participa el estimador y el equipo de gestión que también pertenece a integradas, que es quien supervisa los riesgos detectados. De hecho, en Reingeniería Web se detectan los siguientes **riesgos** que se comunican al cliente de forma inmediata:

- La fecha de entrega de software es demasiado cercana a la fecha de PAP impidiendo que haya un pequeño margen para la finalización de las pruebas integradas y de usuario, aún incluso aunque se solapasen durante varias jornadas las pruebas de integración y las de aceptación. Sería conveniente intentar adelantar en la medida de lo posible la entrega del software para que la realización de las pruebas integradas, estimada en ocho jornadas y la realización de las pruebas de usuario, estimada en diez jornadas quepan con un margen de al menos un par de días antes de la subida a producción, intentando evitar en la medida de lo posible el solape de ambas pruebas por los problemas que lleva asociado este tipo de situación, como son que el software no esté lo suficientemente depurado para que sea ejecutado por los usuarios.

- Existe cerrada una campaña publicitaria por parte del cliente para la primera semana de octubre. Se trata de unas mejoras que van a anunciar junto con la oferta de una nueva promoción por primera vez sobre televisión en el móvil que forma parte de Reingeniería Web. Es un riesgo asociado totalmente al anterior, porque cualquier retraso podría alterar el lanzamiento de la campaña publicitaria.

En esta parte entra en juego un factor que suele condicionar a las pruebas de integración y a su vez a las pruebas de usuario (pues suceden justo después) y es que son las últimas fases antes de la subida a producción de un proyecto. Llegados a este punto, el proyecto suele venir ajustado en tiempo, ya que normalmente se producen retrasos en las planificaciones iniciales, y ello conlleva a que se ejerza sobre el equipo de integradas una presión añadida para que su ejecución sea lo más rápida posible y que el soporte a las pruebas de usuario se realice evitando todos los problemas posibles. Desde el equipo de integración, reivindicamos continuamente nuestro hueco en el ciclo de vida de cualquier proyecto y es cierto que vamos consiguiendo mejoras en las planificaciones de los tiempos de los proyectos, pero la sombra de constituir el final de la cadena siempre nos persigue. No hay que olvidar que la finalidad es obtener un software depurado y de calidad y para ello es imprescindible otorgar el tiempo suficiente para la realización de pruebas.

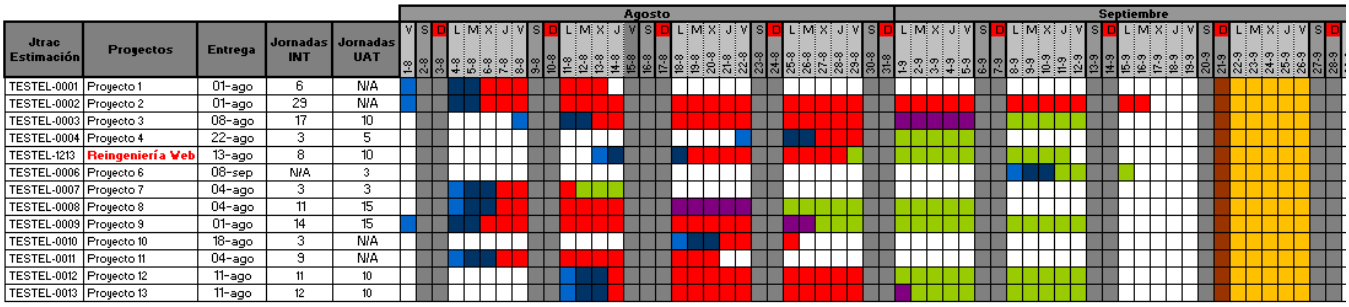
Por esto que indico, cobra mucha importancia el hecho de registrar todo tipo de retrasos o bloqueos que impidan el avance de las pruebas integradas tal y como estaba previsto. El objetivo único y explícito es poder demostrar, si llegase el caso, que un proyecto no se ha finalizado a tiempo alegando los motivos registrados y por tanto, liberando de la responsabilidad de tal hecho al equipo de integradas. Por ejemplo, que se produzcan fallos en el entorno en los días de pruebas que impidan el avance esperado de las mismas.

Días más tarde recibimos contestación al riesgo escalado y el cliente acepta realizar la entrega de software el 13 de agosto, casi semana y media antes de lo previsto para conceder un margen mayor a la ejecución de las pruebas integradas y de usuario.

A continuación, se realiza un estudio entre el equipo de gestión, y el cliente, para considerar la release en la que debe o tiene que probarse. El estudio se realiza en base a los siguientes factores:

- Fecha de lanzamiento prevista ya sea por campaña publicitaria dispuesta, por una nueva normativa legislativa o por objetivos de la empresa.

- Estudio de la capacidad de la release al introducir el proyecto en dicha release. Tanto de la capacidad global para asumir un proyecto más en cuanto a recursos se refiere, como de la capacidad por funcionalidad específica que tenga, ya que se conoce de antemano la capacidad en base a los conocimientos técnicos del equipo de pruebas, pudiendo asumir un número limitado de proyectos que afecten a una funcionalidad concreta que por su carácter complejo no suele ser del conocimiento de todo el equipo.
- Si por exceso de la capacidad de la release en la que se desee encajar un proyecto, no puede entrar, entonces el cliente debe determinar qué proyecto tiene una mayor prioridad indicando así su indispensable entrada en la release. Como por el bien de todos, y sobre todo con el fin de cumplir las planificaciones que se realizan, no se debe sobrepasar la capacidad de una release y es por ello, que en casos como este, el equipo de gestión realiza un estudio de los proyectos que podrían sacarse de la release sin que supongan un inconveniente para el cliente. Así se ajustaría la capacidad de proyectos que pueden ser probados en una release. Sí debo añadir que en casos excepcionales, se asume la totalidad de los proyectos que indica el cliente y para ello se aumenta la plantilla de recursos que se van a dedicar a ejecutar pruebas.



Jtrac Estimación	Proyectos	Entrega	Jornadas INT	Jornadas UAT
TESTEL-0001	Proyecto 1	01-ago	6	N/A
TESTEL-0002	Proyecto 2	01-ago	29	N/A
TESTEL-0003	Proyecto 3	08-ago	17	10
TESTEL-0004	Proyecto 4	22-ago	3	5
TESTEL-1213	Reingeniería Web	13-ago	8	10
TESTEL-0006	Proyecto 6	08-sep	N/A	3
TESTEL-0007	Proyecto 7	04-ago	3	3
TESTEL-0008	Proyecto 8	04-ago	11	15
TESTEL-0009	Proyecto 9	01-ago	14	15
TESTEL-0010	Proyecto 10	18-ago	3	N/A
TESTEL-0011	Proyecto 11	04-ago	9	N/A
TESTEL-0012	Proyecto 12	11-ago	11	10
TESTEL-0013	Proyecto 13	11-ago	12	10

Figura 4-14: Planificación Release

En la figura 4-14 podemos ver el conjunto de proyectos que forman parte de la release de verano a la que pertenece Reingeniería Web. Y el código de colores se explica en la leyenda de la figura 4-15, que muestro a continuación:

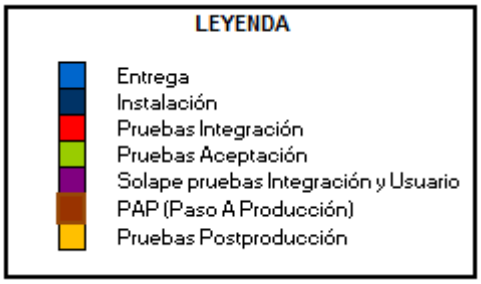


Figura 4-15: Leyenda Colores Planificación

4.3.2 Análisis y Diseño del Caso de Estudio

La fase de análisis comienza realizando un análisis detallado sobre la documentación disponible sobre un proyecto. La documentación que analizamos, es la siguiente:

- **PO (Proyect Outline)**: Agrupa los requisitos a modificar y/o añadir sobre el software existente solicitados por el usuario. Los diferentes sistemas implicados explican las medidas a tomar en cada punto y se detalla qué y cómo va a modificarse.

- **SO (Solution Options)**: Es el estudio de viabilidad del proyecto. Se realiza a partir del PO, pero es mucho más completo y detallado, convirtiéndose así en uno de los documentos imprescindibles para la realización de un plan de pruebas.

- **CHLD**: Es un informe sobre las funciones específicas del proyecto. Va asociado a FACTPOS que es quien desglosa el funcionamiento interno de sus funciones en este documento.

- **IPS (Informe Pruebas Sistema)**: Es un informe detallado de las pruebas y resultados obtenidos por el desarrollo de cada sistema. Puede servir de guía para las pruebas integradas porque aclara qué tipo de modificaciones se han realizado y sobre qué módulos en concreto. Pero es importante no ceñirse a él completamente para diseñar el plan de pruebas, pues se podría caer en una redundancia de las pruebas.

En el análisis englobamos una serie de tareas:

- Toda la documentación que el cliente nos ha facilitado sobre Reingeniería Web hasta el momento, está disponible en un repositorio común en el que vamos almacenando la documentación de cada proyecto ordenados por release. En caso de que no fuese así, se solicitaría la documentación que echemos de menos al jefe de proyecto. Por otro lado, hay que tener en cuenta los reversionados de la documentación, para lo que preguntamos al jefe de proyecto si la versión 1 que tenemos del SO, es la definitiva, lo cual nos confirma.

- Disponemos solo del SO, pero lo cierto es que para este proyecto es suficiente para realizar el diseño del plan de pruebas. Aun así, como deberían habernos hecho llegar el PO y el IPS, y dado que los hemos solicitado al jefe de proyecto sin éxito alguno, y que el proceso de pruebas ha llegado hasta la documentación. Procedemos de manera inmediata a la apertura de una TI de documentación, la WEB_TI_1762 la cual engloba la falta de entrega de los dos documentos. Con esta metodología, que aplicamos desde solo hace unos meses, se obliga a mejorar la documentación ofrecida por parte de los desarrollos. Es una TI de severidad baja y no es bloqueante para los casos de prueba que se van a diseñar. En tan solo unos días, se resuelve la WEB_TI_1762 con la entrega del PO y del IPS.

Tras el estudio de lo contenido en el SO, se obtiene la siguiente información, que paso a exponer y que será la base fundamental sobre la que realizaremos el diseño del plan de pruebas:

- **Propósito**: Se pretende mejorar el aplicativo Web, optimizando tanto el modo en el que se obtiene la información, como el modo en que se representa.

- **Sistemas impactados**: Tal y como puede verse en la figura 4-16 los sistemas con impacto alto son Web y Middleware. Los facturadores y CRM se ven reflejados porque participan en las operaciones implicadas pero no han realizado ninguna actualización sobre su software para este proyecto. Con esta información, concluimos que solo realizarán entrega de software los sistemas: Web y Middleware.

Sistema - Aplicación	Área	Impacto
Web	DSI	Alto
MDW	API	Alto
FACTPOS	BILLING	
FACTPRE	BILLING	
CRM	Cliente	

Figura 4-16: Sistemas Impactados

- **Impacto total en Web**:
    - Se modifica la maquetación de las pantallas de la Web.
    - Sustitución de los flujos de negocio que actualmente residen en Web, eliminando el flujo actual e introduciendo un nuevo flujo de datos, el servicio 12010 para el logado en la Web de los clientes.
    - Creación de una nueva promoción denominada New TV que será visible en la Web, desde donde la podrán gestionar los propios clientes que podrán activarla, desactivarla y consultarla.
    - Encolado de peticiones de Web para que en el caso en el que se produzca un error en los sistemas finales a la hora de gestionar algún servicio, como es un timeout, sea posible almacenar dicha petición y relanzarla minutos más tarde. Se creará un demonio cuya función será la de monitorizar si Middleware responde correctamente o si existe algún problema.En caso de que se detecten indisponibilidades en los servicios Middleware, se introduce la petición lanzada en una cola para su próximo relanzamiento, el cual es asíncrono.Por lo que si la petición era síncrona, se transformará en asíncrona en caso de que se produzca este error.

- **Impacto total en Middleware**:
    - Se elimina un servicio por otro para optimizar el tiempo de consulta de datos personales de los clientes. Se trata de un servicio nuevo para el logado en Web, que se realizará a través del servicio 12010 y que obtendrá información de FACTPOS.

- **Impacto total en FACTPOS**:
    - Optimización de la operación de consulta 09020 para evitar utilizar los contextos hasta ahora empleados de cliente/contrato. El cambio será transparente para el resto de sistemas que utilizan dicha función.
- **Impacto total en FACTPRE**:
    - Inclusión de la nueva promoción New TV que se facturará desde prepago como sistema maestro de la promoción.
- **Matriz de cobertura de requisitos de usuario**: Se trata de las especificaciones solicitadas por el usuario y que serán de cumplimiento al 100% por su viabilidad de implementación, son los siguientes:

ID-RU	Nombre de Requisito	Descripción Funcional
1	Edición de pantallas en Web	Se crearán las plantillas y componentes necesarios para permitir la edición de contenidos.
2	Simplificación y optimización de las llamadas a MDW	Se optimizarán los accesos a las consultas 09020 a FACTPOS.
3	Promoción New TV	Nueva promoción Promo New.
4	Nueva operación de logado en Web	Introducción nuevo servicio 12010, mediante el cual se modifica el acceso a la web. Incluye el logado de los clientes monolíneas: Residenciales tanto Pospago como Prepago y Empresa Pospago.
5	Encolado de peticiones a MDW	Se creará un proceso para, en caso de que los sistemas finales no estuvieran disponibles, no se envíen peticiones de modificación y se queden almacenadas para relanzarlas una vez se estabilicen los sistemas.
6	UATs	Soporte técnico a las pruebas de usuario.

Figura 4-17: Matriz de Requisitos

- **Hitos clave del proyecto**:

Hito	Fecha	Comentario
Subida a Producción	21-sep	Va asociado al lanzamiento de una campaña publicitaria.

Figura 4-18: Hitos de Reingeniería Web

- **Estrategia de pruebas**:

Tipo de Validación	Prioridad
Logado	Alta
Encolado	Media
Promoción	Baja

Figura 4-19: Estrategia de Pruebas

El siguiente paso consiste en el **diseño** de los casos de prueba. En primer lugar, escogeremos una técnica de diseño que se adapte lo mejor posible al tipo de pruebas que vamos a realizar. Nos encontramos ante el diseño de las pruebas integradas donde el principal objetivo es probar la conexión y comunicación entre los diferentes sistemas. Desconocemos cómo funciona cada sistema por dentro, de manera que nos centraremos en las técnicas concernientes a las pruebas de caja negra y más en concreto, nos ceñiremos a las clases o particiones de equivalencia.

Al desglosar la información contenida en el SO podemos deducir que las **particiones de equivalencia** serían las siguientes:

- Acceso a la aplicación desde un nuevo servicio. Corresponde al requisito 4 del SO.
- Encolado de operaciones. Es el requisito 5 del SO.
- Promoción nueva. Es el requisito 3 del SO.
- Optimización de las consultas. Es el requisito 2 del SO.

Para cada partición, los **casos OK** serían:

- **Acceso a la aplicación** desde un nuevo servicio: Al tratarse de un nuevo servicio es conveniente realizar una prueba por cada tipo de respuesta que pueda producirse. Por tanto, se crea un caso de prueba por cada tipo de cliente que puede acceder a la aplicación, que en este caso son los clientes monolíneas residenciales pospago y prepago y los empresa pospago. Es decir, tres casos de prueba para el logado, cada uno con un tipo de cliente porque se devolverán respuestas diferentes en cada caso. Hay que

tener especial cuidado con las redundancias, y en este caso en concreto, entraríamos en ella si diseñásemos más de una prueba en la que se acceda con el mismo tipo de cliente, por ejemplo, con dos clientes residenciales prepago para los que la respuesta en sí en cuanto a campos sería la misma, ya que el que solo sea diferente el contenido de los campos se considera la misma prueba.

- **Encolado** de operaciones: Se diseña un caso por cada operación que se va a encolar, más en concreto se tratará de un caso para la recarga por tarjeta de crédito o la que para nosotros es la 06001, otro para la 09030, que es un alta/baja/modificación de una promoción, otro para los cambios de tarifa o 26032y otro para la 26033que es la actualización de los servicios de un cliente. En todos ellos la finalidad absoluta es asegurarse de que se produce el reenvío de las peticiones al producirse un timeout. Para completar los casos correctamente, lo suyo sería que con una de las peticiones de reenvío finalmente se conecte con el sistema destino. La completitud se produciría si el sistema destino contesta, no es relevante cómo sea el tipo de contestación, OK o KO.

- **Promoción nueva**: En el caso de las promociones de telefonía, la experiencia nos indica que siempre se probarán al menos tres casos: el alta, la consulta y la baja de la promoción, que es justo lo que haremos en este proyecto. Asimismo, suele probarse a modo de regresión una consulta donde no figure la nueva promoción como contratada, con el fin de asegurar que la operación de consulta, que es muy utilizada, sigue funcionando tal y como se espera de ella, a pesar de que no se hayan realizado cambios sobre esta operación. Existen proyectos en los que además se habla sobre la compatibilidad de promociones, en los cuales se añadirían casos de prueba para verificar que diferentes promociones pueden estar dadas de alta para el mismo cliente, pero no es el caso.

- **Optimización** de las consultas: Nos encontramos en el entorno de integradas, que es un entorno no productivo y es por ello que las máquinas que lo componen no tienen ni mucho menos la capacidad del entorno real. El caso de optimización de consultas lo vamos a diseñar porque desarrollo FACTPOS nos lo ha pedido explícitamente vía mail porque con la ejecución de la prueba quieren comprobar ciertos valores en su sistema, de cara a estudiar el comportamiento que podría tener esta misma consulta en el entorno

real. Este tipo de peticiones suceden de vez en cuando y sin problema alguno, diseñaremos la prueba que nos indica desarrollo, la ejecutaremos y enviaremos a desarrollo los resultados obtenidos. Eso sí, no daremos el caso por OK por nuestra cuenta, esperaremos a que sea desarrollo quien nos indique cuando la prueba esté verificada OK. Por tanto, diseñamos un caso de optimización de consultas, que se trata de la operación 09020 que lanzaremos desde el sistema CRM que es el maestro de clientes de la teleoperadora.

Para cada partición, los **casos de error o KO** para comprobar la robustez del sistema y validar cómo se comportaría el sistema ante datos que en principio no espera, serían:

- **Acceso a la aplicación** desde un nuevo servicio: la propia Web controla que en el campo donde se introduce el número de teléfono no puedan introducirse valores que no sean numéricos y que no comiencen por seis. Por tanto, la única forma de crear un caso de error sería introduciendo un número de teléfono que no esté registrado en el facturador y es lo que haremos. Para ello, escogeremos un teléfono al azar y comprobaremos que no se encuentre registrado en bases de datos y realizaremos la prueba con él.

- **Encolado** de operaciones: para esta funcionalidad, el error o KO sería que no se encole, y no lo vamos a tratar como un caso de prueba, puesto que en este caso un error no sería indicativo de robustez sino de fallo. Por tanto, para esta partición de equivalencia no existirían casos de error.

- **Promoción nueva**: Es la Web la encargada de mostrar la disponibilidad de la nueva promoción solo para los clientes que puedan contratarla pero en este proyecto en concreto, la promoción es accesible para cualquier tipo de cliente, por lo que no tendremos que realizar este tipo de validación. Lo que sí controlaremos es el hecho de intentar contratar la promoción cuando ya esté contratada e intentar dar de baja la promoción si no se tiene contratada, porque por experiencia sabemos que Web falla en su diseño original en que es posible contratar algo que ya tengas contratado. En conclusión, son dos casos de prueba de error a los que denominamos Re-alta y Re-baja.

- **Optimización** de las consultas: Esta operación la lanzamos para que desarrollo valide una serie de datos de tiempos en su sistema, no realizaremos caso de error.

Al ir profundizando en la información contenida en toda la documentación a lo largo del diseño de los casos de prueba, observamos que falta el código de la promoción nueva y en el caso de los encolados, nos falta por saber el tiempo que transcurrirá entre el error de Middleware y el relanzamiento de la operación, además del número de reintentos que se producirán como máximo. Este tipo de aspectos a veces quedan sin atar, pero al detectarlos, inmediatamente escribimos un email al jefe de proyecto en el que solicitamos respuesta a las dudas indicadas.

En la contestación el jefe de proyecto nos indica lo siguiente:

- El código de la promoción New TV es 2221.
- El tiempo de espera para reenviar una petición en caso de error en Middleware es de:
    - Cinco minutos en los flujos de operaciones.
    - Seis minutos en las operaciones punto a punto.
- Se relanzará dos veces como máximo cada operación que se encole.

Dicha información será añadida al detalle de los casos correspondientes para tenerlos en cuenta a la hora de ejecutarlos.

Seguramente parece una obviedad pero es importante asociar cada caso de prueba a un requisito del proyecto, ya que no hay que olvidar que las pruebas se realizan para comprobar el funcionamiento de unos requisitos establecidos. Toda prueba sin trazabilidad de requisitos, debería ser un cambio de alcance, una petición específica del cliente o una prueba de regresión. En el caso de Reingeniería Web, cada partición de equivalencia que hemos creado, pertenece a un requisito diferente. Podemos visualizar la idea en la tabla 4-17 de requisitos de usuario y finalmente lo veremos mejor plasmado cuando finalicemos el plan de pruebas.

Una vez finalizado el diseño de los casos, el líder del equipo de integradas que se encarga de cada proyecto, debe revisar el diseño hecho para corregirlo si fuese necesario. En nuestro caso, el

proyecto de Reingeniería Web, lógicamente cayó en mi equipo, yo misma realicé el diseño y al ser yo también la líder del equipo, la revisión se da por hecha.

Por último, apuntaría que trabajo para una empresa que se basa en costes (como todas), por lo que hay que ceñir el diseño, en la medida de lo posible, a la estimación que se realizó en su momento sobre el proyecto en cuestión, porque el acuerdo entre las empresas es el cobro en base a las estimaciones. Por tanto, si finalmente el plan de pruebas resulta ser mucho mayor a la estimación realizada para un proyecto, supondrá una pérdida de tiempo, esfuerzo y coste. En casos desorbitados, en los que por ejemplo, se pueda demostrar que en el momento en el que se realizó la estimación no se disponía de la documentación mínima necesaria, estando la estimación incorrecta o incompleta, podría solicitarse realizar una reestimación que se adecuase mejor a los requisitos finales. En nuestro caso en concreto, el diseño del plan de pruebas y la estimación de Reingeniería Web se corresponden perfectamente, por lo que evitamos este tipo de problema.

Toda lo que hemos plasmado a lo largo de este punto sobre el diseño, hay que crearlo en Quality Center:

- Añadimos el proyecto Reingeniería Web en Release.
- Añadimos el proyecto bajo el Requeriments WEB_RU_916 y MDWAE_RU_1468.
- Añadimos el proyecto Reingeniería Web como un proyecto más en TestPlan, que es donde plasmamos el diseño del plan de pruebas realizado.
- Por último, se añade el proyecto a TestLab, donde se incluyen todos los casos diseñados y donde realmente plasmaremos el estado de los casos según su ejecución.

Esta información la comento, porque no se me permite mostrar los pantallazos de Quality Center donde se refleja todo lo que indico, pero he realizado el plan de pruebas con todo su detalle en una tabla, la cual muestro a continuación:

Num. Caso	Descripción Caso	Precondiciones	Pasos a Seguir	Resultado Esperado	Req.	Sist.	Prioridad	Op.
1	Acceso aplicativo - Residencial Pospago	Datos: * Es necesario disponer de un recurso residencial pospago. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el pesado del sistema FACTPOS.	1. Entrar a la web. 2. Logarse con un recurso residencial pospago.	1. Se accede a la web. 2.Entra a la página de residenciales pospago y se muestran los datos del cliente logado.	4	WEB	Alta	12010
2	Acceso aplicativo - Residencial Prepago	Datos: * Es necesario disponer de un recurso residencial prepago. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado los pesados de los sistemas FACTPOS y FACTPRE.	1. Entrar a la web. 2. Logarse con un recurso residencial prepago.	1. Se accede a la web. 2.Entra a la página de residenciales prepago y se muestran los datos del cliente logado.	4	WEB	Alta	12010
3	Acceso aplicativo - Empresa Pospago	Datos: * Es necesario disponer de un recurso residencial pospago. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el pesado del sistema FACTPOS.	1. Entrar a la web. 2. Logarse con un recurso empresa pospago.	1. Se accede a la web. 2.Entra a la página de empresas pospago y se muestran los datos del cliente logado.	4	WEB	Alta	12010

4	Acceso aplicativo - KO	Datos: * Es necesario disponer de un recurso que no exista en FACTPOS. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el pesado del sistema FACTPOS.	1. Entrar a la web. 2. Logarse con un recurso residencial pospago.	1. Se accede a la web. 2. No puede logarse porque no localiza al cliente introducido mostrando un mensaje de error por pantalla.	4	WEB	Alta	12010
5	Encolado - Recarga	Datos: * Disponer de un recurso residencial prepago. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Hay que tirar el adaptador pesado de recargas del sistema FACTPRE para provocar el timeout y luego levantarlo para finalmente conseguir que comunique.	1. Logarse con un recurso residencial pospago en el sistema WEB. 2. Navegar hasta la página de recargas. 3. Realizar una recarga de saldo de la cantidad que se quiera.	1. Se accede a WEB. 2. Se visualiza el dashboard de recargas. 3. El sistema nos mantiene a la espera sin informar en ningún momento de que se esté produciendo el encolado. Es indiferente si la operación finalmente termina con un OK o un KO, lo importante es asegurarse de que se reenvía la petición a los 5 minutos de haberse producido el timeout.	5	WEB	Media	06001
6	Encolado - Alta de Promoción	Datos: * Disponer de un recurso. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Hay que tirar el adaptador pesado de promociones del sistema FACTPOS para provocar el timeout y luego levantarlo para finalmente conseguir que comunique.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Contratar una promoción cualquiera.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. El sistema nos mantiene a la espera sin informar en ningún momento de que se esté produciendo el encolado. Es indiferente si la operación finalmente termina con un OK o un KO, lo importante es asegurarse de que se reenvía la petición a los 5 minutos de haberse producido el timeout.	5	WEB	Media	09030

7	Encolado - Cambio de Tarifa	Datos: * Disponer de un recurso. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Hay que tirar el adaptador pesado de servicios del sistema FACTPOS para provocar el timeout y luego levantarlo para finalmente conseguir que comunique.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de servicios del cliente. 3. Realizar un cambio de tarifa a la tarifa que se quiera.	1. Se accede a WEB. 2. Se visualiza el dashboard de servicios del cliente. 3. El sistema nos mantiene a la espera sin informar en ningún momento de que se esté produciendo el encolado. Es indiferente si la operación finalmente termina con un OK o un KO, lo importante es asegurarse de que se reenvía la petición a los 5 minutos de haberse producido el timeout.	5	WEB	Media	26032
8	Encolado - Cambio de Servicios	Datos: * Disponer de un recurso. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Hay que tirar el adaptador pesado de servicios del sistema FACTPOS para provocar el timeout y luego levantarlo para finalmente conseguir que comunique.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de servicios del cliente. 3. Escoger un servicio de entre los disponibles para el cliente y contratarlo.	1. Se accede a WEB. 2. Se visualiza el dashboard de servicios del cliente. 3. El sistema nos mantiene a la espera sin informar en ningún momento de que se esté produciendo el encolado. Es indiferente si la operación finalmente termina con un OK o un KO, lo importante es asegurarse de que se reenvía la petición a los 6 minutos de haberse producido el timeout.	5	WEB	Media	26033

9	Alta Promo New TV	Datos: * Disponer de un recurso que no tenga activa la promoción New TV. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de promociones del sistema FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Contratar la promoción New TV.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. En la ventana, en primer lugar se indica que el alta está en curso. En segundo lugar, se muestra un mensaje por pantalla para notificar de que el alta de la promoción New TV se ha realizado correctamente.	3	WEB	Baja	09030
10	Re-Alta Promo New TV	Datos: * Disponer de un recurso que tenga activa la promoción New TV. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de promociones del sistema FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Contratar la promoción New TV.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. En la ventana, en primer lugar se indica que el alta está en curso. En segundo lugar, se muestra un mensaje de error notificando que se ha producido un error en el alta por tener la promoción ya activa.	3	WEB	Baja	09030
11	Consulta Promo New TV	Datos: * Disponer de un recurso que tenga activa la promoción New TV. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de consultas en los sistemas: FACTPRE, FACTOTROS y FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Realizar una consulta de las promociones contratadas. 4. Revisar los valores devueltos de unidades disponibles y unidades consumidas de New TV.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. Se muestran las promociones contratadas y entre ellas (o solo ella si no tiene más promociones contratadas) se encuentra la promoción New TV. 4. Se verifica que las unidades disponibles son 100 minutos y que las unidades consumidas son 0 minutos.	3	WEB	Baja	09020

12	Consulta Promo New TV con Consumo	Datos: * Disponer de un recurso que tenga activa la promoción New TV. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de consultas en los sistemas: FACTPRE, FACTOTROS y FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Realizar una consulta de las promociones contratadas. 4. Revisar los valores devueltos de unidades disponibles y unidades consumidas de New TV.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. Se muestran las promociones contratadas y entre ellas (o solo ella si no tiene más promociones contratadas) se encuentra la promoción New TV. 4. Se verifica que las unidades disponibles son 50 minutos y que las unidades consumidas son 50 minutos.	3	WEB	Baja	09020
13	Baja Promo New TV	Datos: * Disponer de un recurso que tenga activa la promoción New TV. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de promociones del sistema FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Escoger la promoción New TV y darla de baja.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. En la ventana, en primer lugar se indica que la baja está en curso. En segundo lugar, se muestra un mensaje notificando que la baja de la promoción New TV ha finalizado correctamente.	3	WEB	Baja	09030
14	Re-Baja Promo New TV	Datos: * Disponer de un recurso que no tenga activa la promoción New TV. Middleware: * Publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de promociones del sistema FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Escoger la promoción New TV y darla de baja.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. En la ventana, en primer lugar se indica que la baja está en curso. En segundo lugar, se muestra un mensaje de error notificando que se ha producido un error en la baja porque el cliente no tiene activa New TV.	3	WEB	Baja	09030

15	Consulta NO Promo New TV	Datos: * Disponer de un recurso que no tenga activa la promoción New TV. Middleware: * Se publica por librería de acceso y los demonios suelen estar arriba por defecto. * Tiene que estar levantado el adaptador pesado de consultas en los sistemas: FACTPRE, FACTOTROS y FACTPOS.	1. Logarse con el recurso en el sistema WEB. 2. Navegar hasta la página de promociones. 3. Realizar una consulta de las promociones contratadas.	1. Se accede a WEB. 2. Se visualiza el dashboard de promociones. 3. Se muestran las promociones contratadas (o ninguna si no tiene promociones contratadas) y entre ellas no se encuentra la promoción New TV.	3	WEB	Baja	09020
16	Optimización 09020	Datos: * Disponer de un recurso que con alguna promoción activa. Middleware: * Tiene que estar levantado el adaptador ligero de promociones de CRM. * Tiene que estar levantado el adaptador pesado de consultas en los sistemas: FACTPRE, FACTOTROS y FACTPOS.	1. Acceder al sistema CRM. 2. Llegar hasta la ventana de promociones. 3. Realizar una consulta de promociones para el recurso escogido. 4. Se almacena el resultado y se envía a desarrollo FACTPOS.	1. Se accede a CRM. 2. Se ve la ventana de promociones. 3. Se muestran las promociones contratadas por el recurso escogido. 4. Quedamos a la espera de que desarrollo FACTPOS nos indique si el resultado es el esperado por ellos.	2	CRM	Baja	09020

Figura 4-20: Plan de Pruebas de Reingeniería Web

4.3.3 Implementación y Ejecución del Caso de Estudio

Nos encontramos ya en la fase de implementación y ejecución de las pruebas diseñadas para el proyecto Reingeniería Web.

Como bajo las condiciones de trabajo que tenemos, el entorno en principio está siempre arriba, no hace falta que realicemos validaciones sobre el entorno y su estado, por tanto el primer paso a dar dentro de esta fase, será la **preparación de recursos** para las pruebas diseñadas, más en concreto, la preparación de números de teléfono que cumplan las condiciones para poder ejecutar las pruebas. Disponemos de un rango de recursos de pruebas que pertenecen al equipo de pruebas y de los cuales hacemos uso desde el equipo de integradas ya sea para nuestras propias pruebas o para proporcionárselos a los usuarios y que los utilicen en las pruebas de aceptación. Asimismo, dentro de dicho rango, cada equipo de pruebas integradas, disponemos de una asignación de recursos, de manera que así el propio equipo de pruebas viendo el número de teléfono o MSISDN como nosotros lo denominamos, en los logs sabemos a qué equipo corresponde la prueba. Cada equipo de pruebas gestiona estos recursos como mejor entienda, y en muchas ocasiones, pueden reutilizarse los recursos sin realizar cambios sobre ellos de unos proyectos a otros. Por supuesto, existen también ocasiones en los que habrá que resetearlos y configurarlos como si fuesen dados de alta en ese momento. Dicho esto, solo queda estudiar cuántos y de qué tipo son los recursos que nos hacen falta para ejecutar el plan de pruebas de Reingeniería Web:

- Para las pruebas de logado: necesitamos un residencial pospago, un residencial prepago y un empresa pospago.
- Para las pruebas de encolado: no especifica el tipo de recursos a utilizar, lo que indica que es indiferente. Así que podremos utilizar uno de los anteriores o utilizar alguno que tengamos de otro proyecto.
- Para las pruebas de New TV: da igual el tipo del que sean los recursos, buscaremos uno de proyectos anteriores que pueda servirnos.

En la figura 4-21 especifico los recursos que vamos a utilizar en la ejecución de las pruebas:

MSISDN	Tipo Contrato	Proyecto	Entorno
111111111	Residencial Pospago	Reingeniería Web	Integración
111111114	Residencial Pospago	Reingeniería Web	Integración
111111116	Residencial Pospago	Reingeniería Web	Integración
111111112	Residencial Prepago	Reingeniería Web	Integración
111111115	Residencial Prepago	Reingeniería Web	Integración
111111113	Empresa Pospago	Reingeniería Web	Integración

Figura 4-21: Recursos o Datos

Además, es importante tener en cuenta que para la realización de un trabajo completo es imprescindible documentar las pruebas ejecutadas. Para ello, al ejecutar una prueba dada habrá que almacenar los resultados obtenidos para poder analizarlos y para en un futuro demostrar el resultado obtenido en esa ejecución de la prueba. En las pruebas de integración, la documentación de las pruebas consiste en almacenar los logs obtenidos al ejecutar una prueba, a los que llamamos **evidencia** y subirlos a Quality Center en el momento en el que se transita el caso de prueba en TestLab al estado Pasado.

El siguiente paso ya sería la ejecución en sí, y para mostrar con el mayor detalle posible la ejecución, vamos a realizarlo en base a un calendario, que por supuesto parte del documento *Planificación* que mostramos en la fase de planificación.

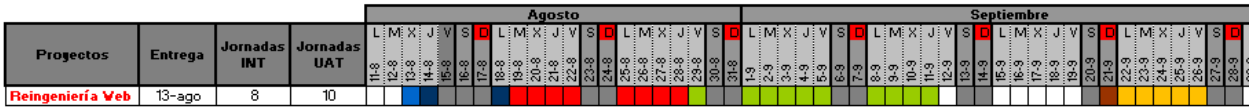


Figura 4-22: Planificación de Reingeniería Web

Lo primero de todo, es imprescindible realizar la entrega de software para poder instalarlo en el entorno de integración y a continuación realizar las pruebas oportunas sobre él. De manera que nos encontramos a la espera de la entrega de las RU: WEB_RU_916 y MDWAE_RU_1468. Hay ocasiones, en las que con la entrega de algunos sistemas, aunque no sean todos los concernientes a un proyecto, pueden comenzarse las pruebas de esa parte, pero, en este caso, tienen que realizarse ambas entregas e instalaciones para poder comenzar las pruebas de Reingeniería Web.

4.3.3.1 Lunes, 18 de agosto

Sufrimos un retraso en la entrega del proyecto de dos días. Debido a la experiencia, la planificación se realiza teniendo en cuenta un día de entrega y dos días para la instalación. Es cierto que muchas entregas son pequeñas y que muchas instalaciones se finalizan empleando unos minutos, pero no es menos cierto que en otras muchas ocasiones tanto las entregas como las instalaciones dan problemas provocando que se emplee en ellas más tiempo de lo necesario. Es por ello, que jugamos con el margen establecido en planificación de tres jornadas completas de trabajo para entregar e instalar y así no asfixiar a las jornadas de pruebas en caso de que se produzcan retrasos.

Por fin se realiza la entrega del software al final de la jornada laboral o lo que es lo mismo, WEB_RU_916 y MDWAE_RU_1468 se han transitado a Resuelta Proveedores en PVCS y ahora están en manos de SCM. El proyecto se queda bloqueado al 100% a la espera de la instalación.

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	Bloqueado	Pendiente Instalación
2	Acceso aplicativo - Residencial Prepago	Bloqueado	Pendiente Instalación
3	Acceso aplicativo - Empresa Pospago	Bloqueado	Pendiente Instalación
4	Acceso aplicativo - KO	Bloqueado	Pendiente Instalación
5	Encolado - Recarga	Bloqueado	Pendiente Instalación
6	Encolado - Alta de Promoción	Bloqueado	Pendiente Instalación
7	Encolado - Cambio de Tarifa	Bloqueado	Pendiente Instalación
8	Encolado - Cambio de Servicios	Bloqueado	Pendiente Instalación
9	Alta Promo New	Bloqueado	Pendiente Instalación
10	Re-Alta Promo New	Bloqueado	Pendiente Instalación
11	Consulta Promo New	Bloqueado	Pendiente Instalación
12	Consulta Promo New con Consumo	Bloqueado	Pendiente Instalación
13	Baja Promo New	Bloqueado	Pendiente Instalación
14	Re-Baja Promo New	Bloqueado	Pendiente Instalación
15	Consulta NO Promo New	Bloqueado	Pendiente Instalación
16	Optimización 09020	Bloqueado	Pendiente Instalación

Figura 4-23: Estado PP en TestLab, lunes 18

4.3.3.2 Martes, 19 de agosto

Comenzamos bien el día porque SCM nos solicita permiso para instalar transitando las entregas a Petición a Integración. Nosotros damos el OK porque nos encontramos en plena ventana de instalación y transitamos MDWAE_RU_1468 y WEB_RU_916 a Aprobada Integración en PVCS para que procedan con la instalación en el entorno de integración. La instalación transcurre sin problemas y en breve las tenemos en estado En Test, que es el estado que da luz verde al comienzo de las pruebas integradas.

Por otro lado, recibimos un mail del equipo de desarrollo de WEB donde nos indican que en la entrega que han realizado finalmente no se ha implementado el software para encolar la operación de recarga, 06001. Como esta información es nueva para nosotros, nos ponemos en contacto con el jefe de proyecto para que nos lo confirme, el cual nos indica que así es, que finalmente la operación 06001 se sacó del alcance funcional del proyecto. Ante este hecho, la consecuencia directa es que debemos anular el caso de prueba 6 que diseñamos para este supuesto. Para ello, desde TestLab en Quality Center, realizamos un cambio de estado en el caso de prueba 6, transitándolo a Cancelado e indicando en el motivo "sale del alcance funcional". Esto se traduce en que tendremos un caso menos que ejecutar.

A primera hora de la tarde, todo está listo para comenzar con las pruebas, al final comenzaremos según lo previsto en la planificación. Por fin tenemos los casos en estado Pendiente y teniendo en cuenta la prioridad de ejecución marcada, comenzamos ejecutando las pruebas de logado en la web correspondientes a la operación 12010.

Ejecuto la prueba 1, cuya única finalidad es el logado del cliente en WEB.Utilizo un recurso residencial pospago de los que tenemos preparados. Los logs con el resultado se muestran en el anexo 7-1. Para nuestra sorpresa, con la primera prueba, ya estamos obteniendo un error. Por pantalla indica que el cliente no existe y en los logs comprobamos que el error indica que el MSISDN no existe. Analizando más en profundidad la traza obtenida para intentar buscar una explicación, vemos que el sistema que devuelve el error es FACTPOS.

La operación de logado 12010 comienza en el sistema WEB el cual pregunta a FACTPOS en primer lugar si el cliente es prepago o pospago, y lo hace a través de la operación 03018. Y justo a continuación debería publicarse una 03045 para preguntar a FACTPOS los datos de cliente en caso de que se trate de un pospago o una 10004 para preguntar a FACTPRE los datos de cliente si fuese prepago. En el log publicado estamos viendo que al responder FACTPOS con error en

la03018 ya no continua el flujo, por ello, nos paramos a estudiar más de cerca la petición de datos que se hace a través de la consulta del tipo de cliente 03018 a FACTPOS. Y ahí está, de inmediato se observa que el campo MSISDN se está enviando sin valor a FACTPOS, y claro, FACTPOS indica que no existe ese cliente. La cuestión es que en la petición del logado con la 12010, WEB sí está enviando el MSISDN, por tanto, es el sistema MIDDLEWARE quien se está olvidando de rellenar el número de teléfono para enviárselo a FACTPOS.

Al ser MIDDLEWARE quien está cometiendo el error, en principio podríamos afirmar que la incidencia se encuentra en MIDDLEWARE y que el fallo se va a cometer para todos los tipos de recursos, pues MIDDLEWARE no diferencia el tipo de cliente, simplemente debe enviar los datos que recibe.

Para comprobar esta teoría, el siguiente paso es ejecutar los otros dos casos de logado para asegurarnos. Así que **lanzo la prueba 2 y la prueba 3**. Las pruebas confirman las sospechas tenidas, de manera que se está produciendo un error en el sistema MIDDLEWARE que está recibiendo el MSISDN informado como campo de entrada en la petición de la 12010 enviada desde WEB pero lo está enviando vacío a FACTPOS, el cual, por supuesto, devuelve un error indicando que no existe el cliente.

Llegado este punto, procedemos a:

- Abro una incidencia TI en PVCS, al sistema MIDDLEWARE para que solucionen el problema. Se trata de la MDWAE_TI_2130, que se queda en estado Análisis Desarrollo.
- Redacto un mail a desarrollo MIDDLEWARE con la plantilla de errores para informarles del fallo localizado con el mayor detalle posible sobre lo sucedido, además de indicarles cual es la TI abierta y enviarles los logs obtenidos.

PROYECTO	Reingeniería WEB		
Sistema Origen	WEB		
Sistema Destino	FACTPOS		
Recurso Utilizado	111111111, pero podría ser cualquiera porque está fallando para todos		
Descripción del recurso (NIF, Tarifas, etc)	Residencial pospago		
Descripción Caso Prueba (Prueba Realizada)	Logado en WEB a través del nuevo servicio de logado 12010	Severidad	Grave, Bloqueante
Resultado obtenido (Problema encontrado)	No se loga en WEB porque MIDDLEWARE está enviando vacío el campo MSISDN, el cual sí se está enviando informado desde WEB en la petición de la 12010		
Entorno prueba (INT / UAT)	Integración		

Figura 4-24: Plantilla Incidencias - MDWAE_TI_2130

- Actualizo el estado de la ejecución de los casos en TestLab de Quality Center, reflejando para el caso 1 el fallo, transitándolo a estado Fallado. Y los casos 2 y 3 los transito a Bloqueado, añadiendo la incidencia TI como motivo de bloqueo. También podrían transitarse a Fallado porque realmente los he ejecutado, pero como el motivo de bloqueo se localizó en la ejecución del caso 1, es el que pongo como fallado y el resto bloqueados.
- Por último, al ser esta nueva forma de logado la única posible de acceder al sistema WEB, procedo a bloquear el resto de casos de Reingeniería WEB por el mismo motivo, excepto el caso de optimización de consultas que se lanza desde el sistema CRM. Asimismo, y como información adicional, bloqueo todos los casos de otros proyectos que también llevo en estos momentos y desde los que ejecutaría desde el sistema WEB como origen, por el mismo motivo. E informo a mis compañeros del resto de equipos que componen las pruebas integradas para que hagan lo mismo con sus proyectos hasta que se resuelva la MDWAE_TI_2130, dado que es bloqueante para todo el sistema WEB.

Cerramos la jornada laboral con el siguiente estado de ejecución de Reingeniería WEB porque como es de esperar, la incidencia localizada necesita su tiempo para corregirse:

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	Bloqueado	No llega a logarse MDWAE_TI_2130
2	Acceso aplicativo - Residencial Prepago	Bloqueado	No llega a logarse MDWAE_TI_2130
3	Acceso aplicativo - Empresa Pospago	Failed	No llega a logarse MDWAE_TI_2130
4	Acceso aplicativo - KO	Bloqueado	No llega a logarse MDWAE_TI_2130
5	Encolado - Recarga	Cancelado	sale del alcance funcional
6	Encolado - Alta de Promoción	Bloqueado	No llega a logarse MDWAE_TI_2130
7	Encolado - Cambio de Tarifa	Bloqueado	No llega a logarse MDWAE_TI_2130
8	Encolado - Cambio de Servicios	Bloqueado	No llega a logarse MDWAE_TI_2130
9	Alta Promo New	Bloqueado	No llega a logarse MDWAE_TI_2130
10	Re-Alta Promo New	Bloqueado	No llega a logarse MDWAE_TI_2130
11	Consulta Promo New	Bloqueado	No llega a logarse MDWAE_TI_2130
12	Consulta Promo New con Consumo	Bloqueado	No llega a logarse MDWAE_TI_2130
13	Baja Promo New	Bloqueado	No llega a logarse MDWAE_TI_2130
14	Re-Baja Promo New	Bloqueado	No llega a logarse MDWAE_TI_2130
15	Consulta NO Promo New	Bloqueado	No llega a logarse MDWAE_TI_2130
16	Optimización 09020	Pendiente	

Figura 4-25: Estado TestLab martes 19

4.3.3.3 Miércoles, 20 de agosto

Estamos en plena release y cada día se va trabajando con mayor celeridad para llegar al objetivo de finalizar todos los proyectos para su subida a producción. Y el esfuerzo no viene tan solo del equipo de pruebas de integración, si no que se nota que desarrollo también se encuentra a pleno rendimiento. Es por ello que a primera hora de la mañana desarrollo MIDDLEWARE realiza la entrega de MDWAE_TI_2130, lo sabemos porque se ha transitado al estado Resuelta Proveedores en PVCS, y ahora está en manos de SCM quien debe aprobar la entrega para solicitarnos permiso para su instalación. Aprobamos su instalación y cuando queremos darnos cuenta, la tenemos en Test, es el momento de volver a ejecutar las pruebas de logado para ver qué sucede.

De nuevo, comenzamos por el caso de prueba 1, el cual consiste en el logado en WEB para un residencial pospago, **reejecuto el caso 1**. En esta ocasión, sí consigue entrar a la WEB. En los logs comprobamos que todo ha ido bien. Puede verse en el anexo 7-2. Es importante, en esta ocasión fijarse en el campo MSISDN de la petición de la 03018 para asegurarse de que va informado, y por tanto, poder resolver la MDWAE_TI_2130, y después, observar la respuesta en la que deben venir los datos del cliente que se muestran en la pantalla de la WEB, la cual por motivos obvios no puedo mostraros, pero en los logs podemos ver todo el detalle.

Al observar que el campo MSISDN ya se envía informado:

- Resuelvo la MDWAE_TI_2130 en PVCS transitando la incidencia a Resuelta Integración e informo a desarrollo MDWAE y a mis compañeros de integración.
- Compruebo que el resultado mostrado por pantalla coincide con lo devuelto en la traza. Y verifico que los datos son los mismos, por tanto, están bien transcritos a la ventana WEB.
- Paso el caso de prueba en TestLab de Quality Center, es decir, lo pongo en estado Pasado y desbloqueo el resto de casos de todos los proyectos transitándolos a Pendiente porque ya funciona el nuevo logado en WEB.
- Almaceno la evidencia obtenida (los logs) con la ejecución del caso y la subo asociada al caso en Quality Center.

En principio los otros dos casos de logado funcionarán sin problema, **reejecuto los casos 2 y 3**. Y como es de esperar, funcionan tal y como se pretende de ellos. Almaceno los logs y los subo a Quality Center al transitar los casos al estado Pasado. En los anexos7-3 y 7-4, añado los logs de los casos 2 y 3, respectivamente.

Para terminar con el primer bloque funcional y que además se trata del bloque con mayor prioridad de ejecución, procedo a **ejecutar** la prueba del KO del logado, el **caso de prueba 4**, en el que introduciré un MSISDN que no esté contenido en las bases de datos de integración. El sistema vuelve a comportarse tal y como se espera de él y se produce un log muy parecido al que se obtenía con la MDWAE_TI_2130 solo que ahora el MSISDN sí se envía informado, pero es FACTPOS quien al no contener registrado en base de datos el MSISDN por el que se le está preguntando, devuelve el mismo error indicando que no existe el cliente por el que se le pregunta. Es justo aquí donde se encuentra respuesta a la importancia de los logs, porque aquí se trata de un error lógico, pero con el mismo tipo de error, antes hemos averiguado que realmente se estaba produciendo un error en el traspaso de datos en MIDDLEWARE. Vamos, buenas noticias, tenemos las pruebas de logado OK.

Continuamos con la ejecución de los casos de prueba siguiendo el orden de prioridad marcado desde los comienzos de Reingeniería Web. Por tanto, nos ponemos de lleno con los casos del encolado de las operaciones que finalicen en error. El funcionamiento esperado para los casos de encolado es que se lance la operación de forma manual, se produzca un fallo de tráfico, como puede ser un timeout por falta de respuesta de un sistema final y pasados unos minutos, la operación se relance de nuevo, esta vez automáticamente, para intentar conseguir conectar con el sistema final. Esto se podrá repetir hasta en un máximo de dos ocasiones por cada operación que se encole, con el fin claro de no colapsar el sistema con un número desmesurado de intentos.

La dificultad de estos casos de encolado reside en que no podemos ejecutar las pruebas continuamente esperando a que se produzca el fallo en MIDDLEWARE de manera fortuita, por tanto debemos ser nosotros quienes provoquemos el timeout. Para ello lo que debemos hacer es que el sistema destino no conteste y esto lo conseguimos si tiramos el adaptador pesado del sistema destino para evitar que le llegue la petición al sistema destino y por tanto, evitamos que vaya a contestar, provocando consecuentemente un timeout. Y nada más producirse el primer reintento, levantaremos el adaptador para que el sistema destino conteste al segundo reintento.

Nos vamos al plan de pruebas y recordamos que el caso número 5, el del encolado para la operación de recarga finalmente se canceló por salirse del alcance funcional, así que nos ponemos manos a la obra con el **caso de prueba número 6** que consiste en el encolado para una operación 09030 que es un alta/baja/modificación de una promoción. Esta operación va a FACTPOS, así que nos disponemos a tirar su adaptador pesado de promociones para provocar el timeout. A continuación, entramos en WEB con el recurso residencial pospago que ya hemos utilizado en pruebas anteriores e intentamos dar de altauna promoción cualquiera de las disponibles. Vemos en los logs que la operación se publica y que se produce un timeout, esperamos a la publicación del reintento pero no sucede. Hay algún problema con el encolado o con el reenvío de la petición.

Es necesario asegurarse si el problema se está produciendo solo para esta operación o para el resto de operaciones de encolado, por lo que rápidamente **ejecuto los casos 7 y 8** para los cuales sucede exactamente lo mismo. Se publica, se produce el timeout pero no se publican automáticamente los reintentos. Añado uno de los ejemplos en el anexo 7-5. Acabo de descubrir otra incidencia, en este caso es bloqueante para la funcionalidad, así que procedo con las siguientes tareas:

- Abro una TI en PVCS, en este caso a WEB porque es quien debería reenviar las peticiones lanzadas al producirse un timeout y no lo está haciendo. Se trata de la WEB_TI_1770.
- Se lo comunico a desarrollo vía email, rellenando la plantilla de incidencias.

PROYECTO	Reingeniería WEB		
Sistema Origen	WEB		
Sistema Destino	FACTPOS		
Recurso Utilizado	111111111		
Descripción del recurso (NIF, Tarifas, etc)	Residencial pospago		
Descripción Caso Prueba (Prueba Realizada)	Encolado de operaciones	Severidad	Bloqueante para la funcionalidad
Resultado obtenido (Problema encontrado)	Se publica una operación, y al producirse un timeout no se está reenviando la petición. O bien hay un fallo en el encolado o en el reenvío. Está fallando para: 09030, 26032 y 26033.		
Entorno prueba (INT / UAT)	Integración		

Figura 4-26: Plantilla Incidencias - WEB_TI_1770

- Actualizo el estado del caso 6 a Fallado y los casos 7 y 8 a Bloqueado. Todos ellos con el motivo WEB_TI_1770: No se realiza el encolado.

Queda finalizada la segunda jornada de pruebas, y el estado de los casos es el siguiente:

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	OK	
2	Acceso aplicativo - Residencial Prepago	OK	
3	Acceso aplicativo - Empresa Pospago	OK	
4	Acceso aplicativo - KO	OK	
5	Encolado - Recarga	Cancelado	sale del alcance funcional
6	Encolado - Alta de Promoción	Failed	No realiza encolado WEB_TI_1770
7	Encolado - Cambio de Tarifa	Bloqueado	No realiza encolado WEB_TI_1770
8	Encolado - Cambio de Servicios	Bloqueado	No realiza encolado WEB_TI_1770
9	Alta Promo New	Pendiente	
10	Re-Alta Promo New	Pendiente	
11	Consulta Promo New	Pendiente	
12	Consulta Promo New con Consumo	Pendiente	
13	Baja Promo New	Pendiente	
14	Re-Baja Promo New	Pendiente	
15	Consulta NO Promo New	Pendiente	
16	Optimización 09020	Pendiente	

Figura 4-27: Estado TestLab, miércoles 20

4.3.3.4 Jueves, 21 de agosto

Nada más comenzar el tercer día de las pruebas de integración, comienzan los problemas. El sistema WEB está caído. Nos ponemos en contacto con el equipo de Gestión de Entornos, el cual nos indica que ya lo están tratando y que no pinta muy bien. Para formalizar el problema de entorno localizado procedo con dos tareas:

- Abro una incidencia jtrac de entornos, se trata de la INCE_0568, la cual asigno al equipo de Gestión de Entornos.
- Bloqueo los casos que faltan con origen WEB del proyecto Reingeniería Web, que son todos excepto el caso 16, con el motivo INCE_0568.
- Comunico a mis compañeros de integradas el problema y la INCE mediante la cual se realizará el seguimiento del estado del problema.

Nos quedamos sin más remedio, a la espera de la resolución de la INCE_0568 para poder volver a entrar al sistema WEB. Mientras tanto, nos queda una **prueba**, la **16**, cuyo origen es CRM, por lo que aunque su prioridad es baja, mientras solucionan el problema, ejecutaremos la prueba que nos ha solicitado desarrollo. Necesitamos un recurso con promociones activas y sabemos que el residencial pospago que venimos utilizando tiene más de una activa, así que entramos en el sistema CRM, que es el maestro de clientes y lanzamos una consulta de promociones activas 09020. Recogemos los logs en SecureCRT y enviamos lo obtenido a desarrollo, que por lo que nos han comentado quieren validar una serie de tiempos de respuesta internos de FACTPOS. El caso de prueba se queda a Pendiente, esperando la respuesta que nos de desarrollo.

Y para nuestra mala suerte, el bloqueo de acceso a WEB se mantiene durante el resto del día, así que cerramos la jornada con el siguiente estado de casos en TestLab de Quality Center:

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	OK	
2	Acceso aplicativo - Residencial Prepago	OK	
3	Acceso aplicativo - Empresa Pospago	OK	
4	Acceso aplicativo - KO	OK	
5	Encolado - Recarga	Cancelado	sale del alcance funcional
6	Encolado - Alta de Promoción	Failed	No realiza encolado WEB_TI_1770
7	Encolado - Cambio de Tarifa	Bloqueado	No realiza encolado WEB_TI_1770
8	Encolado - Cambio de Servicios	Bloqueado	No realiza encolado WEB_TI_1770
9	Alta Promo New	Bloqueado	Problemas Acceso WEB INCE_0568
10	Re-Alta Promo New	Bloqueado	Problemas Acceso WEB INCE_0568
11	Consulta Promo New	Bloqueado	Problemas Acceso WEB INCE_0568
12	Consulta Promo New con Consumo	Bloqueado	Problemas Acceso WEB INCE_0568
13	Baja Promo New	Bloqueado	Problemas Acceso WEB INCE_0568
14	Re-Baja Promo New	Bloqueado	Problemas Acceso WEB INCE_0568
15	Consulta NO Promo New	Bloqueado	Problemas Acceso WEB INCE_0568
16	Optimización 09020	Pendiente	

Figura 4-28: Estado TestLab, jueves 21

4.3.3.5 Viernes, 22 de agosto

A lo largo de la mañana por fin se resuelve la incidencia que bloquea el acceso a WEB, la INCE_0568. Revisamos la jtrac de incidencias y el equipo de Gestión de Entornos ya ha transitado la incidencia al estado Tratada. Y como se accede sin problemas, transitamos dicha incidencia al estado Resuelta y finalmente a Cerrado. Podemos continuar con las pruebas.

Mientras realizan la entrega con la resolución al problema del encolado reflejado en la WEB_TI_1770, continuamos las pruebas integradas con la ejecución de los casos de la nueva promoción New TV. Comenzamos con el caso del alta de promoción, el **caso de prueba 9**. Es indiferente el tipo de recurso porque la promoción es accesible para cualquier cliente, por lo que escogemos de nuestro rango un recurso residencial prepago que no tenga activa New TV, nos logamos y en la página de promociones y contratamos la promoción New TV. En solo unos instantes, por pantalla nos indica que la contratación ha ido OK, así que me dispongo a localizar los logs en SecureCRT, donde compruebo que efectivamente, la operación ha ido perfectamente. El detalle de los logs lo comento en el anexo 7-6. Actualizo el Quality Center con el caso 9 a Pasado y subo la evidencia.

El siguiente caso a ejecutar, es la consulta de la promoción que se acaba de dar de alta. Se trata del **caso de prueba 11** y consiste en entrar en la WEB con el recurso que acabamos de utilizar y lanzar una consulta de promociones contratadas para asegurarse de que muestra a New TV como una promoción más del cliente. Tal y como se detalla en el anexo 7-7, la promoción New TV se muestra como una de las contratadas para el recurso en cuestión en los logs, además, por pantalla, se visualiza como activa. Esto se traduce en que tenemos otro caso OK. Procedo a transitar el estado del caso 11 en Quality Center y a subir la evidencia correspondiente.

Observamos el plan de pruebas, y nos ponemos a ejecutar el **caso de prueba 10**, que corresponde con la Re-Alta de la promoción New TV. Es un caso de prueba en el que se espera que se produzca un error, y que dado que la promoción ya está activa, no permita activarla de nuevo. Por consiguiente, empleamos el recurso de las dos últimas pruebas, entramos en WEB, localizamos la promoción New TV y lanzamos el alta, como si el recurso no la tuviese activa. Por suerte, el resultado es satisfactorio, el sistema maestro de la promoción devuelve un error indicando que la promoción ya está activa para ese recurso. El mensaje de error se visualiza perfectamente por pantalla y además, se puede ver el detalle en los logs recogidos de SecureCRT

y que muestro en el anexo 7-8, el cual subo como evidencia a TestLab. La consecuencia directa de este resultado, es otro caso en estado Pasado en Quality Center.

Para continuar con el siguiente **caso de prueba**, el **12**, hay que realizar una consulta exactamente igual que para el caso 11. La diferencia reside en que para el caso 12 hay que mostrar los contadores con consumo realizado. Para ello, no hay que olvidar que nos encontramos en el entorno de integración, que se trata de un entorno de pruebas y que por tanto, no se puede realizar consumo real, en este caso más concreto de minutos de televisión vistos. Este tipo de pruebas las realizamos modificando los valores de los consumos a mano sobre la base de datos donde se almacenan. Como el maestro de la promoción es FACTPRE los valores del consumo están incluidos en su base de datos y es donde modifico a mano el valor del bono consumido a, por ejemplo, 50 minutos ya vistos de televisión, y los minutos disponibles a 50 minutos, dado que la promoción consta de 100 minutos. Una vez modificado los valores, lanzo la consulta y valido que efectivamente el valor mostrado por pantalla es el esperado, es decir, que se han consumido 50 minutos de la promoción New TV. Además, se puede comprobar dicho valor, en el log que añado en el anexo 7-9. A continuación, transito el caso de prueba 12 a Pasado y subo la evidencia con los logs obtenidos.

Parece que los casos de la promoción están funcionando a la perfección, de manera que continuamos con el caso de la baja de promoción New TV, el **caso número 13**. Para ello, se necesita el mismo recurso residencial prepago que estamos utilizando a lo largo de las pruebas de New TV, que además en estos instantes tiene la promoción activa, que es el requisito indispensable para poder tramitarla baja. Ejecuto el caso, y una vez más, se obtiene un respuesta satisfactoria. La promoción se da de baja y se notifica al cliente por pantalla, además se puede comprobar el resultado en los logs que muestro en el anexo 7-10. Una vez más, en TestLab de Quality Center procedo a transitar el caso al estado Pasado y a subir la evidencia conseguida.

Por último, durante la jornada de pruebas de hoy, paso a ejecutar la Re-Baja de la promoción, que corresponde con el **caso de prueba 14**. Podemos utilizar el recurso al que acabamos de dar de baja la promoción New TV para intentar darla de baja de nuevo y ver qué contesta el sistema maestro. Dicho y hecho, ejecuto la prueba y el resultado es el correcto, se muestra un mensaje de error por pantalla indicando que el cliente no tiene dada de alta la promoción New TV y en los logs que podemos ver en el anexo 7-11se observa como FACTPRE no ha podido realizar la baja

de una promoción para un recurso que no tiene dicha promoción activa. Por tanto, todo OK, subo la evidencia y transito el caso de prueba 14 a Pasado.

Podemos concluir la jornada de trabajo como fructífera, y el estado en el que quedan los casos en TestLab es el siguiente:

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	OK	
2	Acceso aplicativo - Residencial Prepago	OK	
3	Acceso aplicativo - Empresa Pospago	OK	
4	Acceso aplicativo - KO	OK	
5	Encolado - Recarga	Cancelado	sale del alcance funcional
6	Encolado - Alta de Promoción	Failed	No realiza encolado WEB_TI_1770
7	Encolado - Cambio de Tarifa	Bloqueado	No realiza encolado WEB_TI_1770
8	Encolado - Cambio de Servicios	Bloqueado	No realiza encolado WEB_TI_1770
9	Alta Promo New	OK	
10	Re-Alta Promo New	OK	
11	Consulta Promo New	OK	
12	Consulta Promo New con Consumo	OK	
13	Baja Promo New	OK	
14	Re-Baja Promo New	OK	
15	Consulta NO Promo New	Pendiente	
16	Optimización 09020	Pendiente	

Figura 4-29: Estado TestLab, viernes 22

4.3.3.6 Lunes, 25 de agosto

Comenzamos el quinto día de pruebas integradas con un buen escenario, ya que tenemos un caso para ejecutar, otro para el que estamos esperando respuesta de desarrollo y tres casos bloqueados por la incidencia WEB_TI_1770 que por lo que nos han informado, tienen intención de entregar a lo largo del día.

Para empezar con la ejecución, escogemos el **caso de prueba 15**. Se trata de una prueba de regresión, consiste simplemente en lanzar una consulta de promociones en la que no debe aparecer como contratada la nueva promoción New TV y el objetivo de su lanzamiento es

asegurarse de que el funcionamiento de dicha operación es el de siempre. Aunque no se haya producido una modificación en la funcionalidad de la 09020 como tal, se trata de una operación que se utiliza continuamente y está bien realizar una prueba a modo de regresión para asegurar que su funcionamiento es el de siempre.

La ejecuto con el recurso que hemos estado utilizando para las pruebas de New TV, ya que ahora no tiene activa la promoción. Lanzo la consulta desde WEB y como es de esperar, funciona a la perfección. No considero relevante añadir este log al anexo porque en base es lo mismo que la consulta para la promoción New TV, solo que en esta ocasión, no figura la promoción New TV entre las promociones activas para el recurso. Damos el caso de regresión por OK transitando el caso número 15 al estado Pasado en Quality Center y subiendo la evidencia obtenida.

Lo siguiente que sucede a lo largo de la mañana es que desarrollo FACTPOS nos contesta sobre la operación de consulta 09020 que les remitimos y nos indican que la demos por OK porque ha funcionado como esperaban dentro de su sistema. Actuamos de inmediato transitando el **caso número 16**a estado Pasado, por supuesto, adjuntando la evidencia que les enviamos.

Por la tarde desarrollo modifica el estado de PVCS de la entrega WEB_TI_1770, por fin realizan la entrega de la incidencia. La entrega, revisión por parte de SCM y posterior preparación para la instalación se llevan lo que queda de jornada laboral. Transitamos la entrega a Aprobada Integración al final de la jornada, mañana probaremos si funciona la corrección.

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	OK	
2	Acceso aplicativo - Residencial Prepago	OK	
3	Acceso aplicativo - Empresa Pospago	OK	
4	Acceso aplicativo - KO	OK	
5	Encolado - Recarga	Cancelado	sale del alcance funcional
6	Encolado - Alta de Promoción	Failed	No realiza encolado WEB_TI_1770
7	Encolado - Cambio de Tarifa	Bloqueado	No realiza encolado WEB_TI_1770
8	Encolado - Cambio de Servicios	Bloqueado	No realiza encolado WEB_TI_1770
9	Alta Promo New	OK	
10	Re-Alta Promo New	OK	
11	Consulta Promo New	OK	
12	Consulta Promo New con Consumo	OK	
13	Baja Promo New	OK	
14	Re-Baja Promo New	OK	
15	Consulta NO Promo New	OK	
16	Optimización 09020	OK	

Figura 4-30: Estado TestLab, lunes 25

4.3.3.7 Martes, 26 de agosto

Como es de esperar, nada más llegar nos encontramos con la WEB_TI_1770 en estado En Test en PVCS, así que procedemos a probar si ya funcionan los encolados, que por otro lado, es la única funcionalidad que nos falta para finalizar las pruebas de integración del proyecto.

Para probar la TI, hay que utilizar el mismo recurso que utilizamos y ejecutar la misma prueba, **el caso número 6**.Utilizaremos el residencial pospago con el que obtuvimos el error y ejecutaremos un alta de una promoción cualquiera desde WEB, teniendo en cuenta que tiraremos el adaptador pesado de promociones del sistema FACTPOS para provocar el timeout esperado y que así se encole la petición en WEB y la reenvíe pasados cinco minutos. En esta ocasión, sí funciona. Incluyo el log completo que obtengo de SecureCRT en el anexo 7-12. Tras producirse el timeout, a los cinco minutos se reenvía la petición, y al seguir el adaptador caído, la reenvía una segunda vez. Por tanto:

- En PVCS resuelvo la WEB_TI_1770 transitando su estado a Resuelta Integración. Y se lo comunico al equipo de desarrollo vía mail.

- En TestLab de Quality Center, transito el caso 6 al estado Pasado.
- Subo la evidencia adjuntada al caso de prueba.

Vamos a por la segunda prueba del encolado, el **caso número 7** el cual se refiere al encolado para la operación de un cambio de tarifa. Da igual el tipo de recurso y las tarifas entre las que se realice el cambio, en esta ocasión escojo un residencial pospago. Primero tiro el adaptador pesado de servicios de FACTPOS para que no le llegue ninguna petición. A continuación, entro a WEB con dicho recurso, voy a la página de servicios de cliente y realizo un cambio a una tarifa cualquiera. Se publica la petición, se produce el timeout y a los cinco minutos vuelve a publicarse la operación 26032 para el cambio de tarifa que se había encolado, está funcionando como se espera. A continuación, levanto el adaptador pesado de servicios de FACTPOS y se publica la segunda petición de reenvío para el cambio de tarifa, que en esta ocasión, sí llega hasta el sistema destino, el cual contesta. Añado los logs de esta prueba en el anexo 7-13. Tenemos el penúltimo OK del proyecto. Paso el caso de prueba 7 en Quality Center y subo la evidencia.

Para dar por finalizadas las pruebas integradas, nos ponemos manos a la obra con el último caso de prueba a ejecutar, que es el último caso de la funcionalidad del encolado, el **caso número 8**.Esperamos, que al igual que las otras dos operaciones funcione a la perfección, solo falta comprobarlo.

Consiste en realizar un cambio en los servicios contratados, por ejemplo, desactivando el servicio de roaming datos. El tipo de recurso vuelve a ser indiferente, así que en este caso escojo otro residencial pospago. Al igual que en la prueba anterior, primero tiro el adaptador pesado de servicios de FACTPOS y después lanzo el cambio de servicios 26033 desde WEB. La diferencia entre esta operación y la del caso de prueba anterior, es que la 26033 es una operación punto a punto, es decir, solo hay un destino. Por lo que MIDDLEWARE no comunica literalmente el timeout en los logs, de manera que en este caso el reenvío de la operación encolada se debe producir seis minutos después de que el sistema destino no conteste a la petición que se le envía. Dicho esto, ejecuto la prueba y el resultado es satisfactorio, lo detallo en el anexo 7-14. E inmediatamente transito el caso de prueba 8 a Pasado y adjunto la evidencia obtenida en SecureCRT. Hemos finalizado las pruebas de integración!

Num. Caso	Descripción Caso	Estado	Motivo Bloqueo
1	Acceso aplicativo - Residencial Pospago	OK	
2	Acceso aplicativo - Residencial Prepago	OK	
3	Acceso aplicativo - Empresa Pospago	OK	
4	Acceso aplicativo - KO	OK	
5	Encolado - Recarga	Cancelado	
6	Encolado - Alta de Promoción	OK	
7	Encolado - Cambio de Tarifa	OK	
8	Encolado - Cambio de Servicios	OK	
9	Alta Promo New	OK	
10	Re-Alta Promo New	OK	
11	Consulta Promo New	OK	
12	Consulta Promo New con Consumo	OK	
13	Baja Promo New	OK	
14	Re-Baja Promo New	OK	
15	Consulta NO Promo New	OK	
16	Optimización 09020	OK	

Figura 4-31: Estado TestLab, martes 26

Por la tarde nos ponemos a avanzar tareas de soporte a los usuarios, como es la preparación de los recursos que nos han solicitado que les proporcionemos en el entorno de aceptación para ejecutar sus pruebas sobre Reingeniería Web.

4.3.3.8 Pruebas de aceptación

Para nuestra tranquilidad como equipo de integración, hemos finalizado las pruebas de integración del proyecto Reingeniería Web antes de lo previsto. Y dado que faltan por realizar las pruebas de aceptación antes de que el proyecto pueda subir a producción y el proyecto está asociado a una campaña publicitaria, sin dudarlo ni un instante, comunicamos al equipo de instalaciones SCM que vayan preparando la instalación de las entregas del proyecto en el entorno de aceptación. Como el proyecto ya ha sufrido las pruebas de integración, en el entorno del usuario se debe instalar las RU (MDWAE_RU_1468 y WEB_RU_916) además de las TI localizadas y resueltas durante las pruebas. De esta forma, el usuario ejecuta sus pruebas en un

entorno lo más depurado posible esperando que los errores que puedan localizar sobre Reingeniería Web, o cualquier otro proyecto, sean de bajo impacto.

Minutos más tarde, las entregas ya se encuentran en estado Petición UAT en PVCS y las transitamos a Aprobado UAT para que se instalen. Una vez que se encuentran en estado En Test procedemos a la liberación del proyecto al usuario entregando los recursos solicitados y dando permiso al inicio de la ejecución de su plan de pruebas sobre el proyecto Reingeniería Web en el entorno de aceptación.

A lo largo de las pruebas de UAT, los usuarios no necesitan mucho soporte por parte del equipo de integración, dado que no hay necesidad de limpiar los recursos y con los que nos solicitaron, son suficientes. Además, no localizan errores en la funcionalidad como tal porque queda totalmente depurada con las pruebas de integración. Sin embargo, sí localizan un error de un literal mostrado en la Web nada más logarse. Se trata de un error en la edición de los textos que el cliente debe visualizar al entrar en Web y es una prueba que corresponde con el requisito 1 para el que el equipo de integradas no realizamos pruebas al no tratarse de cambios funcionales.

El usuario nos abre un defecto para especificar el error localizado, es el defecto DF-0162 que se queda en estado Abierto para que podamos tratarlo. De inmediato lo transito a En-Análisis, y al comprobar que efectivamente se trata de un error en el texto esperado:

- Abro una SM de severidad baja, es la WEB_SM_1153 y se lo comunico a desarrollo Web vía mail.
- Transito el defecto DF-0162 al estado En-Resolución, estado en el cual permanecerá hasta que se realice la entrega de la SM.
- Informo al usuario de la apertura de la WEB_SM_1153 para que pueda bloquear los casos a los que le afecte la incidencia.

La siguiente comunicación que tenemos con el usuario se produce al día siguiente y es para indicarle de que se ha realizado entrega de la SM.

- Transito la WEB_SM_1153 a Aprobado UAT para que SCM proceda con la instalación, la cual queda instalada en unos minutos.
- Una vez finalizada la instalación, transito el defecto en Quality Center al estado Asignado y les comunico al usuario que pueden probar la incidencia.

Al día siguiente el usuario transita el defecto DF-0162 al estado Resuelto, por lo que procedo a resolver la WEB_SM_1153 transitándola al estado Resuelta UAT en PVCS. Además, cierro el defecto.

Tan solo un día más tarde, el usuario finaliza las pruebas de aceptación sobre el proyecto Reingeniería Web antes de lo esperado, lo cual es una buena noticia porque significa que el proyecto está listo para cumplir con el PAP establecido.

4.3.4 Evaluación del Caso de Estudio

La fase de evaluación, en primer lugar consiste en evaluar los criterios de finalización de las pruebas de integración. El criterio más habitual es el que evalúa el porcentaje de casos de prueba ejecutados OK, y en el caso concreto de Reingeniería Web es el más lógico, pues se ha conseguido ejecutar con resultado satisfactorio todos los casos de prueba existentes y la finalización de las pruebas es evidente.

Por otro lado, la finalización de la fase de las pruebas incluye una realización de una salida, una documentación a modo de output en la que se registra el histórico de cómo han ido las pruebas. Además, ya que desde nuestro equipo de integradas se da soporte y se realiza un seguimiento continuo sobre el estado de las pruebas de aceptación, una vez finalizadas las mismas los documentos que se realizan contienen la información tanto de las pruebas integradas como de las pruebas de aceptación, eso sí, en documentos diferentes. La documentación se obtiene de forma automática y los procesos obtienen la información de PVCS y de Quality Center. Los documentos que se crean son:

- **Plan de Pruebas (PP)**: Es la información concerniente al plan de pruebas sin ejecución. Contiene información detallada sobre los casos de prueba diseñados con sus descripciones, pasos a seguir, precondiciones, entradas, salidas esperadas, tipo de prioridad de ejecución... Asimismo, se incluye la trazabilidad de los requisitos con sus respectivos casos de prueba.

- **Informe de Pruebas (IP)**: Incluye la información del plan de pruebas tras su ejecución. En él se registran las entregas asociadas al proyecto, la descripción de los casos de prueba con su trazabilidad con requisitos, el estado final en el que se encuentra el caso de prueba (si se ha anulado, ejecutado, si está OK, fallado o bloqueado). También se registran las incidencias abiertas junto con el caso al que están asociadas y su estado al cerrar el proyecto.

- **Informe Ejecutivo de Pruebas (IE)**: Contiene información sobre los porcentajes de casos ejecutados, ordenados por prioridad de lanzamiento y sobre las incidencias totales abiertas, ordenadas por severidad.

4.3.5 Cierre del Caso de Estudio

Esta fase es realmente la última del proceso de pruebas, ya que las otras dos fases transcurren en paralelo. En la fase de cierre se realizan las tareas concernientes a crear el punto y final sobre un proyecto dado. Las tareas que realizamos son las siguientes:

- Es imprescindible cerrar todas las incidencias que se hayan localizado durante las pruebas. O lo que es lo mismo, para que el proyecto pueda subir a producción, las RU, TI, y SM tienen que estar en estado Resuelta. En el caso concreto de Reingeniería Web:
    - Se transita a Resuelta Integración: MDWAE_RU_1468, WEB_RU_916, MDWAE_TI_2130, WEB_TI_1762 y WEB_TI_1770.
    - Se transita a Resuelta UAT: WEB_SM_1153.
- Limpieza de los recursos utilizados: los recursos de los que disponemos para la ejecución de las pruebas son finitos y solemos cuidar su estado al finalizar un proyecto porque podrán ser utilizados en futuras pruebas. Por ejemplo, desactivando las promociones que se le hayan dado de alta, o directamente creando un recurso como si fuese un cliente nuevo pero bajo la misma numeración de teléfono.
- En nuestro caso en concreto no realizamos limpieza de entornos, es algo bastante habitual en otros sistemas pero a nosotros nos interesa que el entorno de integración se

parezca lo máximo posible al entorno de producción en entregas instaladas. Por ello, el entorno no se limpia como tal, se mantiene con todas las instalaciones realizadas.

- Como lecciones aprendidas del proyecto Reingeniería Web podríamos destacar:
    - La importancia de escalar riesgos a tiempo para prevenir problemas en un futuro.
    - Inconveniente supuesto por los cambios de alcance cuando un proyecto ya está en fase de pruebas integradas. En este caso, consistió en cancelación de casos, que realmente ahorran tiempo en ejecución pero es importante contabilizar el tiempo perdido en análisis y diseño para una casuística que finalmente no se implementa.

- Se realiza una comprobación, por parte de los equipos de pruebas de que se ha realizado un archivado de toda la documentación sobre los proyectos. Se trata de una actividad importante porque periódicamente se realizan auditorías sobre la documentación y debe estar todo perfectamente organizado por release y por proyecto. Por supuesto, archivo la documentación que tenemos de Reingeniería Web en el repositorio común de integradas.

- Como broche final, al finalizar cada release desde el equipo de gestión se realiza un informe al que cariñosamente denominamos *Postmorten* en el que se engloban los valores de medición de la calidad aportada por el equipo de pruebas de integración sobre la release. En él se realizan comparaciones con los resultados de releases anteriores y sin duda alguna es un informe completo en el que se plasman datos o KPIs, obtenidos de Quality Center y PVCS, como pueden ser: si las entregas se han realizado en el tiempo planificado o con retraso, el número de casos de prueba totales de la release, el número de casos resueltos OK, el número de incidencias abiertas, el tiempo de tardanza en las resoluciones de incidencias por parte de integradas, el número de incidencias abiertas y posteriormente anuladas por su no aplicación, el cumplimiento o no de la planificación inicialmente prevista en cada proyecto, el número de casos conseguidos OK respecto de los casos diseñados, ... Por supuesto, mostrar este informe aquí excedería el ámbito del proyecto pero sí me parecía relevante exponer la existencia de tal informe.

4.3.6 Monitorización y Control del Caso de Estudio

A lo largo de esta fase de monitorización y control se realizan una serie de tareas y todas ellas transcurren de forma paralela a las fases que hemos visto hasta ahora. La idea es poder controlar el estado, corregir posibles desviaciones sobre lo planificado o en algunos casos, más bien solo justificar, un comportamiento alejado a lo previsto durante la fase de pruebas.

Una de las primeras tareas a realizar es la monitorización de los esfuerzos y actividades que se realizan durante la fase de pruebas. Para ello utilizamos herramientas que nos ayudan a realizar un seguimiento exhaustivo del avance real frente al planificado. En el documento al que denominamos *Planificación* y del que ya hemos hablado en alguna ocasión, plasmamos la duración en cuanto a tiempo estimado en cada proyecto en las fases tempranas del ciclo de vida de las pruebas de integración. Después, según se va avanzando en el ciclo, lo que hacemos es ir anotando los retrasos notables que pueden producir un retraso en tiempo en la finalización de las pruebas. Por ejemplo, la imposibilidad de cumplir el tiempo estimado en ejecución de pruebas debido a la excesiva tardanza en la entrega de una incidencia, o el bloqueo total de un proyecto por indisponibilidad de un sistema. De hecho, al finalizar las pruebas integradas de Reingeniería Web, la planificación queda con el aspecto que se muestra en la figura 4-32. Mostrando claramente cómo se produjo un retraso en la entrega del software de dos días, que hubo un bloqueo de todos los casos de prueba del proyecto pendientes por ejecutar durante una jornada completa (aunque se tengan bloqueos parciales, en la *Planificación* solo se muestra el bloqueo total durante una jornada completa) y cuando se produjo la liberación del proyecto al usuario.

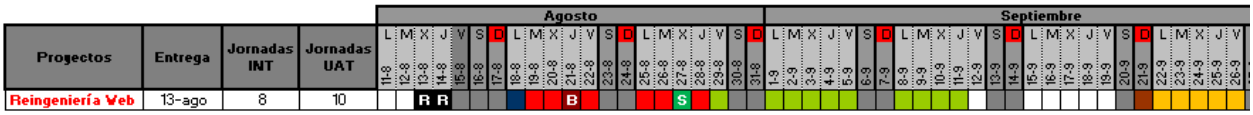


Figura 4-32: Planificación Final Reingeniería Web

Otra de las actividades que componen la fase de monitorización es el estudio de la progresión del plan de trabajo. Para lo cual, disponemos de un documento en el que se detallan los casos de prueba diseñados para cada proyecto y su estado de avance. El documento se llama *Status* y

forma parte de un informe diario generado por el equipo de gestión y que se entrega al cliente para que revise el estado del avance de cada release. En la figura 4-33, se muestra el documento *Status* correspondiente al día en que estaba planificado que se entregase Reingeniería Web. Y se puede consultar información como por ejemplo, que el proyecto 1 tiene 15 casos de prueba, que hoy ha avanzado un total de 46,67%, que tiene en estado Pendiente un 6,67% que corresponde con un caso de prueba, y que tiene en estado Bloqueado otro caso de prueba. Además, puede verse como Reingeniería Web tiene los 16 casos de prueba bloqueados.

	INTEGRACIÓN				
PAP 21/09	Nº Casos	Avance Global	Avance Diario	Pendiente	Bloqueado
Proyecto 1	15	86,67%	46,67%	6,67%	6,67%
Proyecto 2	230	17,39%	3,04%	0,00%	82,61%
Proyecto 3	119	25,21%	8,40%	7,56%	67,23%
Proyecto 4	8	0,00%	0,00%	0,00%	100,00%
Reingeniería Web	16	0,00%	0,00%	0,00%	100,00%
Proyecto 6	N/A	N/A	N/A	N/A	N/A
Proyecto 7	20	100,00%	0,00%	0,00%	0,00%
Proyecto 8	80	93,75%	0,00%	0,00%	6,25%
Proyecto 9	130	53,85%	5,38%	23,08%	23,08%
Proyecto 10	25	0,00%	0,00%	0,00%	100,00%
Proyecto 11	79	21,52%	18,99%	2,53%	75,95%
Proyecto 12	95	0,00%	0,00%	0,00%	100,00%
Proyecto 13	91	0,00%	0,00%	0,00%	100,00%

Figura 4-33: Status de la Release, miércoles 13

A lo largo de las pruebas integradas, otra de las actividades que componen la monitorización es el control sobre el entorno y los datos. Tal y como hemos visto, son los testers los que se ocupan de prepararse sus propios datos. Lo que se hace para llevar un control sobre los recursos, es una excel (se ve un pequeño detalle en la figura 4-21) en la que se detalla el entorno al que pertenecen, número de teléfono, tipo de recursos y proyecto para el que van a ser utilizados.

El entorno es propiedad del cliente el cual procura tenerlo siempre disponible. No se trata de un entorno que se "apague" al finalizar unas pruebas, porque lo cierto es que sobre el entorno de integración siempre se están ejecutando pruebas, ya que después de una release con X proyectos, viene otra release con Y proyectos. Por tanto, en cuanto al control del entorno, simplemente nos

encargamos de notificar mediante jtrac de entornos, tal y como hemos visto, cualquier incidencia que detectemos para que se solucione lo antes posible y poder seguir probando.

Durante las pruebas integradas, es muy importante la valoración e información sobre los riesgos que puedan afectar a un proyecto. En la fase de análisis deben detectarse los posibles riesgos que pueden complicar el avance esperado de uno o varios proyectos. Estos riesgos se detallarán y se informará al cliente de inmediato para que se tengan en cuenta a la hora de planificar en tiempo los proyectos y por supuesto, habrá que pensar en posibles alternativas llegado el momento de afrontar el riesgo cara a cara. Además, en los informes de seguimiento que se realizan a diario por el equipo de gestión, los riesgos deben estar muy presentes para que todo el personal implicado los conozca, pues es de gran importancia tener constancia en todo momento de los riesgos que impactan al proyecto para intentar mitigarlos. En el caso de Reingeniería Web, el riesgo inicial fue el que no se llegase a la subida a producción a tiempo dado que había en marcha una campaña publicitaria y la pérdida económica asociada sería importante. Pero al conseguir que se adelantase la entrega de software para poder comenzar las pruebas antes de lo previsto inicialmente y sobretodo, dado que las pruebas se finalizaron antes de lo planificado, el riesgo quedó mitigado totalmente.

La última actividad concerniente a la fase de monitorización es el conocimiento de los recursos. Se refiere a los recursos humanos disponibles y es una información interna, pues el cliente contrata un servicio y no a recursos específicamente. Pero sí es cierto que al cliente se le facilitan porcentajes en cuanto a recursos disponibles para ejecutar todos los proyectos de una release o los recursos disponibles para ejecutar una funcionalidad concreta, como podría ser el programa de puntos, que requiere una formación específica.

4.3.7 Mejora Continua del Caso de Estudio

Las actividades teóricas de la fase de mejora continua, en la práctica, quedan simplificadas en una serie de reuniones entre los diferentes equipos que componen las pruebas. A lo largo de una release, se realizan:

- Reuniones diarias en las que participan los jefes de los equipos de integración.
- Reuniones diarias entre el equipo de integración, el usuario y el cliente.
- Reuniones semanales entre el equipo de integración, el equipo de gestión de entornos, el equipo de instalación SCM y el cliente.

En todas ellas se realiza un análisis de la situación actual, se escalan los riesgos para su pronta resolución, se realizan propuestas para mejorar la situación y se hace un seguimiento sobre la evolución de los problemas planteados.

No puede olvidarse que además se realizan reuniones tanto antes de comenzar la release como al finalizar la misma para poder comentar los estados en el momento de la release y plantear las mejoras necesarias de cara a la siguiente release.

5 Conclusiones y Líneas Futuras

5.1 Conclusiones

Podemos obtener una serie de conclusiones del trabajo realizado:

- Es evidente la evolución de la ingeniería del software a lo largo de su historia y aquí hemos visto dos ejemplos claros de ello, que son:
    - Se produce una aparición de nuevas técnicas de diseño, como son las técnicas dinámicas de la experiencia.
    - Por suerte, se va volatilizando la negatividad sobre las pruebas. Esto es gracias a que las empresas que han invertido en pruebas han visto reducido los costes a emplear en solucionar problemas en el software, debido a la localización de los mismos en fases tempranas de su creación.
- Se va comprobando que las pruebas van cobrando mayor importancia, hasta el punto de tener su propio ciclo de vida y que incluso existan empresas como la mía, que se dedica única y exclusivamente al sector de las pruebas de software.
- Aquí se podría apuntar de nuevo la conclusión obtenida de la comparación entre Pressman e ISTQB donde se verifica que a pesar de las diferencias entre ambas fuentes de teoría de la informática, los pilares básicos sobre los que se asienta la ingeniería del software siguen siendo los mismos.

5.2 Líneas Futuras

En cuanto al futuro de las pruebas se refiere, atisbo dos líneas futuras claramente, las cuales se van introduciendo cada día más y más en el diario de cualquier empresa dedicada a las pruebas:

- **Automatización**: En el proyecto hemos hablado un poco de las pruebas automatizadas. Se trata de pruebas que se pueden ejecutar automáticamente, sin la necesidad de un tester que vaya realizando la ejecución de la prueba. Son pruebas para las que se dedica tiempo en su diseño e implementación y posteriormente en el mantenimiento, puesto que siempre se pueden producir cambios en las aplicaciones sobre las que se prueba, que obligan a modificar el script con el que se ejecuta la prueba automatizada. Y su ventaja es que se dedica el tiempo mínimo en su ejecución. La idea sobre la automatización consiste en automatizar todo lo posible las pruebas, dejando el testing manual para las pruebas más artesanales.

  La característica más importante por la que se emplea cada vez más la automatización de casos de prueba es sin duda el ahorro de tiempo en ejecución que supone para un equipo de pruebas disponer de este tipo de pruebas entre las que se tenga para ejecutar. Además, tener pruebas automatizadas permite el hecho de poder probar en paralelo, en background y sin atender a la ejecución de la prueba, solo analizando el resultado que obtenga.

  Sobretodo interesa poner foco en la automatización de pruebas que sean repetitivas, como son las pruebas de regresión en las que se repite la misma prueba en diferentes momentos o como pueden ser las pruebas no funcionales de carga, volumen y/o estrés en las que se repite la misma prueba una y otra vez en el mismo momento.

- **Metodologías ágiles**: También son llamadas metodología Scrum y se caracterizan por tener una estrategia incremental en lugar de la secuencial que se utiliza normalmente y en la que además, se produce un solapamiento de las fases. Asimismo son una metodología fácil de aprender y que requiere poco esfuerzo para su implantación.

  Las metodologías Scrum se componen de equipos autoorganizados en los que se fomenta la continua comunicación de los miembros para que entre todos se pueda realizar una buena gestión de los requisitos que solicite el cliente, se puedan obtener algunos resultados por anticipado, se mitiguen los riesgos localizados y se aumente la

calidad y productividad; todo ello con un importante valor de flexibilidad y adaptación continua.

Una de las bases de las metodologías ágiles es que se da por hecho que el cliente puede modificar los requisitos acordados inicialmente en cualquier momento para lo cual, la mayoría de metodologías responden de manera rígida provocando que cualquier modificación sea muy costosa de realizar. Sin embargo, Scrum se centra en que el equipo sea capaz de responder satisfactoriamente a cualquier requisito inesperado.

6 Referencias

- [1] "Ingeniería del Software - Un enfoque práctico" por Roger S.Pressman
- [2] "Formación para el Probador Certificado - Nivel Básico" de la certificación ISTQB
- [3] Documentación interna de MTP
- [4] HP - Quality Center
- [5] PVCS
- [6] Jtrac
- [7] SecureCRT

7 Anexos

7.1 Caso 1, Ejecución 1

Es la operación de logado 12010 y se produce un fallo en MIDDLEWARE. Es una petición lanzada por la librería de acceso del sistema WEB. Va dirigida al sistema MIDDLEWARE que es quien se encarga de interpretar el número de petición que le llega para procesar a qué sistema se lo envía. WEB envía un XML donde incluye los parámetros de entrada informados, que en este caso consisten en indicar quien es el sistema origen y cual es el MSISDN:

2013-08-19 15:38:05 (2013-08-19 14:38:05.038910000Z): subject=MOVILES.INT.**PETICION.MDW.12010**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="RV_PUBLICADOR_ACCMDW OUT"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw"} ^data^={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML**="<XML_Peticion>**

<DatosConsulta>

<Origen>WEB</Origen>

<MSISDN>111111111</MSISDN>

</DatosConsulta>

</XML_Peticion>"} SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-19 14:38:05.028000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-19 14:38:05.023000000Z}}

Ahora es MIDDLEWARE quien envía al sistema FACTPOS una petición 03018 que sirve para preguntarle si el MSISDN que le envía es prepago o pospago. Como puede observarse, justo al final de esta petición, el MSISDN que MIDDLEWARE está enviando, viaja vacío. Esto se traduciría como que se le está preguntando a FACTPOS por el tipo de un MSISDN sin numeración.

2013-08-19 15:38:05 (2013-08-19 14:38:05.046315000Z): subject=MOVILES.INT.**PETICION.FACTPOS.03018**, reply=MOVILES.INT.RESPUESTA.MDW.03018.12010, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03018" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="C_03018_Peticion" SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-19 14:38:05.044000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-19 14:38:05Z MsisdnAmpliado={^idx^=1 ^class^="MsisdnAmpliado" **Msisdn=""**}}}

En el siguiente tramo de log, es FACTPOS quien contesta a MIDDLEWARE sobre la petición que recibió de 03018, la cual forma parte de la operación 12010 de logado y se indica con "03018.12010". Como no recibió la numeración del MSISDN contesta con que el MSISDN no existe, ya que el vacío no se encuentra registrado en su base de datos como un MSISDN.

2013-08-19 15:38:05 (2013-08-19 14:38:05.828774000Z): subject=MOVILES.INT.**RESPUESTA.MDW.03018.12010**, message={^type^=1 ^pfmt^=10 ^ver^=30 ^encoding^=1 SecurityPlugin={SecurityPlugin="RV_PUB_03018_ConsPrePospagoEndPoint"} ^prefixList^={1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas" default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03018"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="C_03018_Respuesta" **CabeceraError={**^class^="CabeceraError" ^idx^=1 Codigo="NL305" Sistema="FACTPOS" Tipo="Consultas" **Descripcion="El MSISDN no existe"** TimeStamp=2013-08-19 14:38:05.050630000Z**}** SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-19 14:38:05.050630000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-19 14:38:05Z}}

MIDDLEWARE recibe la contestación de FACTPOS y como se trata de un error no continua con el flujo de operación esperado para el logado o 12010, y comunica de inmediato al sistema origen la respuesta recibida. La descripción es el único valor que se muestra literalmente por pantalla, de ahí la importancia de poder contar con la visualización de los logs porque en ocasiones la descripción puede ser algo confusa, pero siempre está asociada a un código de error que puede facilitar su entendimiento. Además, revisando los logs por completo se pueden obtener resultados complejos como el que estamos tratando, donde el error realmente lo origina un sistema diferente al que comunica el error como si fuese suyo.

2013-08-19 15:38:07 (2013-08-19 14:38:07.277895000Z):subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.12010**.weblogi1.2241277535.817, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS.MDW_RESP" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-19 14:38:07.275000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-19 14:38:05Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Error>

<Codigo>NL305</Codigo>

<Descripcion>El MSISDN no existe</Descripcion>

<Tipo>Consultas</Tipo>

<Sistema>FACTPOS</Sistema>

<TimeStamp>2013-08-19 14:38:05</TimeStamp>

</Error>

</XML_Respuesta>"} CabeceraError={^idx^=1 ^class^="CabeceraError" Codigo="NL305" Sistema="FACTPOS" Tipo="Consultas" Descripcion="El MSISDN no existe" TimeStamp=2013-08-19 14:38:05Z}}}

7.2 Caso 1, Ejecución 2

Se trata de la reejecución de este caso de logado, y la primera vez en la que el resultado es el esperado. En primer lugar tenemos la petición de logado por parte del sistema WEB al sistema MIDDLEWARE:

2013-08-20 12:30:15 (2013-08-20 11:30:15.038910000Z): subject=MOVILES.INT.**PETICION.MDW.12010**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="RV_PUBLICADOR_ACCMDW OUT"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw"} ^data^={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML="<XML_Peticion>

<DatosConsulta>

<Origen>WEB</Origen>

<MSISDN>111111111</MSISDN>

</DatosConsulta>

</XML_Peticion>"} SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-20 11:30:15.028000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-20 11:30:15.023000000Z}}

Contenida en el flujo de la 12010, la primera operación a ejecutar es la consulta sobre si el MSISDN es prepago o pospago y se hace a través de la 03018 desde MIDDLEWARE a FACTPOS. En esta ocasión el MSISDN sí viaja informado:

2013-08-20 12:30:15 (2013-08-20 11:30:15.046315000Z): subject=MOVILES.INT.**PETICION.FACTPOS.03018**, reply=MOVILES.INT.RESPUESTA.MDW.03018.12010, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03018" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="C_03018_Peticion" SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-20

11:30:15.044000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-20 11:30:15Z MsisdnAmpliado={^idx^=1 ^class^="MsisdnAmpliado" **Msisdn="111111111"**}}}

FACTPOS responde indicando que es un pospago a través del campo TarifFlg con valor 1 (si fuese 0 sería un prepago) y además devuelve la tarifa, que es el valor contenido en Tmcode="822":

2013-08-20 12:30:16 (2013-08-20 11:30:16.828774000Z): subject=MOVILES.INT.RESPUESTA.MDW.03018.12010, message={^type^=1 ^pfmt^=10 ^ver^=30 ^encoding^=1 SecurityPlugin={SecurityPlugin="RV_PUB_03018_ConsPrePospagoEndPoint"} ^prefixList^={1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas" default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03018"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="C_03018_Respuesta" MsisdnAmpliado={^class^="MsisdnAmpliado" ^idx^=1 Msisdn="111111111"} ContratoBasico={^class^="ContratoBasico" ^idx^=1 **TarifFlg="1"Tmcode="822"**} SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-20 11:30:15.050630000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-20 11:30:15Z}}

MIDDLEWARE recibe la respuesta de la 03018 y actúa en consecuencia, lanzando una 03045 o lo que es lo mismo, una consulta de los datos del cliente a FACTPOS ya que el MSISDN es un pospago:

2013-08-20 12:30:16 (2013-08-20 11:30:16.835279000Z): subject=MOVILES.INT.**PETICION.FACTPOS.03045**, reply=MOVILES.INT.RESPUESTA.MDW.03045.12010, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03045" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="C_03045_Peticion" SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-20 11:30:16.831000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-20 11:30:15Z DatosConsulta={^idx^=1 ^class^="DatosConsulta" Origen="WEB" MSISDN="111111111"}}}

FACTPOS contesta a la petición de la consulta de datos de cliente a MIDDLEWARE, enviando una serie de datos que contiene sobre el cliente:

2013-08-20 12:30:16 (2013-08-20 11:30:17.270721000Z): subject=MOVILES.INT.**RESPUESTA.MDW.03045.12010**, message={^type^=1 ^pfmt^=10 ^ver^=30 ^encoding^=1 SecurityPlugin={SecurityPlugin="RV_PUB_03045_ListaDatosBasicosClienteEndPoint"} ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03045" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas" 2="/tibco/public/sequence/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="**C_03045_Respuesta" InfoCustomer={^class^="InfoCustomer" ^idx^=1 CustId="91919391" CustCode="9.11953921" IdentityType="2" IdentityNum="16034418Z" Pricegroup="POSTRESD" Billcycle="10" Activated="20110704" MonthlyBill="1" SeqNumber="1"} InfoMSISDN={^class^="InfoMSISDN" ^idx^=1 CategoriaOC="OC1" CCEEAttrib={^class^="CCEEAttrib" ^idx^=1 NumAttrib="Y"} NumLineas="1" MsisdnDataList={^class^="SecMsisdnDataList" ^idx^=2 ^1^={^class^="MsisdnDataList" ^idx^=1 MSISDN="111111111" TIPO="2"}} CoStatus="a" Rateplan="822" ShdesTmcod="T1HN" TmcodeCRM="1822" Spcode="493" ShdesSpcode="S1H" SpcodeCRM="0722" ActivatedLinea="20121011" FirstName="Rodolfo" LastName="Sancho"**} SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS" Estado=0 FiltroMDW="weblogi1.2241277535.817" TimeStamp=2013-08-20 11:30:17.265428000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-20 11:30:15Z}}

Finaliza el flujo MIDDLEWARE montando las respuestas que ha recibido de todas sus peticiones. Las monta en un XML y se la envía al sistema origen, en este caso, a WEB. Y WEB lo que hace es pintar todos los datos recibidos en respuesta al logado por pantalla, de manera que se muestra una ventana de acceso con los datos del cliente logado:

2013-08-20 12:30:16 (2013-08-20 11:30:17.277895000Z): subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.12010**.weblogi1.2241277535.817, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="kqwGpicbwwGlt-FuGTzzzt9Ezzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116030"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.2241277535.121112093805.-4238906221159538667" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS.MDW_RESP" Estado=0 FiltroMDW="weblogi1.2241277535.817"

TimeStamp=2013-08-20 11:30:17.275000000Z IdAux="2241277535.-4238906221159538667.817" TimeOut=60000 TimeStampOrigen=2013-08-20 11:30:15Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<MSISDN>

<Msisdn>111111111</Msisdn>

</MSISDN>

<Contrato>

<TariFlag>1</TariFlag>

</Contrato>

<InfoCustomer>

<CustId>91919391</CustId>

<CustCode>9.11953921</CustCode>

<IdentityType>2</IdentityType>

<IdentityNum>16034418Z</IdentityNum>

<Pricegroup>POSTRESD</Pricegroup>

<Billcycle>10</Billcycle>

<Activated>20110704</Activated>

<MonthlyBill>1</MonthlyBill>

<SeqNumber>1</SeqNumber>

</InfoCustomer>

<InfoMSISDN>

<CategoriaOC>OC1</CategoriaOC>

<CCEEAttrib>

<NumAttrib>Y</NumAttrib>

</CCEEAttrib>

<NumLineas>1</NumLineas>
<MsisdnDataList>
<Msisdn>111111111</Msisdn>
<TIPO>2</TIPO>
</MsisdnDataList>
<CoStatus>a</CoStatus>
<Rateplan>822</Rateplan>
<ShdesTmcode>T1HN</ShdesTmcode>
<TmcodeCRM>1822</TmcodeCRM>
<Spcode>493</Spcode>
<ShdesSpcode>S1H</ShdesSpcode>
<SpcodeCRM>0722</SpcodeCRM>
<ActivatedLinea>20121011</ActivatedLinea>
<FirstName>Rodolfo</FirstName>
<LastName>Sancho</LastName>
</InfoMSISDN>
</XML_Respuesta>"}}}

7.3 Caso 2, Ejecución 2

El log es bastante parecido aldel caso anterior: WEB envía la petición de logado a través de una 12010 a MIDDLEWARE. Eso sí, en esta ocasión con un recurso prepago.

2013-08-20 12:45:10 (2013-08-20 11:45:10.349309000Z): subject=MOVILES.INT.**PETICION.MDW.12010**, […]**="<XML_Peticion>**
<DatosConsulta>
<Origen>WEB</Origen>
<MSISDN>111111112</MSISDN>
</DatosConsulta>
</XML_Peticion[…]

MIDDLEWARE pregunta a FACTPOS por el tipo de cliente en la 03018:

2013-08-20 12:45:10 (2013-08-20 11:45:10.355804000Z): subject=MOVILES.INT.**PETICION.FACTPOS.03018**, reply=MOVILES.INT.RESPUESTA.MDW.03018.12010, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03018" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="C_03018_Peticion" SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:10.353000000Z IdAux="3598061166.982490658031006690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z MsisdnAmpliado={^idx^=1 ^class^="MsisdnAmpliado" Msisdn="111111112"}}}

FACTPOS contesta sobre el tipo de cliente indicando que se trata de un prepago con el campo TarifFlg con valor 0:

2013-08-20 12:45:12 (2013-08-20 11:45:12.498418000Z): subject=MOVILES.INT.**RESPUESTA.MDW.03018.12010**, message={^type^=1 ^pfmt^=10 ^ver^=30 ^encoding^=1 SecurityPlugin={SecurityPlugin="RV_PUB_03018_ConsPrePospagoEndPoint"} ^prefixList^={1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas" default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03018"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="C_03018_Respuesta" MsisdnAmpliado={^class^="MsisdnAmpliado" ^idx^=1 Msisdn="111111112"} ContratoBasico={^class^="ContratoBasico" ^idx^=1 **TarifFlg="0"** Tmcode="1"} SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS" Estado=0 FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:10.359010000Z IdAux="3598061166.982490658031006690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z}}

La complejidad de este flujo reside en este punto. Para los prepagos, desde que entró en vigor una ley en la que se obliga a registrar a los dueños de los prepagos, en FACTPOS se almacena una parte de los datos del cliente y además en FACTPRE donde siempre se habían almacenado los datos si el cliente quería darlos, se almacena otra parte. Dicho esto, MIDDLEWARE va a aplicar su lógica para lanzar una petición de datos de cliente a FACTPOS con una 03045 y otra a FACTPRE con una 10004:

2013-08-20 12:45:12 (2013-08-20 11:45:12.505295000Z): subject=MOVILES.INT.**PETICION.FACTPOS.03045**, reply=MOVILES.INT.RESPUESTA.MDW.03045.12010, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03045" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="C_03045_Peticion" SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW" Estado=0 FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:12.501000000Z IdAux="3598061166.982490658031006690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z DatosConsulta={^idx^=1 ^class^="DatosConsulta" Origen="WEB" MSISDN="111111112"}}}

FACTPOS responde a MIDDLEWARE con los datos del cliente prepago:

2013-08-20 12:45:12 (2013-08-20 11:45:12.567735000Z): subject=MOVILES.INT.**RESPUESTA.MDW.03045.12010**, message={^type^=1 ^pfmt^=10 ^ver^=30 ^encoding^=1 SecurityPlugin={SecurityPlugin="RV_PUB_03045_ListaDatosBasicosClienteEndPoint"} ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03045" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas" 2="/tibco/public/sequence/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="**C_03045_Respuesta" InfoCustomer={^class^="InfoCustomer" ^idx^=1 CustId="21379391" CustCode="6.04463921" IdentityType="2" IdentityNum="66850980P" Pricegroup="PRE" Billcycle="" Activated="20120525" MonthlyBill="1" Email="prepago@pruebas.com" SeqNumber="2"} InfoMSISDN={^class^="InfoMSISDN" ^idx^=1 CCEEAttrib={^class^="CCEEAttrib" ^idx^=1 NumAttrib="N"} NumLineas="1" MsisdnDataList={^class^="SecMsisdnDataList" ^idx^=2 ^1^={^class^="MsisdnDataList" ^idx^=1 MSISDN="111111112" TIPO="1"}} CoStatus="a" Rateplan="1" ShdesTmcod="PPAGO" SpcodeCRM="3951" ActivatedLinea="20120912" FirstName="María" LastName="Pérez"**} SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS" Estado=0 FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:12.561621000Z IdAux="3598061166.982490658031006690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z}}

Ahora MIDDLEWARE pregunta a FACTPRE sobre los datos del cliente:

2013-08-20 12:45:12 (2013-08-20 11:45:12.573273000Z): subject=MOVILES.INT.**PETICION.FACTPRE_CNT.10004**, reply=MOVILES.INT.RESPUESTA.MDW.10004.12010, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10004" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="C_10004_Peticion" SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS.MDW" Estado=0 FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:12.570000000Z IdAux="3598061166.982490658031006690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z DatosConsultaPropiedades={^idx^=1 ^class^="DatosConsultaPropiedades" Origen="WEB" MSISDN="111111112"}}}

Y FACTPRE contesta:

2013-08-20 12:45:12 (2013-08-20 11:45:12.857847000Z): subject=MOVILES.INT.**RESPUESTA.MDW.10004.12010**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10004" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT"} SecurityPlugin={SecurityPlugin="RV_PUBLISHER_CONSULTA_PROPIEDADESEndPoint OUT"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="**C_10004_Respuesta" TelefonoBasico={^idx^=1 ^class^="TelefonoBasico" MSISDN="111111112" ICCID="8934011000220058589" IMSI="214031010001681"} TelefonoIdioma={^idx^=1 ^class^="TelefonoIdioma" LanguageId=0} TelefonoTarifa={^idx^=1 ^class^="TelefonoTarifa" TariffId=85 LastChangeTariff=""} TelefonoCiudad={^idx^=1 ^class^="TelefonoCiudad" LastTZChgDate="" CityId=0} TelefonoFecha={^idx^=1 ^class^="TelefonoFecha" InstallationDate="20121018093322" ActivationDate="" DeactiveDate="" Deactivated="" ExpiryDate="" LastCallDate=""} TelefonoSaldo={^idx^=1 ^class^="TelefonoSaldo" SaldoRecargas=5.000000 FechaCaducidadSaldoRecargas="" SaldoPromocional=0.000000 FechaCaducidadSaldoPromocional="99991231000000"} TelefonoResto={^idx^=1 ^class^="TelefonoResto" PIN="0000" PUK="12345678" PIN2="1111" PUK2="12345678" State=2 ServiceProviderId="RET" NumReloads=0 NumTariffs=0 Services=0 TipoSIM="00" TipoServicio="0" TipoLinea="0" fdState=0 fdDate="" fdOPCost=0.000000 fdOPNumber=0} DatosAnticipo={^idx^=1 ^class^="DatosAnticipo" ImporteAnticipo=0.000000 FechaAnticipo="" NumeroAnticipos=0 EstadoAnticipo=0**} SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS.MDW.FACTPRE_CNT" Estado=0 FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:12.578000000Z IdAux="3598061166.98249065803100690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z}}

Por último, MIDDLEWARE monta los datos recibidos en un XML y se lo envía al sistema peticionario WEB, quien los muestra por pantalla:

2013-08-20 12:45:12 (2013-08-20 11:45:12.865236000Z):subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.12010**.weblogi1.3598061166.232, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="EF8c/GC6yC-a7E5zN7zzyWmEzzw" ^1^[1]="BW.OfertaDatosRes-Process_Archive_tibco_des.Procesos/MDW/12010/Pd_12010_ConsAcceAplicat.process.Job-116441"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.3598061166.131112092610.982490658031006690" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPOS.MDW.FACTPOS.MDW.FACTPRE_CNT.MDW_RESP" Estado=0

FiltroMDW="weblogi1.3598061166.232" TimeStamp=2013-08-20 11:45:12.863000000Z IdAux="3598061166.982490658031006690.232" TimeOut=60000 TimeStampOrigen=2013-08-20 11:45:10Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<MSISDN>

<Msisdn>111111112</Msisdn>

</MSISDN>

<Contrato>

<TariFlag>0</TariFlag>

</Contrato>

<InfoCustomer>

<CustId>21379391</CustId>

<CustCode>6.04463921</CustCode>

<IdentityType>2</IdentityType>

<IdentityNum>66850980P</IdentityNum>

<Pricegroup>PRE</Pricegroup>

<Billcycle></Billcycle>

<Activated>20120525</Activated>

<MonthlyBill>1</MonthlyBill>

<Email>prepago@pruebas.com</Email>

<SeqNumber>2</SeqNumber>

</InfoCustomer>

<InfoMSISDN>

<NumLineas>1</NumLineas>

<MsisdnDataList>

<Msisdn>111111112</Msisdn>
<TIPO>1</TIPO>
</MsisdnDataList>
<CoStatus>a</CoStatus>
<Rateplan>1</Rateplan>
<ShdesTmcode>PPAGO</ShdesTmcode>
<ShdesSpcode/>
<SpcodeCRM>3951</SpcodeCRM>
<ActivatedLinea>20120912</ActivatedLinea>
<FirstName>María</FirstName>
<LastName>Pérez</LastName>
</InfoMSISDN>
<TelefonoTarifa>
<TariffId>85</TariffId>
</TelefonoTarifa>
<TelefonoResto>
<State>2</State>
<TipoServicio>0</TipoServicio>
<TipoSIM>00</TipoSIM>
</TelefonoResto>
</XML_Respuesta>"}}}

7.4 Caso 3, Ejecución 2

Como en el fondo es muy parecido al resultado obtenido en el caso 1 porque solo va a FACTPOS, expongo el XML de la petición y el XML de la respuesta final que monta MIDDLEWARE.

2013-08-20 12:58:55 (2013-08-20 11:58:55.939653000Z): subject=**MOVILES.INT.PETICION.MDW.12010**[…]

<DatosConsulta>

<Origen>WEB</Origen>

<MSISDN>111111113</MSISDN>

</DatosConsulta> […]

2013-08-20 12:58:58 (2013-08-20 11:58:58.526371000Z): subject=**MOVILES.INT.RESPUESTA.libAccMiddleware.12010**. […]

<XML_Respuesta>

<MSISDN>

<Msisdn>111111113</Msisdn>

</MSISDN>

<Contrato>

<TariFlag>1</TariFlag>

</Contrato>

<InfoCustomer>

<CustId>86179391</CustId>

<CustCode>2.62463921</CustCode>

<IdentityType>3</IdentityType>

<IdentityNum>B00986679</IdentityNum>

<Pricegroup>POSTEMPB</Pricegroup>

<Billcycle>05</Billcycle>
<Activated>20120523</Activated>
<MonthlyBill>1</MonthlyBill>
<SeqNumber>1</SeqNumber>
</InfoCustomer>
<InfoMSISDN>
<IdMercado>Masivo</IdMercado>
<CategoriaOC>OC0</CategoriaOC>
<NumLineas>3</NumLineas>
<CoStatus>a</CoStatus>
<Rateplan>052</Rateplan>
<ShdesTmcode>T2E</ShdesTmcode>
<TmcodeCRM>9052</TmcodeCRM>
<Spcode>624</Spcode>
<ShdesSpcode>S4D1</ShdesSpcode>
<SpcodeCRM>7052</SpcodeCRM>
<ActivatedLinea>20120808</ActivatedLinea>
<OriginDate>20120830</OriginDate>
<FirstName>Javier</FirstName>
<LastName>López</LastName>
</InfoMSISDN>
</XML_Respuesta>[…]

7.5 Caso 6, Ejecución 1

Se envía la petición del alta de promoción desde WEB a MIDDLEWARE:

2013-08-20 15:56:16 (2013-08-20 14:56:16.817345000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="SEC_RV_PUBLICADOR_ACCMDW OUT"} ^tracking^={^id^="e2/QAYbuyYJBck5ze9zzyWmEzzw"} ^data^={_data={user="weblogi1" _data={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML="<XML_Peticion>
<Promocion>
<IdPeticion>201211131556160703-111111111</IdPeticion>
<IdPromo>9162</IdPromo>
<MSISDN>111111111</MSISDN>
<Accion>A</Accion>
<SistemaOrigen>WEB</SistemaOrigen>
<AtributosPromo>
<Nombre>VOLUMEN</Nombre>
<Accion>A</Accion>
<Valor>500</Valor>
</AtributosPromo>
<AtributosPromo>
<Nombre>PRECIO</Nombre>
<Accion>A</Accion>
<Valor>6</Valor>
</AtributosPromo>
</Promocion>
</XML_Peticion>"} SourceMsgId="weblogi1.8318769585.131112035616.-6157930864313680492" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.8318769585.528" TimeStamp=2013-08-20 14:56:16.802000000Z IdAux="8318769585.-6157930864313680492.528" TimeOut=60000 TimeStampOrigen=2013-08-20 14:56:16.798000000Z} cert=130 sig=[46 opaque bytes]} _acld="INT.SECURITY"}}

MIDDLEWARE envía la petición a FACTPOS:

2013-08-20 **15:56:16** (2013-08-20 14:56:16.846058000Z): subject=MOVILES.INT.**PETICION.FACTPOS.26069**, reply=MOVILES.INT.RESPUESTA.MDW.26069.09030, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOS/C_26069" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOS" 2="/tibco/public/sequence/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOS"} ^tracking^={^id^="e2/QAYbuyYJBck5ze9zzyWmEzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-776404"} ^data^={_data= {user="tibco_des" _data={^class^="C_26069_Peticion" SourceMsgId="weblogi1.8318769585.131112035616.-6157930864313680492" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.8318769585.528" TimeStamp=2013-08-20 14:56:16.840000000Z IdAux="8318769585.-6157930864313680492.528" TimeOut=60000 TimeStampOrigen=2013-08-20 14:56:16Z MsisdnAmpliado={^idx^=1 ^class^="MsisdnAmpliado" MSISDN="111111111"} DatosAccionAmpliado={^idx^=1 ^class^="DatosAccionAmpliado" Accion="A"} Promocion={^idx^=1 ^class^="Promocion" IdPromocion="9162" SistemaOrigen="WEB" AtributosPromocion={^idx^=2 ^class^="SecAtributosPromocion" ^1^={^idx^=1 ^class^="AtributosPromocion" Nombre="VOLUMEN" Valor="500" Accion="A"} ^2^={^idx^=1 ^class^="AtributosPromocion" Nombre="PRECIO" Valor="6" Accion="A"}}} DatosOrigen={^idx^=1 ^class^="CDatosOrigen"}} cert=126 sig=[46 opaque bytes]} _acld="INT.SECURITY"}}

Pasados 30 segundos FACTPOS no contesta, por lo que MIDDLEWARE comunica el timeout al sistema origen WEB:

2013-08-20 **15:56:46** (2013-08-20 14:56:46.854754000Z):subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.weblogi1.8318769585.528, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="e2/QAYbuyYJBck5ze9zzyWmEzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-776404"} ^data^= {^class^="S_Respuesta" SourceMsgId="weblogi1.8318769585.131112035616.-6157930864313680492" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.MDW_RESP" Estado=0 FiltroMDW="weblogi1.8318769585.528" TimeStamp=2013-08-20 14:56:46.850000000Z IdAux="8318769585.-6157930864313680492.528" TimeOut=60000 TimeStampOrigen=2013-08-20 14:56:16Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>
<XML_Respuesta>
<Promocion>
<IdPeticion>201211131556160703-111111111</IdPeticion>
<IdPromo>9162</IdPromo>
</Promocion>

<Error>
<Codigo>MDW_ERR-093005</Codigo>
<Descripcion>Se ha producido un TIMEOUT en el sistema FACTPOS(26069)</Descripcion>
<Tipo>MDW</Tipo>
<Sistema>MDW</Sistema>
<TimeStamp>2013-08-20T15:56:46.849+01:00</TimeStamp>
<IdHost>tibco_des</IdHost>
</Error>
</XML_Respuesta>"} CabeceraError={^idx^=1 ^class^="CabeceraError" Codigo="MDW_ERR-093005" Sistema="MDW" Tipo="MDW" Descripcion="Se ha producido un TIMEOUT en el sistema FACTPOS(26069)" TimeStamp=2013-08-20 14:56:46.849000000Z IdHost="tibco_des"}}}

7.6 Caso 9

Se publica el alta de la promoción New TV con una 09030 desde WEB. Lo primero a comprobar es que los datos que viajan en los parámetros son: el MSISDN escogido 111111115, el código de promoción 2221 que pertenece a New TV y que la acción a realizar es el alta, que se refleja con A:

2013-08-22 12:30:36 (2013-08-22 11:30:36.422279000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="SEC_RV_PUBLICADOR_ACCMDW OUT"}^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw"} ^data^={_data={user="weblogi1" _data={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML="<XML_Peticion>

<Promocion>

<IdPeticion>201211191759360390-111111115</IdPeticion>

<IdPromo>2221</IdPromo>

<MSISDN>111111115</MSISDN>

<Accion>A</Accion>

<SistemaOrigen>WEB</SistemaOrigen>

</Promocion>

</XML_Peticion>"} SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:30:36.405000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:30:36.399000000Z} cert=130 sig=[46 opaque bytes]} _acld="INT.SECURITY"}}

El sistema maestro de esta nueva promoción es FACTPRE, así que MIDDLEWARE dirige la petición del alta hacia FACTPRE:

2013-08-22 12:30:36 (2013-08-22 11:30:36.460774000Z): subject=MOVILES.INT.**PETICION.FACTPRE_CNT.10001**, reply=MOVILES.INT.RESPUESTA.MDW.10001.09030, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10001" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777348"} ^data^={_data={user="tibco_des" _data={^class^="C_10001_Peticion" SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:30:36.448000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:30:36Z DatosPromocion={^idx^=1 ^class^="DatosPromocion" IdPeticion="201211191759360390-111111115" SistemaOrigen="WEB" Accion="A" MSISDN="111111115" IdPromocion="2221"}} cert=126 sig=[46 opaque bytes]} _acld="INT.SECURITY"}}

FACTPRE contesta a MIDDLEWARE:

2013-08-22 12:30:36 (2013-08-22 11:30:36.738685000Z): subject=MOVILES.INT.**RESPUESTA.MDW.10001.09030**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10001"} SecurityPlugin={SecurityPlugin="RV_PUBLISHER_ALTA_BAJA_MODIF_PROMOCIONEndPoint OUT"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777348"} ^data^={^class^="C_10001_Respuesta" SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:30:36.467000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:30:36Z}}

Finalmente MIDDLEWARE monta la respuesta recibida del sistema destino en un XML que entrega al sistema origen WEB. Para todas las promociones siempre ha funcionado de la misma forma, si es OK, en la respuesta indica el MSISDN y el código de la promoción:

2013-08-22 12:30:36 (2013-08-22 11:30:36.747808000Z):subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.weblogi1.6728595321.541, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777348"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW_RESP" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:30:36.743000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:30:36Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Promocion>

<IdPeticion>201211191759360390-111111115</IdPeticion>

<IdPromo>2221</IdPromo>

</Promocion>

</XML_Respuesta>"}}}

7.7 Caso 11

Se realiza la consulta de New TV desde WEB, por tanto la finalidad de esta prueba es que se muestre en la respuesta la promoción asociada al recurso para el que se realiza la consulta. Comenzamos con la petición de consulta desde WEB a MIDDLEWARE a través de una 09020. En los parámetros de entrada viajan el MSISDN, el sistema origen y la fecha de rango de búsqueda de promociones activas, este valor no se

puede controlar desde la WEB, se introduce automáticamente y lo que hace es buscar las promociones que el cliente tienes activas a día de hoy:

2013-08-22 12:40:57 (2013-08-22 11:40:57.988876000Z): subject=MOVILES.INT.**PETICION.MDW.09020**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^=

{default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="RV_PUBLICADOR_ACCMDW OUT"}

^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw"} ^data^={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML="<XML_Peticion>

<Solicitud>

<MSISDN>111111115</MSISDN>

<CustomerId/>

<ConsultaLigera>N</ConsultaLigera>

<Sistema>WEB</Sistema>

<FechaDesde>20130822</FechaDesde>

<FechaHasta>20130822</FechaHasta>

<FlagNoOCC>X</FlagNoOCC>

</Solicitud>

</XML_Peticion>"} SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:57.979000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57.974000000Z}}

Como las promociones pueden estar activas en unos sistemas u otros dependiendo de qué sistema sea el maestro de cada promoción, lo que se hace al lanzar una consulta 09020 es que MIDDLEWARE pregunta a los sistemas posibles que pueden contener promociones activas para el cliente, y son: FACTPRE, RESER y FACTPOS. Comienza por FACTPRE:

2013-08-22 12:40:58 (2013-08-22 11:40:58.000448000Z): subject=MOVILES.INT.**PETICION.FACTPRE_CNT.10002**, reply=MOVILES.INT.RESPUESTA.MDW.10002.09020, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10002" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="C_10002_Peticion" SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:57.997000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z DatosConsulta={^idx^=1 ^class^="DatosConsulta" SistemaOrigen="WEB" Tipo="N" MSISDN="111111115" FechaDesde="20130822" FechaHasta="20130822"}}}

MIDDLEWARE recibe la respuesta de FACTPRE:

2013-08-22 12:40:58 (2013-08-22 11:40:58.815413000Z): subject=MOVILES.INT.**RESPUESTA.MDW.10002.09020**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10002" 1="/tibco/public/sequence/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT" 2="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT"} SecurityPlugin={SecurityPlugin="RV_PUBLISHER_CONSULTA_PROMOCIONEndPoint OUT"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="C_10002_Respuesta" Promociones={^idx^=1 ^class^="SecPromociones" ^1^={^idx^=2 ^class^="Promociones" IdPromocion="2221" Nombre="New TV" SistemaMaestro="FACTPRE" Estado="ACTIVA" FechaActivacion="20130822" ParametrosPromocion={^idx^=1 ^class^="SecParametrosPromocion" ^1^={^idx^=2 ^class^="ParametrosPromocion" AtributosDinamicos={^idx^=2 ^class^="AtributosDinamicos" Nombre="PRODUCTO" Valor="TELEVISION"} Caracteristicas={^idx^=1 ^class^="SecCaracteristicas" ^1^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES CONSUMIDAS" Valor="0"} ^2^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE ALTA" Valor="20130822"} ^3^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES CONTRATADAS" Valor="100"} ^4^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES DISPONIBLES" Valor="100"} ^5^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE BAJA" Valor="20130922"} ^6^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE RESETEO" Valor="20130923"} ^7^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE

CADUCIDAD" Valor="20130923"}}}}} ^2^={^idx^=2 ^class^="Promociones" IdPromocion="653" Nombre="Bono SMS consulta Saldo" SistemaMaestro="FACTPRE" Estado="ACTIVA" FechaActivacion="19000101" ParametrosPromocion={^idx^=1 ^class^="SecParametrosPromocion" ^1^={^idx^=2 ^class^="ParametrosPromocion" AtributosDinamicos={^idx^=2 ^class^="AtributosDinamicos" Nombre="PRODUCTO" Valor="SMS"} Caracteristicas={^idx^=1 ^class^="SecCaracteristicas" ^1^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES CONSUMIDAS" Valor="0"} ^2^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE ALTA" Valor="20121102"} ^3^={^idx^=2 ^class^="Caracteristicas" Nombre="VALOR" Valor="POS"} ^4^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES CONTRATADAS" Valor="5"} ^5^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES DISPONIBLES" Valor="5"} ^6^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE BAJA" Valor="99991231"} ^7^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE RESETEO" Valor="99991231"} ^8^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE CADUCIDAD" Valor="99991231"}}}}} ^3^={^idx^=2 ^class^="Promociones" IdPromocion="DATAOCIO" Nombre="OCC no configurada" SistemaMaestro="FACTPRE" Estado="ACTIVA" FechaActivacion="20111207" ParametrosPromocion={^idx^=1 ^class^="SecParametrosPromocion" ^1^={^idx^=2 ^class^="ParametrosPromocion" AtributosDinamicos={^idx^=2 ^class^="AtributosDinamicos" Nombre="PRODUCTO" Valor="MENSAJES"} Caracteristicas={^idx^=1 ^class^="SecCaracteristicas" ^1^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES CONSUMIDAS" Valor=""} ^2^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE ALTA" Valor="20111207"} ^3^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES CONTRATADAS" Valor=""} ^4^={^idx^=2 ^class^="Caracteristicas" Nombre="UNIDADES DISPONIBLES" Valor=""} ^5^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE BAJA" Valor="99991231"} ^6^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE RESETEO" Valor=""} ^7^={^idx^=2 ^class^="Caracteristicas" Nombre="FECHA DE CADUCIDAD" Valor=""}}}}}} SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:58.064000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z}}

MIDDLEWARE continua preguntando a RESER:

2013-08-22 12:40:58 (2013-08-22 11:40:58.834690000Z): subject=MOVILES.INT.**PETICION.RESER.88003**, reply=MOVILES.INT.RESPUESTA.MDW.88003.09020, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/RESER/C_88003" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/RESER" 2="/tibco/public/sequence/ae/MOVILES/OBJETOS_NEGOCIO/RESER"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="C_88003_Peticion" SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:58.832000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z

TelefonoBasico={^idx^=1 ^class^="TelefonoBasico" MSISDN="111111115"} ListaParametros={^idx^=2 ^class^="SecListaParametros" ^1^={^idx^=1 ^class^="ListaParametros" PARAMETRO="infoSVA"}}}}

RESER envía su respuesta a MIDDLEWARE:

2013-08-22 12:40:58 (2013-08-22 11:40:58.874004000Z): subject=MOVILES.INT.**RESPUESTA.MDW.88003.09020**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 SecurityPluginJava={SecurityPluginJava="RV_PUBLISHER_CONSULTA_REPOSITORIOEndPoint OUT"} ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/RESER/C_88003" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/RESER"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="C_88003_Respuesta" DatosXML={^idx^=1 ^class^="DatosXML" CadenaXML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Linea>

<MSISDN>111111115</MSISDN>

<TIPO_LINEA>PREPAGO</TIPO_LINEA>

<TIPO_CLIENTE>PREPAGO</TIPO_CLIENTE>

<ServiciosLinea>

<RespuestaServicios>

<ID_SERVICIO_PROV>BLACKBERRY</ID_SERVICIO_PROV>

<SERVICIO_PROV>BLACKBERRY</SERVICIO_PROV>

<ESTADO>AC</ESTADO>

<FECHA_ALTA>14/08/2012 10:13:36</FECHA_ALTA>

<RespuestaParametros>

<PARAMETRO>SRV_BLACKBERRY</PARAMETRO>

<VALOR>PROFESIONAL_FD</VALOR>

</RespuestaParametros>

</RespuestaServicios>

<RespuestaServicios>

<ID_SERVICIO_PROV>2943</ID_SERVICIO_PROV>

<SERVICIO_PROV>BACKUPCONT</SERVICIO_PROV>

<ESTADO>PV</ESTADO>

<FECHA_ALTA>19/10/2012 11:13:01</FECHA_ALTA>

<RespuestaParametros>

<PARAMETRO>IDUSER_VOX</PARAMETRO>

<VALOR>PV_5420009</VALOR>

</RespuestaParametros>

</RespuestaServicios>

</ServiciosLinea>

</Linea>

</XML_Respuesta>

"} SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW.AdaptadorRESER" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:58.838000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z}}

Luego, MIDDLEWARE pregunta a FACTPOS:

2013-08-22 12:40:58 (2013-08-22 11:40:58.882550000Z): subject=MOVILES.INT.**PETICION.FACTPOS.03030**, reply=MOVILES.INT.RESPUESTA.MDW.03030.09020, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03030" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"}

^data^={^class^="C_03030_Peticion" SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW.AdaptadorRESER.MDW" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:58.879000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z DatosCliente={^idx^=1 ^class^="DatosCliente" CustomerId=""} MsisdnAmpliado={^idx^=1 ^class^="MsisdnAmpliado" Msisdn="111111115"} IntervaloFechas={^idx^=1 ^class^="IntervaloFechas" FechaInicio="20130822" FechaFin="20130822" FlagNoOCC="X"} Sistema={^idx^=1 ^class^="Sistema" Sistema="WEB"} Flag={^idx^=1 ^class^="Flag" Flag="N"}}}

FACTPOS contesta a MIDDLEWARE:

2013-08-22 12:41:01 (2013-08-22 11:41:01.087997000Z): subject=MOVILES.INT.**RESPUESTA.MDW.03030.09020**, message={^type^=1 ^pfmt^=10 ^ver^=30 ^encoding^=1 SecurityPlugin={SecurityPlugin="RV_PUB_03030_ConsOCCEndPoint"} ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPOSConsultas/C_03030" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas" 2="/tibco/public/sequence/ae/MOVILES/OBJETOS_NEGOCIO/FACTPOSConsultas"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="C_03030_Respuesta" DatosCliente={^class^="DatosCliente" ^idx^=1 CustomerId=""} MsisdnAmpliado={^class^="MsisdnAmpliado" ^idx^=1 Msisdn="111111115"} IntervaloFechas={^class^="IntervaloFechas" ^idx^=1 FechaInicio="20130822" FechaFin="20130822" FlagNoOCC="X"} OCC={^class^="SecOCC" ^idx^=2 ^1^={^class^="OCC" ^idx^=1 IdOccCrm="1809" NombreOcc="Bono Tarifa 10" Estado="ACTIVA" FechaActivacion="20120119" SistemaMaestro="FACTPOS" ParametrosPromo={^class^="SecParametrosPromo" ^idx^=2 ^1^={^class^="ParametrosPromo" ^idx^=1 NombreAtributo="CODIGO FACTPOS" ValorAtributo="23977183_FUIMP_259" Caracteristicas={^class^="SecCaracteristicas" ^idx^=2}} ^2^={^class^="ParametrosPromo" ^idx^=1 NombreAtributo="PRODUCTO" ValorAtributo="MINUTOS" Caracteristicas={^class^="SecCaracteristicas" ^idx^=2 ^1^={^class^="Caracteristicas" ^idx^=1 Nombre="UNIDADES CONSUMIDAS" Valor="0"} ^2^={^class^="Caracteristicas" ^idx^=1 Nombre="UNIDADES CONTRATADAS" Valor="300"} ^3^={^class^="Caracteristicas" ^idx^=1 Nombre="UNIDADES DISPONIBLES" Valor="300"}}} ^3^={^class^="ParametrosPromo" ^idx^=1 NombreAtributo="TARIFA" ValorAtributo="Tarifa 10" Caracteristicas={^class^="SecCaracteristicas" ^idx^=2}}}} ^2^={^class^="OCC" ^idx^=1 IdOccCrm="2923" NombreOcc="Free Friends" Estado="ACTIVA" FechaActivacion="NULL" SistemaMaestro="FACTPOS" ParametrosPromo={^class^="SecParametrosPromo" ^idx^=2 ^1^={^class^="ParametrosPromo" ^idx^=1 NombreAtributo="CODIGO FACTPOS" ValorAtributo="23977183_MGM2" Caracteristicas={^class^="SecCaracteristicas" ^idx^=2}} ^2^={^class^="ParametrosPromo" ^idx^=1 NombreAtributo="PRODUCTO" ValorAtributo="MINUTOS" Caracteristicas={^class^="SecCaracteristicas" ^idx^=2 ^1^={^class^="Caracteristicas" ^idx^=1 Nombre="UNIDADES CONSUMIDAS" Valor="0"} ^2^={^class^="Caracteristicas" ^idx^=1 Nombre="UNIDADES CONTRATADAS" Valor="0"} ^3^={^class^="Caracteristicas" ^idx^=1 Nombre="UNIDADES DISPONIBLES" Valor="0"}}} ^3^={^class^="ParametrosPromo" ^idx^=1 NombreAtributo="NUMERO FRECUENTE" ValorAtributo="111111110" Caracteristicas={^class^="SecCaracteristicas" ^idx^=2

^1^={^class^="Caracteristicas" ^idx^=1 Nombre="ROL" Valor="P"} ^2^={^class^="Caracteristicas" ^idx^=1 Nombre="FECHA DE ALTA" Valor="20120427"} ^3^={^class^="Caracteristicas" ^idx^=1 Nombre="TIPOLOGIA" Valor="telefonia"}}}}}} SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW.AdaptadorRESER.MDW.FACTPOS" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:40:58.887192000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z}}

Por último, MIDDLEWARE monta la respuesta completa con todas las respuestas que ha recibido y se la envía al sistema origen WEB. No incluyo el log por completo porque es demasiado extenso, así que solo incluyo la información devuelta que nos interesa ver, que es la promoción New TV, con su código de promoción 2221:

2013-08-22 12:41:01 (2013-08-22 11:41:01.101737000Z): subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09020**.weblogi1.3744792238.525, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW.AdaptadorRESER.MDW.FACTPOS.MDW_RESP" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 11:41:01.096000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 11:40:57Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Respuesta>

<IdCatalogoCRM>2221</IdCatalogoCRM>

<Nombre>New TV</Nombre>

<SistemaMaestro>FACTPRE</SistemaMaestro>

<Estado>ACTIVA</Estado>

<FechaActivacion>20130822</FechaActivacion>

<ParametrosPromo>
<Nombre>PRODUCTO</Nombre>
<Valor>TELEVISION</Valor>
<Caracteristicas>
<Nombre>UNIDADES CONSUMIDAS</Nombre>
<Valor>0</Valor>
</Caracteristicas>
<Caracteristicas>
<Nombre>FECHA DE ALTA</Nombre>
<Valor>20130822</Valor>
</Caracteristicas>
<Caracteristicas>
<Nombre>UNIDADES CONTRATADAS</Nombre>
<Valor>100</Valor>
</Caracteristicas>
<Caracteristicas>
<Nombre>UNIDADES DISPONIBLES</Nombre>
<Valor>100</Valor>
</Caracteristicas>
<Caracteristicas>
<Nombre>FECHA DE BAJA</Nombre>
<Valor>20130922</Valor>
</Caracteristicas>
<Caracteristicas>

<Nombre>FECHA DE RESETEO</Nombre>

<Valor>20130923</Valor>

</Caracteristicas>

<Caracteristicas>

<Nombre>FECHA DE CADUCIDAD</Nombre>

<Valor>20130923</Valor>

</Caracteristicas>

</ParametrosPromo>

</Respuesta>

7.8 Caso 10

La petición de Re-Alta de la promoción New TV se lanza como si nada, exactamente igual que en el caso del alta:

2013-08-22 12:59:36 (2013-08-22 11:59:36.422279000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="SEC_RV_PUBLICADOR_ACCMDW OUT"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw"} ^data^={_data={user="weblogi1" _data={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML="<XML_Peticion>

<Promocion>

<IdPeticion>201211191759360390-111111115</IdPeticion>

<IdPromo>2221</IdPromo>

<MSISDN>111111115</MSISDN>

<Accion>A</Accion>

<SistemaOrigen>WEB</SistemaOrigen>

</Promocion>

</XML_Peticion>"} SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:59:36.405000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:59:36.399000000Z} cert=130 sig=[46 opaque bytes]} _acld="INT.SECURITY"}}

MIDDLEWARE redirige el alta hacia FACTPRE, que es el maestro. Y sucede como con el fragmento anterior, hasta ahora es igual que en el caso del alta normal:

2013-08-22 12:59:36 (2013-08-22 11:59:36.460774000Z): subject=MOVILES.INT.**PETICION.FACTPRE_CNT.10001**, reply=MOVILES.INT.RESPUESTA.MDW.10001.09030, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10001" 1="/tibco/public/class/ae/MOVILES/OBJETOS_NEGOCIO/FACTPRE_CNT"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777348"} ^data^={_data={user="tibco_des" _data={^class^="C_10001_Peticion" SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:59:36.448000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:59:36Z DatosPromocion={^idx^=1 ^class^="DatosPromocion" IdPeticion="201211191759360390-111111115" SistemaOrigen="WEB" Accion="A" MSISDN="111111115" IdPromocion="2221"}} cert=126 sig=[46 opaque bytes]} _acld="INT.SECURITY"}}

Aquí es donde está la clave de esta prueba, el sistema FACTPRE responde a MIDDLEWARE indicando que se ha producido un error:

2013-08-22 12:59:36 (2013-08-22 11:59:36.738685000Z): subject=MOVILES.INT.**RESPUESTA.MDW.10001.09030**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/FACTPRE_CNT/C_10001"} SecurityPlugin={SecurityPlugin="RV_PUBLISHER_ALTA_BAJA_MODIF_PROMOCIONEndPoint OUT"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777348"} ^data^={^class^="C_10001_Respuesta" CabeceraError={^class^="**CabeceraError" ^idx^=1 Codigo="OC1562" Sistema="FACTPRE" Tipo="Promociones" Descripcion="Error al**

ejecutar alta de promoción: El contrato tiene la promoción activa" TimeStamp=2013-08-22 11:59:36.467000000Z} SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:59:36.467000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:59:36Z}}

MIDDLEWARE monta la respuesta en un XML y se la envía a WEB, quien muestra el error obtenido por pantalla:

2013-08-22 12:59:36 (2013-08-22 11:59:36.747808000Z): subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.weblogi1.6728595321.541, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="ANUUTEZd4NlSJEwKf4zzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777348"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.6728595321.191112055936.5962079691552741798" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW_RESP" Estado=0 FiltroMDW="weblogi1.6728595321.541" TimeStamp=2013-08-22 11:59:36.743000000Z IdAux="6728595321.5962079691552741798.541" TimeOut=60000 TimeStampOrigen=2013-08-22 11:59:36Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Promocion>

<IdPeticion>201211191759360390-111111115</IdPeticion>

<IdPromo>2221</IdPromo>

</Promocion>

<Error>

<Codigo>OC1562</Codigo>

<Descripcion>Error al ejecutar alta de promoción: El contrato tiene la promoción activa</Descripcion>

<Tipo>Promociones</Tipo>

<Sistema>FACTPRE</Sistema>

<TimeStamp>2013-08-22 11:59:36</TimeStamp>

</Error>

</XML_Respuesta>"} CabeceraError={^idx^=1 ^class^="CabeceraError" Codigo="OC1562" Sistema="FACTPRE" Tipo="Promociones" Descripcion="Error al ejecutar alta de promoción: El contrato tiene la promoción activa" TimeStamp=2013-08-22 11:59:36Z}}}

7.9 Caso 12

Es la consulta de la promoción New TV pero con consumo, y dado que es exactamente igual que la prueba 11 pero con los parámetros del consumo (Unidades Consumidas, Contratadas y Disponibles) indicando valores diferentes, solo muestro la traza que monta MIDDLEWARE para el sistema origen WEB una vez ha recibido la respuesta de todos los sistemas:

2013-08-22 13:15:32 (2013-08-22 12:15:32.101737000Z):
subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09020**.weblogi1.3744792238.525, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="o1xw57VW4MhBSkwKeUzzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09020/PC_Flujo_09020.process.Job-777333"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.3744792238.191112054057.-3014809252921130290" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW.AdaptadorRESER.MDW.FACTPOS.MDW_RESP" Estado=0 FiltroMDW="weblogi1.3744792238.525" TimeStamp=2013-08-22 12:15:32.096000000Z IdAux="3744792238.-3014809252921130290.525" TimeOut=15000 TimeStampOrigen=2013-08-22 12:15:25Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Respuesta>

<IdCatalogoCRM>2221</IdCatalogoCRM>

<Nombre>New TV</Nombre>

<SistemaMaestro>FACTPRE</SistemaMaestro>

<Estado>ACTIVA</Estado>

<FechaActivacion>20130822</FechaActivacion>

<ParametrosPromo>

<Nombre>PRODUCTO</Nombre>

<Valor>TELEVISION</Valor>

<Caracteristicas>

<Nombre>UNIDADES CONSUMIDAS</Nombre>

<Valor>50</Valor>

</Caracteristicas>

<Caracteristicas>

<Nombre>FECHA DE ALTA</Nombre>

<Valor>20130822</Valor>

</Caracteristicas>

<Caracteristicas>

<Nombre>UNIDADES CONTRATADAS</Nombre>

<Valor>100</Valor>

</Caracteristicas>

<Caracteristicas>

<Nombre>UNIDADES DISPONIBLES</Nombre>

<Valor>50</Valor>

</Caracteristicas>

<Caracteristicas>

<Nombre>FECHA DE BAJA</Nombre>

<Valor>20130922</Valor>

</Caracteristicas>
<Caracteristicas>
<Nombre>FECHA DE RESETEO</Nombre>
<Valor>20130923</Valor>
</Caracteristicas>
<Caracteristicas>
<Nombre>FECHA DE CADUCIDAD</Nombre>
<Valor>20130923</Valor>
</Caracteristicas>
</ParametrosPromo>
</Respuesta>

7.10 Caso 13

Para ejecutar una baja de una promoción, también se efectúa mediante una operación 09030, solo que en los parámetros de la petición se informa de que la acción es una baja con una D. Además, igual que para el alta, como parámetros viajan el MSISDN, el código de la promoción que se quiere dar de baja y el sistema origen que la ejecuta:

2013-08-22 15:44:32 (2013-08-22 14:44:32.187148000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} SecurityPlugin={SecurityPlugin="SEC_RV_PUBLICADOR_ACCMDW OUT"} ^tracking^={^id^="Se6gmAyd4MuG9-wKeczzzS2Uzzw"} ^data^={_data={user="weblogi1" _data={^class^="S_Peticion" XML_Peticion={^idx^=1 ^class^="CXML" XML="<XML_Peticion>

<Promocion>

<IdPeticion>201211191744320151-111111115</IdPeticion>

<IdPromo>2221</IdPromo>

<MSISDN>111111115</MSISDN>

<Accion>D</Accion>

<SistemaOrigen>WEB</SistemaOrigen>

</Promocion>

</XML_Peticion>"} SourceMsgId="weblogi1.0876472996.191112054432.-5792066556398641826" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware" Estado=0 FiltroMDW="weblogi1.0876472996.528" TimeStamp=2013-08-22 14:44:32.172000000Z IdAux="0876472996.-5792066556398641826.528" TimeOut=60000 TimeStampOrigen=2013-08-22 14:44:32.165000000Z} cert=130 sig=[47 opaque bytes]} _acld="INT.SECURITY"}}

Para evitar un exceso de log, en este caso voy a omitir la petición que MIDDLEWARE envía a FACTPRE, junto con su respuesta, y voy directamente a la respuesta que monta MIDDLEWARE para el sistema origen WEB. En la respuesta, se informa del MSISDN y del código de la promoción, esto significa que ha ido OK y es lo que pinta WEB por pantalla:

2013-08-22 15:44:33 (2013-08-22 14:44:33.221947000Z): subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.weblogi1.0876472996.528, message={^pfmt^=10 ^ver^=30 ^type^=1 ^encoding^=1 ^prefixList^={default="/tibco/public/class/ae/MOVILES/SERVICIOS/COMUN" 1="/tibco/public/class/ae/MOVILES/COMUN"} ^tracking^={^id^="Se6gmAyd4MuG9-wKeczzzS2Uzzw" ^1^[1]="BW.Promociones-Process_Archive_tibco_des.Procesos_MDW/09030/PC_Flujo_09030.process.Job-777336"} ^data^={^class^="S_Respuesta" SourceMsgId="weblogi1.0876472996.191112054432.-5792066556398641826" SourceHost="weblogi1" Version="AE4" AppName="libAccMiddleware.MDW.FACTPRE_CNT.MDW_RESP" Estado=0 FiltroMDW="weblogi1.0876472996.528" TimeStamp=2013-08-22 14:44:33.218000000Z IdAux="0876472996.-5792066556398641826.528" TimeOut=60000 TimeStampOrigen=2013-08-22 14:44:32Z XML_Respuesta={^idx^=1 ^class^="CXML" XML="<?xml version="1.0" encoding="ISO-8859-1"?>

<XML_Respuesta>

<Promocion>

<IdPeticion>201211191744320151-111111115</IdPeticion>

<IdPromo>2221</IdPromo>
</Promocion>
</XML_Respuesta>"}}}

7.11 Caso 14

El caso 14 consiste en la Re-Baja de la promoción. El log es igual que el de la prueba de la Re-Alta del caso de prueba 10, por lo que aquí solo añado logs de la petición de la parte que incluye los parámetros. Y el XML de la respuesta final que llega a WEB:

<IdPeticion>201211191744320151-111111115</IdPeticion>
<IdPromo>2221</IdPromo>
<MSISDN>111111115</MSISDN>
<Accion>D</Accion>
<SistemaOrigen>WEB</SistemaOrigen>

En el XML de la respuesta puede observarse el error que es devuelto por el sistema FACTPRE al no poder realizar la baja solicitada:

<Promocion>
<IdPeticion>201211191744320151-111111115</IdPeticion>
<IdPromo>2221</IdPromo>
</Promocion>
<Error>
<Codigo>OC1565</Codigo>

<Descripcion>Error al ejecutar baja de promoción: El contrato no tiene activa la promoción</Descripcion>
<Tipo>Promociones</Tipo>
<Sistema>FACTPRE</Sistema>
<TimeStamp>2013-08-22 15:01:03</TimeStamp>
</Error>

7.12 Caso 6, Ejecución 2

La prueba es bastante extensa en cuanto a log se refiere porque son 3 peticiones, 2 con timeout y una con respuesta del sistema destino. Es por ello que considero oportuno mostrar solo las cabeceras de cada parte del log. La prueba da comienzo con una petición de una 09030 para un alta de una promoción cualquiera, por lo que vamos a obviar los datos de entrada:

2013-08-26 09:32:36 (2013-08-26 08:32:36.817345000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message[…]

Se redirige la petición a FACTPOS:

2013-08-26 09:32:36 (2013-08-26 08:32:36.846058000Z): subject=MOVILES.INT.**PETICION.FACTPOS.26069**, […]

Tras 30 segundos en los que MIDDLEWARE no recibe respuesta alguna de FACTPOS, comunica el timeout al sistema origen WEB. Hasta aquí es lo mismo que en la primera ejecución de la prueba, cuando se detectó la incidencia:

2013-08-26 **09:33:06** (2013-08-26 08:33:06.854754000Z): subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.[…]
<XML_Respuesta>
<Error>

<Codigo>MDW_ERR-093005</Codigo>
<Descripcion>Se ha producido un TIMEOUT en el sistema FACTPOS(26069)</Descripcion>
<Tipo>MDW</Tipo>
<Sistema>MDW</Sistema>
<TimeStamp>2013-08-26T09:33:06.849+01:00</TimeStamp>
<IdHost>tibco_des</IdHost>
</Error>[…]

Puede verse como ahora sí se produce el reenvío de la petición de nuevo. Transcurridos 5 minutos desde que WEB recibe el timeout, se relanza la operación, exactamente igual que la anterior, con los mismos parámetros:

2013-08-26 **09:38:57** (2013-08-26 08:38:57.708998000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message={^pfmt^=10 ^ver^=30 […]

Se pregunta a FACTPOS:

2013-08-26 09:38:57 (2013-08-26 08:38:57.753298000Z): subject=MOVILES.INT.PETICION.**FACTPOS.26069**, reply=[…]

Y se produce de nuevo el timeout:

2013-08-26 **09:39:27** (2013-08-26 08:39:27.761094000Z): subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.[…]
<Error>
<Codigo>MDW_ERR-093005</Codigo>
<Descripcion>Se ha producido un TIMEOUT en el sistema FACTPOS(26069)</Descripcion>
<Tipo>MDW</Tipo>
<Sistema>MDW</Sistema>

<TimeStamp>2013-08-26T09:39:27.756+01:00</TimeStamp>
<IdHost>tibco_des</IdHost>
</Error>[…]

A los cinco minutos de producirse el timeout se envía por última vez la petición de la 09030. Esta vez sí levantamos el adaptador pesado de FACTPOS para que el sistema conteste:

2013-08-26 **09:45:01** (2013-08-26 08:45:01.570702000Z): subject=MOVILES.INT.**PETICION.MDW.09030**, message[…]

MIDDLEWARE envía la petición a FACTPOS:

2013-08-26 09:45:01 (2013-08-26 08:45:01.600244000Z): subject=MOVILES.INT.**PETICION.FACTPOS.26069**, […]

Y esta vez sí contesta el sistema destino FACTPOS porque ahora sí le ha llegado la petición:

2013-08-26 09:45:06 (2013-08-26 08:45:06.592037000Z): subject=MOVILES.INT.**RESPUESTA.MDW.26069.09030**, message={^type^=1 […]

Por último, MIDDLEWARE monta la respuesta que recibe. Aunque se trate de un error por parte de FACTPOS, es un error lógico derivado de la promoción y que nada tiene que ver con la funcionalidad que estamos probando, que es el encolado y reenvío de la operación, lo cual sí se está realizando y por lo que damos el caso por OK:

2013-08-26 09:45:06 (2013-08-26 08:45:06.623604000Z):subject=MOVILES.INT.**RESPUESTA.libAccMiddleware.09030**.[…]
<XML_Respuesta>
<Promocion>
<IdPeticion>201211131607280492-111111111</IdPeticion>
<IdPromo>9162</IdPromo>

</Promocion>
<Error>
<Codigo>NL8073</Codigo>
<Descripcion>Error al ejecutar FACTPOS Desc: El contrato tiene un módulo upsize activo</Descripcion>
<Tipo>Adaptador</Tipo>
<Sistema>FACTPOS</Sistema>
<TimeStamp>2013-08-26T09:45:06</TimeStamp>
<IdHost>FACTPOSapr</IdHost>
</Error> […]

7.13 Caso 7

Dado que los logs son muy extensos, me limito a exponer las cabeceras de cada parte de los logs que al fin y al cabo es la información que realmente interesa.

En primer lugar, se envía la petición de cambio de tarifa con una 26032:

2013-08-26 09:56:45 (2013-08-26 08:56:45.656421000Z): subject=**MOVILES.INT.PETICION.MDW.26032** […]

MIDDLEWARE se lo transmite a FACTPOS:

2013-08-26 09:56:45 (2013-08-26 08:56:45.672212000Z): subject=**MOVILES.INT.PETICION.FACTPOS.26032**[…]

FACTPOS no contesta y MIDDLEWARE transmite el timeout a WEB:

2013-08-26 **09:57:45** (2013-08-26 08:57:45.688364000Z): subject=**MOVILES.INT.RESPUESTA.libAccMiddleware.26032**[…]

A los cinco minutos de haber recibido el timeout, tal y como se espera, WEB que ha encolado la petición de la 26032 la reenvía por primera vez:

2013-08-26 **10:02:57** (2013-08-26 09:02:57.329575000Z): subject=**MOVILES.INT.PETICION.MDW.26032**[…]

MIDDLEWARE transmite la petición a FACTPOS:

2013-08-26 10:02:57 (2013-08-26 09:02:57.346513000Z): subject=MOVILES**.INT.PETICION.FACTPOS.26032**[…]

FACTPOS una vez más no contesta porque seguimos con el adaptador pesado tirado y por tanto, no le están entrando peticiones. Así que MIDDLEWARE comunica el segundo timeout al sistema origen WEB:

2013-08-26 **10:03:39** (2013-08-26 09:03:39.015273000Z): subject=**MOVILES.INT.RESPUESTA.libAccMiddleware.26032**[…]

Se produce el segundo reintento, una vez más a los cinco minutos, WEB reenvía la petición 26032 encolada por el timeout. Y como es el segundo intento, acabamos de levantar el adaptador pesado, para que esta vez FACTPOS pueda contestar:

12-11-15 **10:09:07** (2013-08-26 09:09:07.763262000Z): subject=**MOVILES.INT.PETICION.MDW.26032** […]

MIDDLEWARE envía la petición a FACTPOS:

2013-08-26 10:09:07 (2013-08-26 09:09:07.780761000Z): subject=**MOVILES.INT.PETICION.FACTPOS.26032**[…]

En esta ocasión, FACTPOS sí recibe la petición y contesta a MIDDLEWARE:

2013-08-26 10:09:12 (2013-08-26 09:09:12.747574000Z): subject=**MOVILES.INT.RESPUESTA.MDW.26032.26032**[…]

MIDDLEWARE monta la respuesta recibida por FACTPOS y se la envía a WEB:

2013-08-26 10:09:12 (2013-08-26 09:09:12.753806000Z): subject=**MOVILES.INT.RESPUESTA.libAccMiddleware.26032**[…]

7.14 Caso 8

Idem que para el caso anterior, para no añadir demasiada información, me limito a las cabeceras de cada tramo. En primer lugar, se publica el cambio de servicios 26033 desde WEB:

2013-08-26 10:27:41 (2013-08-26 09:27:41.229213000Z): subject=MOVILES**.INT.PETICION.MDW.26033** […]

MIDDLEWARE envía la petición a FACTPOS:

2013-08-26 **10:27:41** (2013-08-26 09:27:41.269853000Z): subject=**MOVILES.INT.PETICION.FACTPOS.26033** […]

Como se trata de una operación punto a punto, pasados 6 minutos sin contestación de FACTPOS se considera que se ha producido un timeout porque en este tipo de operaciones, MIDDLEWARE no lo comunica vía log. Por tanto, 6 minutos después de no recibir respuesta, WEB reenvía la petición 26033 encolada:

2013-08-26 **10:33:45** (2013-08-26 09:33:45.346641000Z): subject=**MOVILES.INT.PETICION.MDW.26033** […]

Igual que antes, MIDDLEWARE se lo remite a FACTPOS:

2013-08-26 **10:33:45** (2013-08-26 09:33:45.444078000Z): subject=**MOVILES.INT.PETICION.FACTPOS.26033** […]

A los 6 minutos de no recibir respuesta, se da por hecho el timeout y se reenvía por segunda vez la petición de la 26033. Es ahora cuando levanto el adaptador pesado de servicios de FACTPOS:

2013-08-26 **10:39:51** (2013-08-26 09:39:51.836767000Z): subject=**MOVILES.INT.PETICION.MDW.26033** […]

MIDDLEWARE transmite la petición a FACTPOS:

2013-08-26 10:39:51 (2013-08-26 09:39:51.871756000Z): subject=**MOVILES.INT.PETICION.FACTPOS.26033** […]

Esta vez, al estar el adaptador pesado levantado, FACTPOS recibe la petición y puede contestar:

2013-08-26 10:39:54 (2013-08-26 09:39:54.859564000Z): subject=**MOVILES.INT.RESPUESTA.MDW.26033** […]

MIDDLEWARE recibe la contestación, la monta y se la envía en un XML al sistema origen WEB:

2013-08-26 10:39:54 (2013-08-26 09:39:54.871704000Z): subject=**MOVILES.INT.RESPUESTA.libAccMiddleware.26033** […]