1 Sistemas basados en la web

1.1 Introducción

En las décadas de los 50 y 60 del siglo XX poca gente era capaz de apreciar la importancia de los sistemas computerizados, y casi nadie era capaz de prever el impacto global que iban a tener el hardware y el software en cada uno de los aspectos de la sociedad en los tiempos que se avecinaban. La mayoría de las personas que comenzaban a trabajar en el campo de la informática en esos días se aventuraban de forma incierta en ese nuevo terreno, creando programas informáticos con una combinación de informalidad, urgencia, intuición y arte. Cuando las cosas iban bien, estos programas daban lugar a importantes avances. Pero las cosas no siempre iban bien. Los sistemas computerizados a veces fallaban; eran entregados tarde o no llegaban a entregarse; eran difíciles o imposibles de corregir, adaptar y mejorar de forma razonable. El enfoque que se seguía en esos tiempos pioneros se caracterizaba por la informalidad y estaba poco o nada estructurado.

Pero lo peor de todo es que esa forma de abordar el desarrollo de sistemas computerizados quedó fuertemente arraigada, haciendo pensar a muchos profesionales del sector que informalidad, urgencia, intuición y arte eran las fuerzas directrices de los desarrolladores informáticos. Al fin y al cabo, la informalidad conduce a un ambiente relajado de trabajo en el que cada cual hace lo que cree oportuno. La urgencia antepone la acción y una toma de decisiones pronta. La intuición es una cualidad intangible que permite que cada cual "sienta" cómo salir de situaciones complicadas. Y el arte conduce a una forma y una función estéticas y que complacen a aquel que se lo encuentra. ¿Qué podría haber de malo en este planteamiento tan idílico?

1.2 La web

Es perfectamente conocido por todos el enorme impacto que Internet y la web están teniendo en nuestras vidas. La web se ha convertido en una tecnología fundamental para los negocios, las comunicaciones, la educación, la ciencia, la ingeniería, el entretenimiento, el gobierno, la industria, la medicina, la política, etc.

En todas estas áreas y otras hay algo en común: es necesario un vehículo de reparto que tome la información en bruto asociada con el área de interés; la estructure de forma que resulte significativa; la empaquete en forma de presentación que resulte organizada, estética, ergonómica e interactiva (si es que es necesario); y la entregue a un navegador web de forma que se inicie una conversación.

Esta conversación puede ser pasiva o activa. En una conversación pasiva, el usuario selecciona la información que desea que le sea presentada, pero no tiene un control directo sobre su volumen, tipo o estructura. En una conversación activa, el usuario proporciona una serie de entradas de manera que la información que se le presenta se adapta a sus necesidades específicas.

El vehículo que adquiere la información, la estructura, la empaqueta para su presentación y la entrega es lo que se conoce como una aplicación web (*WebApp*). La combinación de una aplicación web con hardware cliente y servidor, sistemas operativos, software de red y navegadores es lo que se conoce como sistema basado en la web o sistema web a secas.

1.3 Aplicaciones web

Consideramos que una aplicación web es un programa al que se accede a través de una red, en lugar de residir en la memoria principal del ordenador, como es el caso de los programas convencionales. A menudo, las aplicaciones web se ejecutan dentro de un navegador web. No obstante, las aplicaciones web también pueden ser de tipo cliente, cuando una pequeña parte del programa se descarga al ordenador local del usuario pero el procesamiento se hace a través de la red en un servidor externo.

En los albores de la *world wide web* (primera mitad de los años 90), los "sitios web" consistían en poco más que un conjunto de ficheros de hipertexto enlazados que presentaban información mediante texto y gráficos simples. A medida que iba pasando el tiempo, el lenguaje HTML (*Hypertext Markup Language*) fue complementado mediante distintas herramientas y tecnologías (por ejemplo XML - *Extensible Markup Language* y Java), que permitían a los ingenieros web proporcionar capacidad de cómputo y contenidos tanto en la parte cliente como en la parte servidor. En la actualidad, las aplicaciones web han evolucionado hasta convertirse en sofisticadas herramientas informáticas que no sólo son capaces de proporcionar funcionalidad autónoma al usuario, sino que también se integran con bases de datos y aplicaciones externas.

1.3.1 Caso de estudio

CPI Corporation es una empresa (ficticia) que construye, vende y monitoriza sistemas de seguridad para hogares y pequeños negocios. CPI se ha dirigido a tu empresa de desarrollo de software porque no tiene presencia en la web y quiere lanzar un sitio web potente que coincida con el lanzamiento de una nueva línea de sensores de seguridad y un conjunto de innovadores servicios basados en la web. CPI quiere que tu empresa ayude en el desarrollo de la aplicación web, llamada *CasaSegura.com*.

Se te ha pedido que asistas a una reunión en la que se discuten las ideas básicas. En esta reunión has aprendido que CPI ha desarrollado un controlador de sensores compacto e inalámbrico que pretenden que se convierta en el elemento clave de esa nueva línea de sistemas de seguridad llamados *CasaSegura*.

Y así se va poniendo en marcha el proyecto. En estos momentos iniciales todavía hay cosas por definir y especificar, para luego poder implementarlas. La percepción del producto cambiará con el tiempo, y lo mismo ocurrirá con el sistema web en el que se apoyará. Pero eso es algo normal en esta primera fase. *CasaSegura* tiene el apoyo de la alta dirección de CPI (que perciben un gran beneficio potencial gracias a este nuevo producto).

1.3.2 Características de las aplicaciones web

¿Difieren los atributos de las aplicaciones web de los de las aplicaciones convencionales? Respecto a este punto existe cierto debate. Hay quienes argumentan que una aplicación web no es más que una aplicación cliente/servidor en la que se presta especial atención a una presentación estética, pero que a nivel de funcionalidad las aplicaciones web y las convencionales tienen los mismos atributos. Pero otros consideran que las aplicaciones web presentan todo un conjunto de características que las distinguen del software convencional. Los siguientes atributos están presentes en la mayoría de aplicaciones web.

- **Dependencia intensiva de la red**. Toda aplicación web reside en una red y debe servir a las necesidades de una comunidad de usuarios diversa. En el caso del producto llamado *CasaSegura*, muchas de las nuevas características implementadas por CPI serán iniciadas, controladas y/o monitorizadas a través de la web. La red permitirá la comunicación entre las características de tipo cliente de la aplicación web *CasaSegura.com* y los servidores de CPI.

- **Concurrencia**. Una aplicación web puede ser accedida de forma simultánea por un gran número de usuarios. En muchos casos, los patrones de uso pueden variar ampliamente de un usuario a otro. En algunos casos, las acciones de un usuario o de un grupo de usuarios pueden tener un impacto sobre las acciones de otros usuarios o la información que se presenta a otros usuarios. En el caso de *CasaSegura.com*, decenas de miles de hogares y negocios serán monitorizados concurrentemente, cientos o miles de usuarios pueden acceder a la aplicación web en cualquier momento, y docenas de técnicos de servicio pueden estar conectados también a la vez.

- **Impredecibilidad de la carga**. El número de usuarios de la aplicación web puede variar en varios órdenes de magnitud de un día a otro. En el caso de *CasaSegura.com*, el número de lugares monitorizados variará lentamente. Pero la aplicación web debe ser capaz de manejar un número indeterminado de eventos de forma simultánea (por ejemplo, alarma antirrobo, detección de incendios, detección de monóxido de carbono). Puede que el lunes se registren 10 eventos por hora; el martes se registren 100 eventos por hora y el miércoles (después de que la zona haya sufrido un apagón) pueden registrarse miles de eventos por minuto.

- **Rendimiento**. Si un usuario tiene que esperar mucho a una aplicación web (para acceder, para procesamiento de parte del servidor, para formateo y visualización de la información de parte del cliente), podría decidir abandonar dicha aplicación web y buscar otra más satisfactoria. En el caso de *CasaSegura.com*, el rendimiento es crítico ya que puede haber vidas humanas en juego. Una respuesta demasiado lenta de la aplicación web podría acabar incluso en una demanda por parte del usuario.

- **Disponibilidad**. Aunque no es realista esperar una disponibilidad del 100%, los usuarios de aplicaciones web populares a menudo demandan disponibilidad en cualquier momento. Una disponibilidad del 100% es el objetivo de *CasaSegura.com*, ya que se trata de un sistema para la seguridad de hogares y negocios, y esta aplicación web debe ser diseñada para alcanzar este ideal, o al menos lo más próximo posible.

- **Centrada en datos**. La función principal de muchas aplicaciones web es presentar contenidos de texto, gráficos, audio y vídeo al usuario final. Además, es habitual utilizar aplicaciones web para acceder a información almacenada en bases de datos que no son parte integral del entorno web (por ejemplo, aplicaciones de comercio electrónico o financieras). Todo esto se da en el caso de *CasaSegura.com*. La aplicación web debe acceder a una base de datos que contiene información acerca de cada cliente, la configuración del sistema que cada cliente tiene, sus requerimientos de monitorización, un registro de eventos y un registro de mantenimiento.

- **Importancia de los contenidos**. La calidad de los contenidos suele ser un aspecto que determina la calidad de una aplicación web. En el caso de *CasaSegura.com*, los usuarios serán normalmente personas sin conocimientos técnicos específicos que requerirán una presentación de los contenidos simple pero coherente.

- **Evolución continua**. A diferencia de las aplicaciones software convencionales, que evolucionan a lo largo de una serie de entregas planificadas y separadas temporalmente, las aplicaciones web evolucionan de forma continua. No es raro para algunas aplicaciones web (al menos, para su contenido) que se actualicen a cada minuto o que los contenidos a presentar se construyan de forma independiente para cada solicitud. Como veremos más alante, este tipo de situaciones se darán también en el caso de *CasaSegura.com*.

- **Inmediatez**. Esta característica se refiere a una necesidad imperiosa de lanzar al mercado un producto de forma rápida, y es algo que se da para muchos tipos de software, pero en el caso de las aplicaciones web puede reducirse de forma particularmente drástica a un *time-to-market* de semanas e incluso días. Los ingenieros web deben emplear métodos para planificar, analizar, diseñar, implementar y probar adaptados a las planificaciones "comprimidas" que requiere el desarrollo de aplicaciones web. En el caso de *CasaSegura.com*, la directiva de CPI tiene los objetivos de dar un empujón a los ingresos a corto plazo y de obtener beneficios significativos a medio plazo. En situaciones como esta, la aplicación web hace falta para ayer.

- **Integridad**. Puesto que las aplicaciones web están disponibles a través de redes, es difícil, si no imposible, limitar el conjunto de usuarios finales que pueden acceder a la aplicación web. Es necesario implementar medidas de seguridad efectivas y fuertes tanto en la infraestructura sobre la que se asienta una aplicación web como en la misma aplicación, para proteger los contenidos confidenciales y para proporcionar modos seguros de transmitir los datos. En el caso de *CasaSegura.com*, la información fluye tanto hacia como desde hogares y negocios, convirtiéndose en un jugoso potencial botín para delincuentes, y puesto que es el producto de una empresa especializada en seguridad, ¡más vale que sea segura!

- **Estética**. La apariencia suele ser un aspecto determinante en el nivel de aceptación de muchas aplicaciones web. Cuando una aplicación web ha sido diseñada para promocionar y vender productos o proporcionar servicios, con el objetivo de generar beneficios, la estética puede ser tan importante para alcanzar el éxito como el diseño técnico. En el caso de *CasaSegura.com*, la variedad de contenidos y funciones que proporcionará la aplicación web requiere que su presentación sea simple y elegante.

1.3.3 Tipos de aplicaciones web

A medida que se van sucediendo diferentes reuniones, los ingenieros web van entendiendo cada vez mejor la visión de *CasaSegura.com* que tienen los directivos y el personal técnico de CPI. Se va haciendo patente que *CasaSegura.com* será una aplicación web grande y compleja. Aunque en las primeras fases aún no se han hecho compromisos en firme, parece que se requerirá la implementación de las siguientes características (relativas a contenido y funcionalidad):

- Información sobre CPI, sus productos y sus miembros.
- Especificaciones de todos los componentes hardware de seguridad, incluyendo imágenes, descripciones técnicas, instrucciones de instalación, precios y otra información de interés.
- Asistencia sobre el diseño de sistemas de seguridad que permita a los clientes especificar un espacio (por ejemplo, estancias, puertas, ventanas) y entonces obtener una ayuda semiautomatizada del trazado de un sistema de seguridad para dicho espacio.
- Funcionalidad de comercio electrónico que permita a un cliente comprar hardware de seguridad y contratar servicios de monitorización.
- Capacidad por parte de los clientes de monitorización en tiempo real de espacios por vídeo.
- Capacidad de acceso a los clientes basada en cuentas de usuario.
- Capacidad de acceso para el personal de servicio técnico de CPI.

Además, CPI quiere abandonar la estrategia de ventas tradicional "física" (con vendedores y tiendas físicas) y cambiar a un paradigma más tecnológico vendiendo exclusivamente a través de la web.

Pero CPI no tiene en la actualidad una presencia significativa en la web, y carece de unos requisitos por escrito de lo que se supone que *CasaSegura.com* debe llegar a ser. Ni siquiera tiene un conocimiento particularmente bueno de las posibilidades que ofrece la web. La aplicación web *CasaSegura.com* evolucionará a través de una serie de etapas a las que llamaremos incrementos.

A continuación, presentamos diferentes tipos de aplicaciones web que pueden distinguirse. En ocasiones, una misma aplicación web, a medida que va evolucionando, puede ir convirtiéndose de un tipo en otro e ir incorporando características de varias de estas categorías, como ocurrirá con el caso de *CasaSegura.com*.

- **Aplicaciones web informativas**. Se toma la decisión de construir una página de inicio y algunas páginas complementarias que describan CPI y sus productos y servicios. Ésta sería una aplicación web informativa, con contenidos de solo lectura y enlaces para una navegación simple.

- **Aplicaciones web de descarga**. Unas semanas más tarde, se empieza a añadir contenido que describe los sensores de *CasaSegura* y otro hardware de seguridad. CPI proporciona ficheros *pdf* con las especificaciones de cada uno de estos componentes. Se añade la posibilidad de que los visitantes de *CasaSegura.com* descarguen estos ficheros. Ahora la aplicación web incorpora capacidades informativas y de descarga.

- **Aplicaciones web personalizables**. CPI tiene cuatro tipos de usuarios finales potenciales: propietarios de hogares, propietarios de pequeños negocios, personal de CPI de atención al cliente y personal técnico de CPI. Se quiere adaptar el contenido presentado por la aplicación web a las necesidades específicas de cada tipo de usuario. Se lleva a cabo entonces una revisión general de *CasaSegura.com*, convirtiéndola en una aplicación web personalizable para cada usuario.

- **Aplicaciones web interactivas**. El tráfico se incrementa rápidamente y en seguida hay cientos de visitantes al día. Se desea crear un sentido de comunidad entre los visitantes, un lugar donde las personas puedan charlar, preguntar y responder preguntas, dar opiniones acerca de productos, etc. Se decide implementar una extensión de *CasaSegura.com* consistente en un canal de charla o grupo de discusión (*chat*). Ahora la aplicación web cuenta con un componente interactivo.

- **Aplicaciones web con entrada del usuario**. La directiva de CPI desea dejar atrás las llamadas telefónicas y mensajes de correo-e de los clientes potenciales para solicitar presupuestos. Se implementa un sistema de entrada basado en formularios de forma que toda petición de presupuesto esté organizada de una forma homogénea, coherente y predecible. Para elaborar los presupuestos será necesario otro componente software, pero de momento, al menos, no será necesario transcribir una variedad de entradas heterogéneas procedentes de diversas fuentes, por diferentes canales y con formatos variopintos.

- **Aplicaciones web transaccionales**. La entrada basada en formularios para los presupuestos está bien, pero los directivos de CPI se dan cuenta en seguida de que todo el proceso de elaboración del presupuesto puede automatizarse. Proporcionan una serie de fórmulas y algoritmos para calcular las tarifas del hardware de seguridad y de los servicios de monitorización a partir de los datos de entrada proporcionados a través de los formularios. Así el usuario puede obtener de manera inmediata un presupuesto según los datos que haya introducido, ocurriendo así una transacción entre el usuario y la aplicación web.

- **Aplicaciones web orientadas a servicios**. Ahora todo está listo para poder proporcionar una completa capacidad de asistencia al diseño, un sistema de diseño de sistemas de seguridad para hogares y pequeños negocios asistido por computadora proporcionado por una aplicación web. El usuario introduce una descripción gráfica de un espacio y la aplicación web le ayuda a diseñar un sistema de seguridad para ese espacio. Este servicio podría proporcionar una ventaja competitiva a CPI y llevar de forma directa a incrementar sus ingresos. Además, este componente realzaría la percepción por parte de los clientes potenciales del nivel de sofisticación de CPI y de los productos *CasaSegura*.

- **Portales**. El tiempo pasa y el esfuerzo llevado a cabo se ve reflejado en los miles de personas que visitan *CasaSegura.com* cada día. Cada día, el personal de CPI recibe centenares de consultas relativas a seguridad, y no tiene tiempo de responderlas todas. Para resolver este problema, se comienza por proporcionar enlaces a los sitios web que tienen las respuestas a esas cuestiones. Así, una parte de *CasaSegura.com* dirige a los usuarios a una amplia variedad de diferentes fuentes de información, adquiriendo así características de un portal.

- **Acceso a bases de datos**. La línea de productos *CasaSegura* y sus clientes han aumentado de forma espectacular, lo que lleva a la necesidad de desarrollar tres nuevas bases de datos: 1) todos los productos *CasaSegura* junto con sus especificaciones técnicas, precios (dependiendo del tipo de cliente), instrucciones de montaje e información de disponibilidad y entrega; 2) toda la información sobre los clientes; y 3) toda la información de monitorización. Estas bases de datos pueden ser consultadas mediante elementos de entrada de datos de usuario de la aplicación web (formularios).

- **Almacén de datos (*data warehousing*)**. CPI está extendiéndose rápidamente al mercado internacional. Para satisfacer las necesidades de muchos países, es necesario considerar información sobre normativas de construcción, proveedores, instaladores... específicos de dichos países. Es por tanto necesario poder acceder a diversas bases de datos y extraer información que será interesante para los clientes internacionales. Por este motivo, se incorpora un componente de *data warehousing* a *CasaSegura.com*.

2 Ingeniería web

2.1 Introducción

A la hora de desarrollar una aplicación web, no es recomendable aplicar el enfoque de la vieja escuela basado en los planteamientos de informalidad, urgencia, intuición y arte que vimos en la sección 1.1. Éste podría ser un enfoque admisible si el proyecto de desarrollo fuera una aventura estrictamente personal, en la que el desarrollador es la única persona involucrada, pero estas circunstancias se dan raramente. Normalmente habrá unos clientes, que exigen una aplicación web que satisfaga sus necesidades, que sea fiable, extensible y funcional; o, cuando se trabaja para una empresa o una organización gubernamental o de otro tipo, habrá una dirección, que necesitará una aplicación web que juegue un determinado papel dentro de una estrategia de negocio más amplia. Puede que haya otros colaboradores que estén esperando que se les entregue la aplicación web dentro de un plazo establecido para integrarla con otros sistemas y procesos que estén desarrollando. Normalmente existirá un grupo de personas que necesita una aplicación web que funcione, y estas personas no querrán correr grandes riesgos ni estar expuestos a incertidumbres ni imprevistos.

Hay una alternativa a ese enfoque de la vieja escuela, una alternativa que reduce (pero no elimina del todo) el riesgo, ofreciendo una mayor probabilidad de éxito cuando se persigue el desarrollo de aplicaciones web. Esta alternativa es la ingeniería web.

2.2 Ingeniería web

La ingeniería web propone un marco de trabajo ágil pero estructurado para la construcción de aplicaciones web con niveles de calidad industrial.

2.2.1 Estrategias ágiles

Los ingenieros web deben entender que en la actualidad las empresas deben ser adaptables, ya que las estrategias y las reglas de negocio cambian con rapidez, la dirección requiere un grado de reacción casi instantáneo (habitualmente, hasta extremos no razonables) y los participantes en el negocio (*stakeholders*)[1] cambian con frecuencia de opinión pero a la vez siguen exigiendo una pronta entrega. Lo que preocupa a los clientes acerca de una aplicación web es que les sea entregada cuando la necesitan, y no el trabajo que pueda ser necesario para ello. En este tipo de escenario, un equipo de ingeniería web debe hacer énfasis en la agilidad.

Un equipo ágil es aquel capaz de responder de forma adecuada a los cambios. El concepto de cambio es inherente al desarrollo de software. Cambios en el software que se está desarrollando, cambios de los miembros del equipo, cambios debidos a la

[1] Un participante es cualquiera que tenga algún tipo de interés en la terminación con éxito del proyecto, e incluye, entre otros a la dirección de la empresa cliente, los usuarios finales, los ingenieros web, el personal de atención al cliente, etc.

introducción de nuevas tecnologías, cambios de todo tipo que tienen un impacto en el producto a construir. Cualquier cosa que se haga relacionada con el software debe prever y contar con el cambio como algo que va a darse siempre. Un equipo ágil acepta que el software es desarrollado por individuos que trabajan en grupo y que las habilidades de esas personas y su capacidad de colaboración son la clave para el éxito del proyecto. Así pues, la presencia e influencia generalizada del cambio nos llevan de manera natural al concepto de agilidad.

2.2.2 Marco para la ingeniería web

Un marco para la ingeniería web (o marco de proceso para la ingeniería web o modelo de proceso para la ingeniería web) establece las bases para todo un proceso de ingeniería web completo, identificando un pequeño número de actividades marco que son aplicables a cualquier proyecto de aplicación web, independientemente de su tamaño y complejidad. Además, el marco incorpora un conjunto de actividades sombrilla que se aplican a lo largo de todo el proceso de ingeniería web.

Tal como se muestra en la Figura 2-1, cada actividad marco abarca una serie de acciones de ingeniería que, en su conjunto, dan lugar a un producto. Así por ejemplo, el diseño es una acción. A su vez, cada acción comprende una serie de tareas que se encargan cada una de realizar una parte del trabajo que corresponde a la acción.

Las siguientes actividades de la ingeniería web son parte de un marco genérico y son aplicables en la mayoría de los proyectos de aplicaciones web.

- **Comunicación**. Implica un elevado nivel de interacción y colaboración entre los desarrolladores y los clientes (y otros participantes) y abarca la obtención de requisitos y otras actividades relacionadas.

- **Planificación**. Se trata de establecer un plan de tipo incremental[2] para el trabajo a desarrollar. Describe las acciones que tendrán lugar, las tareas a realizar, los posibles riesgos, los recursos que serán necesarios, los resultados que se obtendrán, y un calendario.

- **Modelado**. Comprende la creación de esquemas que ayuden a los desarrolladores y a los clientes (y al resto de participantes) a entender mejor los requisitos de la aplicación web y el diseño que permitirá satisfacer dichos requisitos.

- **Construcción**. Incluye tanto la generación de código HTML, XML, Java y de otros lenguajes como las pruebas necesarias para descubrir errores en dicho código.

- **Despliegue**. Se entrega un incremento de la aplicación web al cliente, que lo evalúa e informa al equipo de ingeniería web de la evaluación realizada.

[2] Un plan incremental asume que la aplicación web será entregada en una serie de incrementos, que incluirán conjuntos de requisitos más robustos en cada entrega sucesiva. Así pues, un incremento proporciona ciertos contenidos y funcionalidad concretos al usuario final. Posteriores incrementos van aumentando los contenidos y funcionalidad entregados hasta que la aplicación web completa sea desplegada.

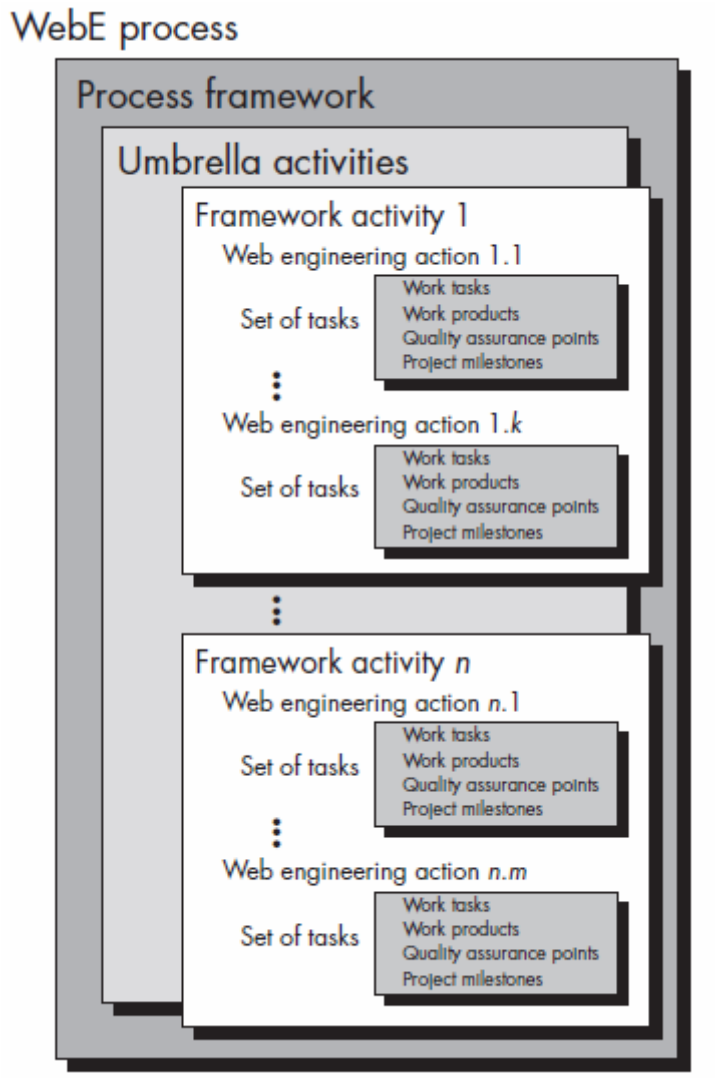

Figura 2-1. Marco para la ingeniería web

Estas cinco actividades genéricas son aplicables en el desarrollo de cualquier aplicación web independientemente de su tamaño y complejidad. Los detalles del marco pueden variar mucho para cada proyecto concreto, pero las actividades genéricas son siempre las mismas.

En la Figura 2-1 también se aprecia que cada acción incluye un conjunto de tareas. En cada caso, debe escogerse el conjunto de tareas que mejor se adapte a las necesidades del proyecto y a las características del equipo de ingeniería web. Por lo tanto, las acciones (como por ejemplo, obtención de requisitos) pueden adaptarse a las necesidades del proyecto de aplicación web y a las características del equipo.

Las actividades sombrilla (como por ejemplo, gestión del riesgo, garantía de calidad, gestión de contenidos...) se aplican a lo largo del proceso de ingeniería web.

La aplicación inteligente de cualquier marco debe aceptar que la adaptabilidad (al problema, al proyecto, al equipo y a la cultura de la empresa cliente) es esencial para alcanzar el éxito. La adaptabilidad afecta a cada una de las siguientes características de un marco:

- Flujo general de las actividades, acciones y tareas e interdependencias entre las mismas.
- Nivel al que se definen las tareas.
- Nivel al que se identifican y requieren los resultados de trabajo.
- Manera de aplicar las actividades de garantía de calidad.
- Modo de aplicar las actividades de seguimiento y control del proyecto.
- Grado general de detalle y rigor en la descripción del proceso.
- Nivel de implicación de clientes y otros participantes en el proyecto.
- Grado de autonomía del que goza el equipo del proyecto software.
- Nivel al que se especifican la organización y papeles dentro del equipo del proyecto software.

2.2.3 Principios para la adaptación del marco

Al adaptar el marco genérico para un proyecto dado, debe hacerse hincapié en la agilidad, para lo que se recomienda seguir 12 principios de agilidad adoptados por la *Agile Alliance*[3].

- La máxima prioridad es la de satisfacer al cliente mediante prontas y continuas entregas de software que aporte valor al cliente.
- Dar la bienvenida a los cambios en los requisitos, incluso en fases avanzadas del desarrollo. Los procesos ágiles contemplan el cambio como una oportunidad para que el cliente gane ventaja competitiva.
- Entregar software funcional con frecuencia, en rangos que pueden ir de un par de semanas a un par de meses, con preferencia siempre por plazos lo más breves posible.
- Clientes y desarrolladores deben trabajar juntos de manera continua a lo largo de todo el proyecto.
- Construir el proyecto apoyándose en individuos motivados, proporcionándoles el entorno y el apoyo que necesiten y confiando en ellos.
- El método más efectivo y eficiente para transmitir información a un equipo y dentro de un equipo es mediante conversaciones cara a cara.
- La principal métrica para medir el progreso de un proyecto es el software operativo.
- Los procesos ágiles fomentan el desarrollo sostenible y regular. Los clientes y los desarrolladores deben ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejoran la agilidad.
- La simplicidad, entendida como la capacidad de maximizar el trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños son los producidos por equipos autogestionarios.
- A intervalos regulares, el equipo reflexiona acerca de cómo ser más efectivo, y entonces adapta y ajusta su comportamiento en consecuencia.

[3] `http://www.agilealliance.com/`

2.2.4 Enfoque clásico

Aun teniendo en cuenta lo visto hasta ahora, siempre puede quedar la duda de si el enfoque clásico, basado en la informalidad, urgencia, intuición y arte tiene algo que aportar a la ingeniería web. La respuesta es un "sí" rotundo. Pero comprendiendo siempre que estas fuerzas deben ser moderadas por una filosofía que se encargue de reducir el riesgo y de aumentar la probabilidad de éxito.

La agilidad fomenta la informalidad, a la vez que admite que hay una sensación de urgencia asociada al desarrollo de toda aplicación web con niveles de calidad industriales. Cada equipo de ingeniería web debería adaptar el marco genérico para que se adapte lo mejor posible al problema en cuestión, confiando en la intuición y en la experiencia para guiar la manera en que se lleva a cabo dicha adaptación. Las acciones y tareas que se llevan a cabo dentro de cualquier marco adaptado aplican métodos técnicos concretos para el análisis de requisitos, el diseño, la generación de código y las pruebas. Sin embargo, estos métodos producirán unos resultados satisfactorios sólo si se conjugan con el arte que cada ingeniero web competente aporta al trabajo que realiza.

2.3 Componentes de la ingeniería web

A estas alturas resultará ya obvio que la idea es que la ingeniería web abarca todas las prácticas y el modelo de la ingeniería del software, pero adaptándolos a las características propias y distintivas de las aplicaciones web. Por este motivo, procedemos a continuación a repasar de forma muy concisa la caracterización de la ingeniería del software.

2.3.1 Estructura de la ingeniería del software

Podemos considerar que la ingeniería del software se estructura en una serie de capas o niveles, tal como se ilustra en la Figura 2-2. La base es un compromiso de toda la organización con la calidad, con un esfuerzo por fomentar una cultura de mejora continua. Es esta cultura la que, a la larga, conduce al desarrollo de enfoques de la ingeniería del software cada vez más efectivos.

La capa de procesos es el pegamento que mantiene unidas todas las capas y que permite un desarrollo racional y a tiempo del software. Constituye la base para la gestión de proyectos software y establece el contexto en el que se aplican los métodos técnicos, se producen los resultados de trabajo, se establecen hitos, se asegura la calidad y se gestiona el cambio.

Los métodos proporcionan las instrucciones concretas acerca de cómo construir el software. Los métodos abarcan un amplio abanico de acciones y tareas que incluyen comunicación, análisis de requisitos, modelado de diseño, programación, prueba y mantenimiento. Estos métodos se apoyan en una serie de principios básicos que gobiernan cada área de la tecnología e incluyen actividades y técnicas descriptivas.

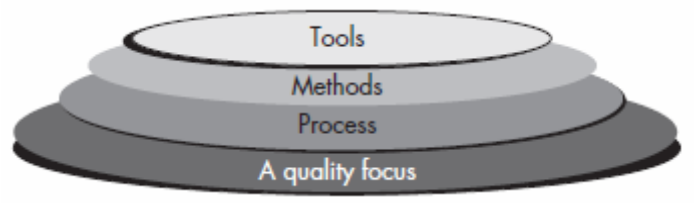

Figura 2-2. Estructura de la ingeniería del software

Las herramientas proporcionan una asistencia automatizada o semiautomatizada para el proceso y para los métodos. Cuando las herramientas se integran de forma que la información generada por una herramienta puede servir como entrada para otra herramienta, podemos hablar de un entorno automatizado de ayuda a la ingeniería del software.

2.3.2 Importancia de la agilidad para la ingeniería web

Ya hemos visto que el marco para la ingeniería web debe ser ágil. Esto implica un enfoque de ingeniería ligero con ciclos de desarrollo rápidos. Cada ciclo da lugar al despliegue de un incremento de la aplicación web.

Internet ha hecho que la principal prioridad del desarrollo de software pase del qué al cuándo. El *time-to-market* reducido se ha convertido en el santo grial de la competitividad por el que luchan todas las empresas. Por este motivo, uno de los objetivos más importantes de la ingeniería del software (y de la ingeniería web) es el de reducir los ciclos de desarrollo.

Es importante entender que el marco para la ingeniería web debe estar definido en el contexto de un proceso que:
- Acepte el cambio.
- Fomente la creatividad y la independencia del personal de desarrollo y una fuerte interacción con los participantes.
- Funcione en base a equipos reducidos.
- Haga hincapié en el desarrollo incremental a través de ciclos de desarrollo cortos.

A la vez que enfatizamos que la agilidad es la principal filosofía subyacente a la ingeniería web, no debemos olvidar que esto no significa que no sea necesario un enfoque riguroso de ingeniería.

2.3.3 Métodos del marco para la ingeniería web

El conjunto de métodos de la ingeniería web comprende una serie de tareas técnicas que permiten a un ingeniero web entender, describir y construir aplicaciones web de calidad. Los métodos de la ingeniería web pueden clasificarse como sigue.

Métodos de comunicación. Definen el enfoque utilizado para facilitar la comunicación entre los ingenieros web y el resto de participantes. Las técnicas de comunicación son especialmente importantes durante la obtención de requisitos y siempre que se evalúe un incremento de la aplicación web.

Métodos de análisis de requisitos. Proporcionan una base para entender los contenidos y funcionalidad que la aplicación web debe proporcionar a los usuarios finales, así como los modos de interacción requeridos por cada clase de usuarios.

Métodos de diseño. Abarcan una serie de técnicas de diseño enfocadas al contenido de la aplicación web, su arquitectura, su interfaz y su estructura de navegación.

Métodos de construcción. Aplican un amplio repertorio de lenguajes, herramientas y otras tecnologías relacionadas para la creación del contenido y la funcionalidad de la aplicación web.

Métodos de prueba. Incluyen revisiones técnicas tanto del contenido como del modelo de diseño y una gran variedad de técnicas de prueba aplicables a nivel de componentes y a nivel arquitectónico, pruebas de navegación, de facilidad de uso, de integridad y de configuración.

Además de los métodos técnicos que se acaban de esbozar, son necesarias una serie de actividades sombrilla (con sus respectivos métodos asociados), si queremos que el proceso de ingeniería web resulte exitoso. Dentro de estas actividades sombrilla podemos mencionar las técnicas de gestión de proyectos (por ejemplo, estimación, planificación, análisis de riesgos...), técnicas para la gestión de la configuración software, técnicas de revisión...

2.3.4 Papel de las herramientas y la tecnología

Las herramientas y la tecnología amplifican la capacidad de los ingenieros para construir sistemas computerizados. Cuando son usadas correctamente, las herramientas nos permiten trabajar más rápidamente y crear productos de alta calidad. Esto hace que las herramientas resulten tremendamente atractivas para los ingenieros, y los ingenieros web no son una excepción.

Pero las herramientas no pueden ser un fin en sí mismo y no tiene sentido utilizarlas sólo porque son "guays". Si uno no entiende realmente el problema, si no tiene forma de acomodar los cambios a medida que se van produciendo, si no ha dedicado tiempo a perfilar una solución viable, si no tiene la intención de garantizar que la "solución" que ha ideado satisfaga las necesidades de los participantes, entonces se está trabajando sobre el vacío, y este tipo de situaciones suelen acabar en desastre.

Las herramientas y la tecnología son muy importantes, pero sólo funcionarán correctamente si son empleadas en el contexto de un marco ágil para la ingeniería web y en conjunción con métodos contrastados para entender un problema, diseñar una solución y probarla de manera concienzuda.

Existe un amplio repertorio de herramientas y tecnología que ha evolucionado en los últimos años a la vez que las aplicaciones web se hacían cada vez más sofisticadas y omnipresentes. Estas tecnologías incluyen una variedad de lenguajes de modelado y de descripción de contenidos, como por ejemplo HTML, XML, VRML (*virtual reality modeling language*); lenguajes de programación, como por ejemplo Java o Python; recursos para el desarrollo basado en componentes, como CORBA (*common object*

request broker architecture), la arquitectura COM (*component object model*), ActiveX, .NET; navegadores, herramientas multimedia, herramientas para la conectividad con bases de datos, herramientas de integridad, servidores y utilidades para servidores, y herramientas para la gestión y análisis de sitios web.

2.4 Recomendaciones para la ingeniería web

En la ingeniería web se utilizan diferentes prácticas y técnicas para la ingeniería web. Dependiendo del proyecto en cuestión y de las características del equipo de desarrollo, procederá adoptar unas u otras de estas prácticas y técnicas. No obstante, en cualquier caso e independientemente de los factores mencionados, hay una serie de consideraciones esenciales y que deberían aplicarse siempre, y que se presentan a continuación.

1. **Dedicar el tiempo necesario a entender las necesidades del negocio y los objetivos del producto, incluso si los detalles de la aplicación web son vagos**. Muchos desarrolladores de aplicaciones web creen, erróneamente, que los requisitos vagos (algo bastante habitual) implican que la aplicación web a desarrollar no tiene por qué tener necesariamente un objetivo de negocio justificado. Esta consideración falaz conduce muy a menudo a un trabajo técnicamente bueno pero que materializa un sistema inadecuado construido por unos motivos erróneos y para un público inadecuado. Hay que actuar con mucha cautela si los participantes no son capaces de explicar una necesidad de negocio para la aplicación web. No debe iniciarse el desarrollo hasta que los participantes sean capaces de proporcionar un conjunto de objetivos de negocio claros a los que contribuiría la aplicación web y cómo contribuiría esa hipotética aplicación web a satisfacer los mencionados objetivos.

2. **Describir cómo interactuarán los usuarios con la aplicación web usando un enfoque basado en escenarios**. Los participantes deberían estar convencidos de la conveniencia de desarrollar escenarios que reflejen cómo diferentes usuarios interaccionarán con la aplicación web. Estos escenarios pueden entonces emplearse para: planificar y seguir el proyecto, servir de guía para el modelado de análisis y diseño, y como entrada para el diseño de casos de prueba.

3. **Desarrollar un plan, aunque sea breve**. El plan debe estar basado en un marco que sea aceptable para todos los participantes. Puesto que los plazos serán muy cortos, debe elaborarse una planificación temporal de grano fino, siendo a menudo necesario que el calendario y el seguimiento se hagan por días.

4. **Dedicar un cierto tiempo a modelar aquello que va a ser construido**. Normalmente, en los proyectos web no se elabora una documentación de análisis y diseño exhaustiva. Sin embargo, algunos contados modelos gráficos pueden aportar una valiosa luz al proyecto.

5. **Revisar la consistencia y calidad de los modelos**. Deben aplicarse diferentes técnicas de verificación y validación a lo largo de un proyecto de ingeniería web. El tiempo dedicado a estas tareas compensa con creces, ya que evita tener que repetir trabajo ya hecho y da lugar a aplicaciones web de alta calidad, aumentando así el nivel de satisfacción de los clientes.

6. **No reinventar la rueda**. Es muy importante aprovechar las ventajas de los componentes reutilizables, tanto a nivel de modelado como de construcción. Existe un amplio abanico de patrones de diseño específicos para aplicaciones web. Estos patrones permiten a un equipo de ingeniería web hacer el diseño arquitectónico, navegacional y detallado de forma rápida y con soluciones ampliamente probadas. Para la construcción de la aplicación web también existen numerosas herramientas que permiten a los desarrolladores construir partes significativas de la aplicación web empleando componentes reutilizables.

7. **No confiar en los primeros usuarios para depurar la aplicación web, sino diseñar casos de prueba sistemáticos y ejecutarlos antes de desplegar el sistema**. Habitualmente, los usuarios de una aplicación web le darán una única oportunidad; si no les satisface, la abandonarán para siempre y buscarán otra que les complazca. Por este motivo, debe seguirse la filosofía de "primero probar y después desplegar", incluso cuando los plazos son estrechos.

3 Modelo para la ingeniería web

3.1 Introducción

Hay quien piensa que una organización puede definir un marco para el desarrollo de aplicaciones web de antemano, ponerlo en la estantería (metafóricamente), y cuando haga falta, cogerlo, desempolvarlo y aplicarlo tal cual cuando haya que comenzar un nuevo proyecto de ingeniería web. Pero las cosas no funcionan así.

Es cierto que puede desarrollarse un marco para la ingeniería web y aplicarlo a todos los proyectos web que se presenten. Pero tal como se vio en el tema 2, el marco debe adaptarse a las características específicas de cada problema, al proyecto y a los participantes.

Un marco para la ingeniería web genérico proporciona la capacidad de entender el problema a abordar (tanto desde el punto de vista del negocio como de la tecnología). Una vez logrado este entendimiento básico, hay que trabajar para adaptar el marco genérico para que se ajuste a las necesidades específicas del problema en cuestión, teniendo en cuenta la cultura del equipo que hará el trabajo, los deseos de los clientes, el grado de estabilidad de los requisitos, las exigencias de los jefes de la empresa de desarrollo y otros factores.

Una vez que el trabajo haya comenzado, es necesario ir adaptando continuamente el marco para asegurar que:

1. No entorpezca la agilidad.
2. Realmente se adapte a las necesidades de los miembros del equipo de ingeniería web.
3. Sólo se generen los productos de trabajo intermedios necesarios para progresar de forma rápida hacia la entrega del incremento de la aplicación web planeado.
4. Permita evaluar continuamente la calidad del trabajo.
5. Permita acomodar los cambios que se van presentando y que afectan al calendario y al enfoque establecidos previamente.

En su esencia, el marco no cambia, pero sí cambia la manera de aplicarlo para cada proyecto y cada equipo. El marco es aplicado iterativamente a medida que se crea cada incremento de la aplicación web.

3.2 Definición del marco

Antes de definir un marco para la ingeniería web, conviene reiterar algunas ideas que se cumplen en la mayoría de los proyectos de aplicaciones web:

1. **Los requerimientos evolucionan a lo largo del tiempo**. La incertidumbre es parte inherente de la mayoría de los proyectos de aplicaciones web, por lo que los cambios en los requisitos son habituales. Además, estos cambios pueden venir

causados por la reacción (*feedback*) de los usuarios frente a los incrementos que se van entregando y por un entorno de negocio cambiante.

2. **Plazos cortos**. Esta circunstancia descarta la creación de una documentación voluminosa, pero, en cualquier caso, el análisis del problema, el diseño y las pruebas deben documentarse de alguna manera.

Esto conduce de manera natural a un desarrollo incremental de las aplicaciones web. Así, las actividades del marco se llevarán a cabo repetidamente a medida que se desarrolla y entrega cada uno de los incrementos de la aplicación web (ver Figura 3-1). Además, deben aplicarse los principios y recomendaciones presentados en el tema 2 (secciones 2.2.3 y 2.4), aunque no deben aplicarse de forma dogmática; a veces puede ser razonable contemplarlos de una forma relativamente relajada.

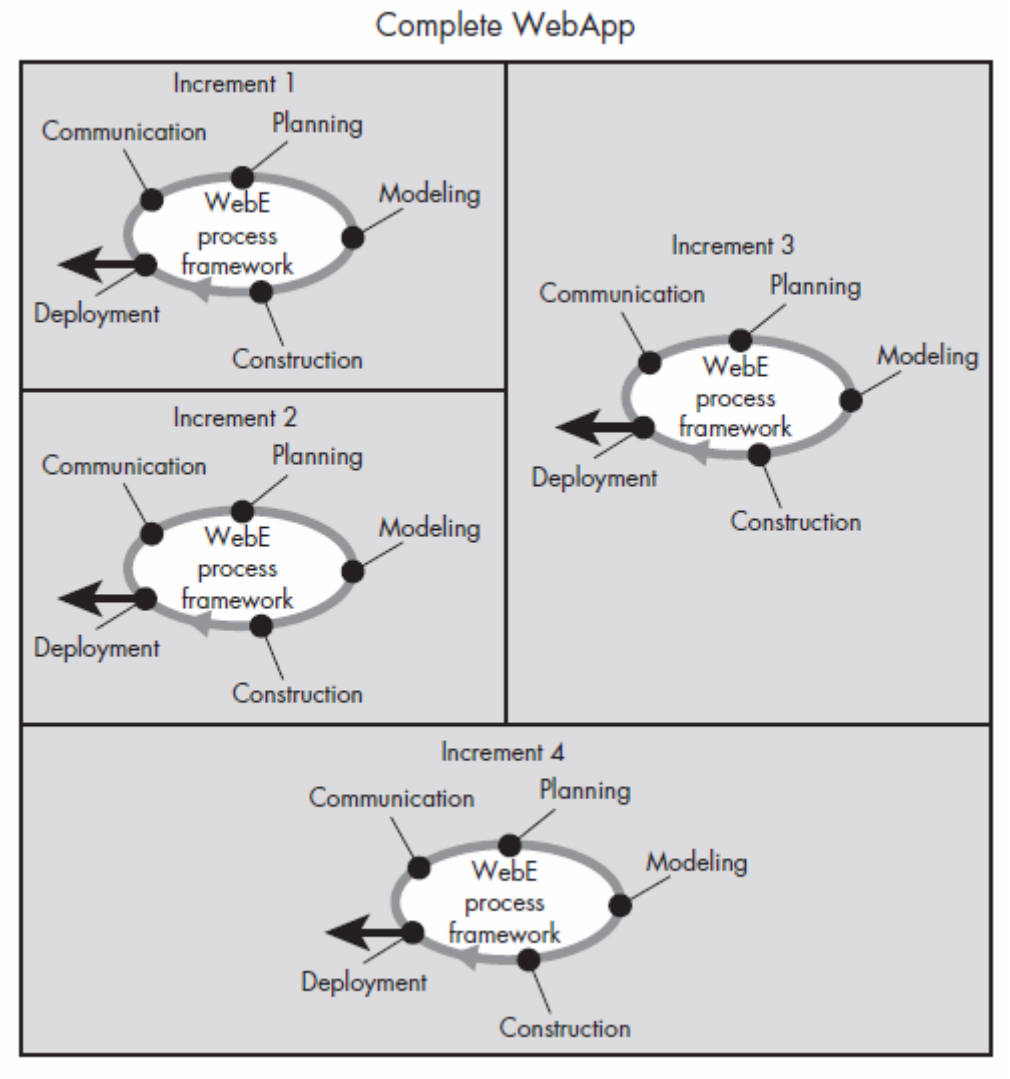

Figura 3-1. Aplicación web entregada en 4 incrementos

Teniendo en cuenta estas ideas y con la Figura 3-2 como referencia, ampliamos lo expuesto en el tema 2 acerca del marco para la ingeniería web.

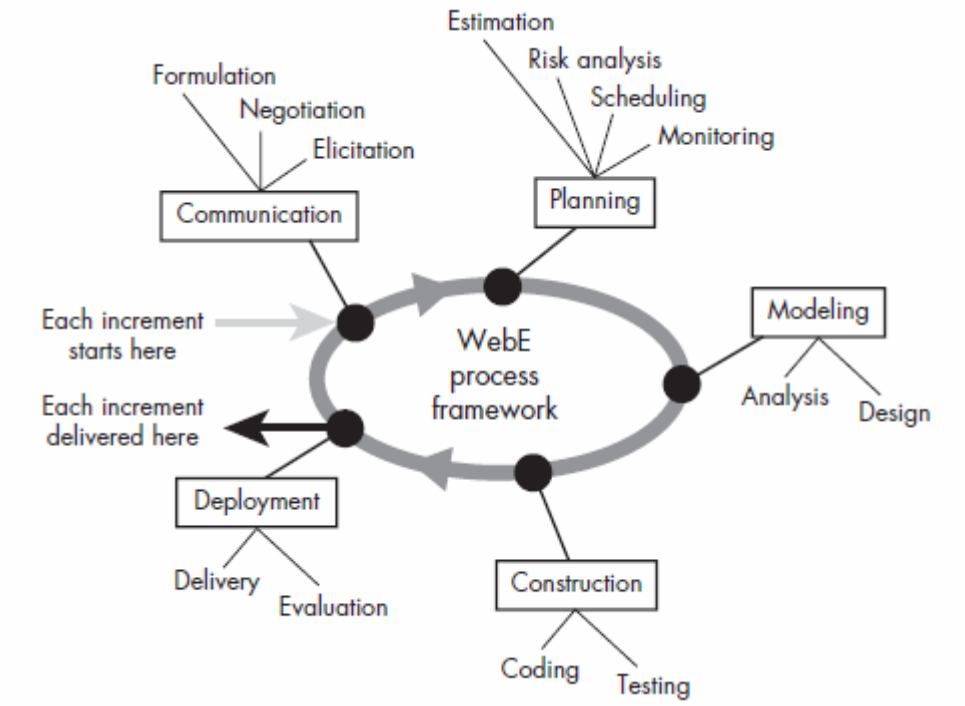

Figura 3-2. Flujo de proceso con acciones

Comunicación. Dentro del proceso de ingeniería web, la comunicación viene caracterizada por tres acciones: formulación, obtención y negociación. La formulación define el contexto de negocio y organizacional para la aplicación web. Además, se identifica a los participantes, se prevén posibles cambios en el entorno de negocio y los requisitos, y se especifica la integración de la aplicación web con otras aplicaciones, bases de datos y funciones. La obtención es una actividad de recopilación de requisitos que implica a todos los participantes. El objetivo es describir el problema que la aplicación web va a ayudar a resolver. Además, se intenta identificar áreas de incertidumbre en las que podrían surgir cambios. Finalmente, la negociación es necesaria a menudo para tratar de resolver diferencias entre distintos participantes.

Planificación. Se identifica el número de incrementos[4] y se crea un breve plan de proyecto para el siguiente incremento a entregar. Se hace una estimación de los recursos necesarios para el proyecto, se consideran los riesgos, se eligen y temporalizan tareas y comienza la monitorización y seguimiento del proyecto. En la mayoría de los casos, la planificación de un incremento corresponderá a un periodo de unas cuantas semanas.

Modelado. Las tareas propias del análisis y diseño de software convencional se adaptan al desarrollo de aplicaciones web. El objetivo es desarrollar modelos de análisis y diseño de forma ágil que permitan especificar los requerimientos y la aplicación web que los satisfaga.

Construcción. Se aplican herramientas y tecnologías propias de la ingeniería web para construir la aplicación web que ha sido modelada. Una vez construido el incremento de la aplicación web, se llevan a cabo una serie de pruebas rápidas para tratar de descubrir

[4] Aunque el número de incrementos puede variar a medida que se recogen las reacciones a los incrementos que se van entregando.

errores de diseño en la arquitectura, interfaz y navegación. Otras pruebas se enfocan a otros aspectos de las aplicaciones web.

Despliegue. La aplicación web es configurada para su entorno operativo y entonces es entregada a los usuarios finales, comenzando entonces un periodo de evaluación, en base a la cual se modifica el incremento según se solicite.

Estas cinco actividades se aplican siguiendo un enfoque incremental, tal como se ilustra en la Figura 3-2.

3.3 Flujo de proceso incremental

En el tema 1 se introdujo el caso de CPI Corporation junto con sus productos *CasaSegura* y *CasaSegura.com*, una completa aplicación web que servirá tanto como eje para el marketing y las ventas como la llave para una nueva generación de servicios de monitorización. De momento no existe nada, salvo una estrategia general y algunas buenas ideas.

Hay una empresa desarrolladora de software a la que se ha encargado formar un equipo de ingeniería web para crear esta aplicación web. La necesidad es imperiosa, y es necesario que haya una presencia operativa en la web lo antes posible. Es obvio que el desarrollo debe ser ágil. También es evidente que las cosas irán cambiando (posiblemente, de forma drástica) a medida que el proyecto vaya avanzando.

La empresa desarrolladora decide emplear las actividades definidas para el marco genérico (comunicación, planificación, modelado, construcción y despliegue) y que la única forma razonable de entregar la aplicación web es por incrementos. Los incrementos son identificados durante la primera iteración en la actividad de comunicación, pero la lista de incrementos puede ser refinada posteriormente.

3.3.1 Realización de las actividades

Si nos fijamos en la Figura 3-1 y la Figura 3-2, podemos apreciar que el flujo de proceso iterativo se aplica a cada incremento. La primera iteración se centra en definir los requisitos generales de la aplicación web y en identificar los incrementos a desplegar en las distintas iteraciones.

Primera iteración. El objetivo de la primera actividad de comunicación es definir el contexto de negocio, establecer los requisitos generales, crear un conjunto de escenarios de uso, negociar necesidades contrapuestas de diferentes participantes y, a partir de toda esta información, obtener el conjunto de incrementos de la aplicación web a entregar. Por ahora, suponemos que el primer incremento a entregar consiste en una aplicación web informativa que presenta a CPI y sus productos. Los participantes indican que esta aplicación web informativa debe entregarse en una semana.

En el caso del proyecto *CasaSegura.com*, se inicia la actividad de comunicación para el primer incremento celebrando una reunión de una hora con los participantes, lo que permite hacerse una idea razonablemente buena de qué se espera de este primer incremento. Se sabe entonces que el primer incremento a entregar consiste en una aplicación web informativa que presenta a CPI y sus productos. Los participantes indican que esta aplicación web informativa debe entregarse en una semana, aunque, para alivio de la empresa desarrolladora, todos los contenidos necesarios para este primer incremento están ya disponibles en ficheros que posee CPI y que un grafista de CPI está ya trabajando en el diseño del aspecto estético que el cliente quiere para la web. Tras la reunión, el ingeniero web que ha asistido revisa sus notas:

Logotipo y gráficos: necesitan diseño estético
Introducción de uno o dos párrafos:
    Declaración de la misión de CPI (existe fichero)
    Unas palabras destinadas a los visitantes (alguien las escribirá mañana)
La barra básica de navegación tendrá el siguiente aspecto:
    Acerca de la compañía
    Nuestras ofertas
        Productos de seguridad para el hogar (jerarquía al siguiente nivel)
        Servicios de monitorización (lista)
    Nuestra tecnología (el nuevo sensor)
    Contacto
Otras cuestiones:
    El contenido informativo cambiará con el tiempo
    Esta página web será el punto inicial de navegación para el contenido y funcionalidad necesarios para los sucesivos incrementos

Como se ha decidido seguir el marco para la ingeniería web, la siguiente actividad es la de planificación. Como todo el contenido necesario en esta etapa está disponible y no es necesario implementar ninguna funcionalidad, este ingeniero decide hacer todo el trabajo él solo, mientras va pensando en el pequeño equipo que va a formar para los siguientes incrementos. En una media hora crea una planificación temporal sencilla:

Día 1: Crear un prototipo de la aplicación web
    Recopilar y revisar todos los contenidos de CPI
    Obtener opiniones de participantes acerca del prototipo, si es posible
Día 2: Comenzar la construcción del incremento usando el prototipo como guía
    Construir la barra de navegación
    Distribuir las áreas de contenidos
    Integrar gráficos, enlaces, etc.
    Probar todos los enlaces
    Revisar si el contenido es completo y correcto
Día 3: Subir todos los ficheros a un dominio (ya existente)
    Hacer pruebas de navegación
    Despliegue: informar a determinados participantes de que el incremento está disponible
Día 4: Sondear la opinión de los participantes
    Hacer modificaciones en base a opiniones de participantes

Ya está listo el plan para el primer incremento de la aplicación web *CasaSegura.com*. A lo largo de los 4 siguientes días, este ingeniero web procede con el modelado, construcción y despliegue.

Siguiente iteración. Tan pronto como termina el despliegue del primer incremento puede comenzar la siguiente iteración del proceso de ingeniería web. En el caso de *CasaSegura.com*, la actividad de comunicación en esta siguiente iteración permitirá identificar los requisitos de contenido y funcionalidad para el segundo incremento. Por lo tanto, se vuelve a comenzar por el principio, llevando a cabo la actividad de comunicación, lo que permite saber que en este segundo incremento, el objetivo es disponer de una aplicación web de descarga, que permita descargar especificaciones de productos y otra información relacionada.

Las tareas concretas que se llevan a cabo dentro de cada actividad pueden variar de un incremento a otro, pero el flujo de proceso general es siempre el mismo. A medida que los incrementos se van volviendo más complejos, es probable que aumente tanto el número de tareas necesarias para llevar a cabo cada actividad como el nivel de complejidad de dichas tareas. Las actividades deben adaptarse y refinarse en cada iteración.

A lo largo de todas las iteraciones la agilidad debe ser el principio rector. Los desarrolladores y el resto de participantes son un equipo, y los compañeros de equipo se hablan, comunicándose como parte del flujo de cada incremento. Hay que planificar, pero sólo en la medida en que el plan proporcione un itinerario hacia el resultado final y una forma de seguir el avance del proyecto. Debe modelarse, pero únicamente si el incremento a construir es complejo o los requisitos o el enfoque de diseño no están claros. Debe construirse el incremento haciendo pruebas antes de desplegarlo. Hay que escuchar la reacción de los usuarios finales para solucionar los problemas del incremento desplegado, pero también para guiar el trabajo de los siguientes incrementos. Además, se llevarán a cabo actividades sombrilla para asegurar la calidad, controlar el cambio, gestionar el riesgo y supervisar el proyecto.

3.3.2 Cómo se refina el marco

Las acciones y tareas necesarias para refinar cada actividad dependen del criterio del equipo de ingeniería web. En algunos casos, una actividad se lleva a cabo de manera informal. En otros casos, un conjunto de distintas acciones será definido y llevado a cabo por los miembros del equipo. Cuando una acción es compleja (por ejemplo, diseño), puede descomponerse en un conjunto de tareas (por ejemplo, diseño estético, diseño de contenidos, diseño arquitectónico, diseño de navegación y diseño de componentes).

Pero, ¿cómo se decide qué tareas deben aplicarse? Para responder a esta pregunta es necesario entender el ámbito de la actividad en cuestión para un determinado incremento. Por ejemplo, la comunicación es una actividad que se aplica en todos los proyectos de ingeniería web. Veamos cómo podría detallarse la actividad de comunicación para dos incrementos distintos de *CasaSegura.com*.

Para el primer incremento (una aplicación web informativa) debe entenderse el contenido a presentarse, la estética general (esquema de colores, disposición, formato de menú, fuentes tipográficas, estilos...), el número de categorías de información a presentar y otra información relacionada. Puesto que se desea hacer hincapié en la agilidad, se desea obtener esta información lo antes posible (el primer incremento debe estar desplegado en una semana). Por lo tanto, la comunicación implica una única acción: reunirse con el cliente. No se va a generar ningún resultado de trabajo formal.

Consideremos ahora la actividad de comunicación para el tercer incremento, que permitirá funcionalidad de comercio electrónico. En este incremento, la actividad de comunicación será más compleja, llevándose a cabo tres acciones: formulación, obtención y negociación. Para cada acción se definirán tareas de trabajo.

La formulación consiste en comprender el contexto de negocio. La obtención comprende el especificar los requisitos de contenido y funcionalidad del comercio electrónico (requisitos sobre procesamiento por formularios, tarifas, costes de envío, impuestos, pedidos especiales, etc.), así como restricciones y requisitos no funcionales (rendimiento, integridad, etc.). La negociación implica resolver diferencias entre participantes.

Siguen siendo necesarias las reuniones con los demás participantes, pero aumentará tanto la duración de las reuniones como el número de tareas a realizar. En este caso sí que es bastante probable que se generen algunos productos de trabajo formales (por ejemplo escenarios de uso para diferentes clases de usuarios, un boceto del formulario de pedido...) y que se dedique más tiempo a revisar estos productos con los participantes. Una alternativa sería usar una plantilla de comercio electrónico ya existente, lo que permitiría presentar un prototipo a los participantes, acelerando así la tarea de comunicación.

3.4 Acciones y tareas genéricas

Aunque es necesario adaptar las acciones y tareas para cada iteración de cada proyecto y equipo de ingeniería web diferentes, es necesario tener un punto de partida. Por este motivo, se proporcionan a continuación unas descripciones de acciones y tareas genéricas como referencia de base a partir de las cuales refinar las acciones y tareas concretas para cada incremento de un proyecto en particular.

3.4.1 Comunicación

Si uno no sabe a dónde va, es extremadamente difícil llegar a destino o saber si se ha llegado o no. Aunque esto es de sentido común, sigue siendo bastante habitual que se comience el desarrollo de una aplicación web sin tener ni idea del destino. Puede que al final se llegue a un destino aceptable, pero dando muchos rodeos y vueltas atrás a lo largo del viaje, lo que resultará en que se llegará tarde. En otros casos nunca se llega a buen puerto. También hay ocasiones en que se llega a destino a tiempo, pero no se es consciente de ello, por lo que se continúa adelante innecesariamente sin saber cuándo parar.

La comunicación es la actividad que establece el destino para un proyecto de aplicación web. Para un destino simple, basta con un conjunto relativamente pequeño de acciones y tareas informales para estar seguro de saber a dónde se va. Si el destino es más difícil de describir, será necesario refinar la actividad de comunicación más cuidadosamente. Las siguientes tareas y cuestiones relacionadas deberían permitir ponerse en marcha:

- **Identificar a los participantes en el negocio**. Exactamente, ¿quién es el "cliente" de la aplicación web? ¿Qué empleados pueden actuar como expertos de dominio y representantes de los usuarios finales? ¿Quién actuará como miembro activo del equipo? ¿Cuál es el grado de consenso entre los participantes? ¿Quién es el árbitro decisorio cuando surjan diferencias entre los participantes?

- **Identificar clases de usuarios**. ¿Cuántos tipos distintos de usuarios interactuarán con la aplicación web? ¿Cuáles son los conocimientos de cada clase de usuarios? ¿Quién identificará las necesidades particulares de cada clase de usuarios y cuáles son esas necesidades? ¿Qué contenidos y funcionalidades especiales requiere cada clase de usuarios?

- **Formular el contexto de negocio**. ¿Cómo encaja la aplicación en la estrategia de negocio global? ¿Es firme la estrategia de negocio y existen reglas de negocio consolidadas?

- **Definir los objetivos de negocio correspondientes a la aplicación web**. ¿Cómo va a medirse el éxito de la aplicación web? Si hay varios objetivos, ¿cuáles son sus prioridades relativas? ¿Diferentes participantes tienen distintos objetivos? ¿Son coherentes entre sí todos los objetivos?

- **Identificar el problema**. ¿Qué problema específico ayuda a resolver la aplicación web? ¿Qué información se genera para el usuario final? ¿Qué información debe introducir el usuario final? ¿Qué funcionalidad se requiere para manipular los datos? ¿Qué información almacenada usa la aplicación web? ¿Qué otros sistemas interactuarán con la aplicación web?

- **Definir objetivos informacionales y de aplicación**. ¿Qué tipos de contenidos van a proporcionarse a los usuarios finales? ¿Cómo de volátil es el contenido (con qué frecuencia cambia)? ¿Qué funciones y tareas de usuario deben poder realizarse con la aplicación web? ¿Cómo de volátil es la funcionalidad requerida?

- **Recoger requisitos**. ¿Qué tareas de usuario serán implementadas? ¿Qué contenidos deben desarrollarse? ¿Qué tipo de interacción va a emplearse? ¿Qué funciones de cómputo va a proporcionar la aplicación web? ¿Cómo se configurará la aplicación web para su uso en red? ¿Qué esquema de navegación se desea? ¿Qué restricciones hay? ¿Qué requisitos no funcionales deben considerarse?

- **Desarrollar escenarios de uso**. ¿Se ha obviado alguna clase de usuarios? ¿Los escenarios de uso son consistentes con los requisitos del incremento? ¿Es necesario refinar los escenarios de uso?

Como se decía antes, si el destino es relativamente simple, cada una de las tareas de comunicación (y cuestiones relacionadas) puede llevarse a cabo de manera informal reuniéndose con los participantes adecuados y tomando notas. Sin embargo, si el destino es más complejo, puede ser necesario un enfoque más estructurado.

3.4.2 Planificación

La actividad de comunicación nos proporciona un destino, lo que nos permite, a continuación, planificar el viaje. Puesto que el viaje puede ser bastante complejo, lo más normal es planificar el viaje por etapas, definiendo puntos de paso para asegurarnos de ir en la dirección correcta y para ir progresando paso a paso hacia el objetivo final.

Aunque es en la primera iteración de la actividad de comunicación donde se establecen los puntos de paso (incrementos de la aplicación web), es en la actividad de planificación en la que se definen los recursos que serán necesarios para alcanzar cada punto de paso y en la que se estima el tiempo necesario para llegar a ellos. Las siguientes tareas y cuestiones relacionadas permitirán desarrollar un plan incremental:

- **Refinar la descripción de los incrementos a entregar**. ¿Hay cambios solicitados por algún participante que requieran una modificación del número de incrementos o de la definición de alguno de los incrementos que quedan por entregar? Si hay que hacer modificaciones de este tipo, ¿qué cambios de contenido y funcionalidad son necesarios? ¿Qué esfuerzo es necesario para cada incremento que queda por entregar? ¿Cuánto tiempo llevará cada uno de estos incrementos? ¿Cuál es la fecha de despliegue estimada para cada incremento?

- **Seleccionar el incremento a entregar a continuación**. ¿Hay suficiente información acerca del incremento como para comenzar otras actividades? ¿Se tiene un entendimiento claro del contenido y funcionalidad a entregar en el incremento? ¿Se entienden bien las restricciones y requisitos no funcionales? ¿Están disponibles y completos todos los escenarios de uso necesarios?

- **Estimar el esfuerzo y tiempo requeridos para desplegar el incremento**. ¿Cuánto esfuerzo y tiempo son necesarios para modelar, construir y desplegar el incremento? ¿Qué recursos (personas, hardware y software) harán falta para realizar el trabajo?

- **Estimar el riesgo asociado con la entrega del incremento**. ¿Qué riesgos deben ser tenidos en cuenta durante el desarrollo del incremento? ¿Cómo se abordarían los riesgos altamente probables y de alto impacto? ¿Qué riesgos a largo plazo deberían considerarse?

- **Definir el calendario de desarrollo para el incremento**. ¿Cómo se acomodarán las tareas a lo largo del plazo para el incremento? ¿Qué hitos intermedios se establecerán?

- **Establecer entregables resultantes de cada actividad**. ¿Qué entregables (por ejemplo, escenarios por escrito, bocetos, modelos, documentos) serán desarrollados a medida que procede el trabajo en el incremento?

- **Definir un enfoque para el control de cambios**. ¿Cómo se solicitarán, evaluarán y ejecutarán los cambios en contenido y funcionalidad dentro del contexto de otras actividades de desarrollo?

- **Establecer un enfoque para la garantía de calidad**. ¿Cómo se evaluará la calidad a medida que el incremento es modelado, construido y desplegado? ¿Se harán revisiones? En caso de hacerse, ¿de qué tipo? ¿Se utilizarán métricas? En caso de utilizarse, ¿cuáles?

Puesto que los incrementos suelen desarrollarse en plazos de semanas, es razonable cuestionarse si la planificación está justificada en el caso de la ingeniería web. Siempre es aconsejable establecer un itinerario hacia el destino, pero hay que tener cuidado de no dedicar tanto tiempo a la planificación que quede poco tiempo para alcanzar el objetivo. Por lo tanto, es recomendable hacer una planificación ágil para cada incremento, mediante una única reunión en la que todos los miembros del equipo de ingeniería web ayuden a realizar la planificación. También es aconsejable optar por una planificación adaptable. Las cosas cambian, con lo que es necesario modificar la planificación a medida que se desarrolla el incremento.

3.4.3 Modelado

En el contexto del marco para la ingeniería web propuesto, el modelado es una actividad encargada de crear una o más representaciones conceptuales de algún aspecto de la aplicación web a construir. Tal representación conceptual puede adoptar una o varias de las siguientes formas: documentos escritos, bocetos, diagramas esquemáticos, modelos gráficos, escenarios escritos, prototipos ejecutables o no y código ejecutable. El modelado comprende dos acciones: análisis y diseño.

3.4.3.1 Tareas de modelado de análisis

En el análisis, se empieza examinando los requisitos de los participantes utilizando información recogida durante la actividad de comunicación. En muchos casos, la información recogida durante la comunicación basta como base para construir un incremento y ni siquiera es necesario el modelado de análisis. Sin embargo, cuando los requisitos son complejos, puede ser recomendable crear un modelo que permita entender mejor la aplicación web a desarrollar. En líneas generales, el modelo de análisis se centra en el contenido de la aplicación web, los modos de interacción (incluyendo la navegación), la funcionalidad y la configuración técnica[5]. Las siguientes tareas y cuestiones relacionadas pueden ayudar a decidir si desarrollar o no un modelo de análisis y a desarrollarlo, en caso de que sea necesario.

- **Decidir si hace falta un modelo de requisitos**. ¿La información disponible (recogida durante la actividad de comunicación) ofrece suficientes detalles sobre: (1) el contenido de la aplicación web, (2) los modos de interacción requeridos, (3) la funcionalidad requerida y (4) las cuestiones sobre la configuración técnica? ¿Han

[5] En este contexto, la configuración técnica se refiere al entorno hardware y software en el que residirá la aplicación web.

sido desarrollados los escenarios de uso con el suficiente nivel de detalle como para guiar el diseño y la construcción? Si esta información existe y es completa, entonces no hay necesidad de hacer un modelado de análisis para este incremento. Si la información es incompleta o tiene un grado de complejidad tal que requiere un estudio más avanzado, entonces debe procederse con las tareas de modelado de análisis que siguen.

- **Representar los contenidos de la aplicación web**. ¿Qué contenidos deben ofrecerse? ¿Cuál es su origen? ¿Quién es responsable de obtener dichos contenidos y de desarrollarlos? ¿Es recomendable organizar los contenidos en una colección de clases? ¿Son complejas las relaciones entre las clases de contenidos? ¿Qué clases de contenidos son estáticas (no cambian según el tipo de usuario ni posibles entradas) y cuáles son dinámicas (se generan en base al tipo de usuario o sus entradas)?

- **Identificar relaciones entre contenidos**. ¿Cómo se relaciona una clase de contenidos con otras? ¿Cuál es la forma y estilo de cada clase de contenidos?

- **Refinar y ampliar escenarios de uso**. ¿Qué tareas de usuario son llevadas a cabo como parte de este incremento? ¿Cómo realiza el usuario la tarea? ¿Qué información necesita el usuario para realizar una tarea? ¿Qué información proporciona el usuario para llevar a cabo una tarea? ¿Qué pasos son necesarios y cómo exactamente interactúa el usuario con la aplicación web? ¿Qué funciones deben existir para permitir al usuario realizar una tarea?

- **Revisar escenarios de uso**. ¿Hay inconsistencias u omisiones en algún escenario? ¿Están los escenarios suficientemente detallados? ¿Concuerda el escenario con el contenido y funcionalidad a implementar en el incremento?

- **Crear un modelo de interacción para escenarios complejos**. Si la secuencia de acciones especificada en un escenario es compleja, ¿cuál es la relación entre las tareas de usuario y los contenidos requeridos para cada tarea? ¿Qué estados[6] observables externamente pueden identificarse? ¿Qué acciones del usuario causan transiciones de un estado a otro?

- **Refinar los requisitos de interfaz**. ¿Permite el aspecto y disposición de la interfaz dar cabida a los escenarios de uso definidos? ¿Es necesario hacer modificaciones a los menús, la organización o la navegación?

- **Identificar funciones**. ¿Qué funciones realizará la aplicación web para el usuario? ¿Qué datos proporcionará el usuario para invocar una función? ¿Se entiende bien el algoritmo subyacente a cada función?

- **Definir restricciones y requisitos no funcionales**. ¿Se han descrito con suficiente detalle las restricciones y los requisitos no funcionales (definidos en la actividad de comunicación)? ¿Qué políticas de privacidad van a implementarse?

[6] Un estado es un modo de comportamiento observable externamente. En el contexto de una aplicación web, un modo de comportamiento puede ser un cambio significativo en el contenido mostrado en pantalla, el inicio de una función de cómputo o una condición concreta dentro de un flujo de trabajo.

- **Identificar requisitos de bases de datos**. ¿Qué base(s) de datos van a ser accedidas? ¿Está bien definido el protocolo de interfaz con la(s) base(s) de datos? ¿Qué clases de contenidos estarán implicadas?

Existe una gran variedad de métodos y notaciones que pueden aplicarse para llevar a cabo las tareas de modelado de análisis.

3.4.3.2 Elementos de un modelo de diseño

El diseño es fundamental en la ingeniería web. Adaptando los criterios que el crítico arquitectónico romano Vitruvio propuso en su tiempo acerca de los edificios, podemos enunciar tres características que debería tener toda aplicación web. Firmeza: una aplicación web no debería tener ningún defecto que inhiba su funcionalidad. Eficacia: una aplicación web debería ser adecuada para los propósitos para los que fue creada. Deleite: la experiencia de utilizar una aplicación web debería ser placentera. El objetivo del diseño en la ingeniería web es el de producir un modelo o representación que tenga firmeza, eficacia y deleite.

¿Debe crearse siempre un modelo de diseño cuando se desarrolle un incremento de una aplicación web? La respuesta es "sí", pero la forma de plasmar ese modelo puede variar de un incremento a otro. Si el incremento está perfectamente entendido y es muy fácil de construir, entonces el modelo de diseño puede reducirse a un simple boceto. Si por el contrario el incremento es más complejo, puede ser necesario crear un modelo de diseño más detallado. El modelo puede tener en cuenta algunos o todos de los siguientes aspectos relativos al diseño de aplicaciones web:

- **Diseño de la interfaz**. Describe la estructura y organización de la interfaz de usuario. Incluye una representación de la distribución de la pantalla, una definición de los modos de interacción y descripciones de los mecanismos de navegación.

- **Diseño estético**. También conocido como diseño gráfico, describe el aspecto (*look and feel*) de la aplicación web. Incluye esquemas de colores, disposición geométrica, tamaño del texto, fuentes tipográficas, uso de gráficos y decisiones estéticas relacionadas.

- **Diseño de contenidos**. Describe la disposición, estructura y esquema de todos los contenidos que deban presentarse como parte de la aplicación web. Representa también las relaciones entre objetos de contenido.

- **Diseño de navegación**. Representa el flujo de navegación entre objetos de contenido y para todas las funciones de la aplicación web.

- **Diseño arquitectónico**. Identifica la estructura global de hipermedios de la aplicación web.

- **Diseño de componentes**. Describe la lógica detallada necesaria para implementar los componentes funcionales que permiten llevar a cabo la funcionalidad completa de la aplicación web.

3.4.3.3 Tareas de modelado de diseño

El grado en que un modelo de diseño aborda cada uno de los aspectos indicados en el apartado anterior depende del nivel de complejidad del incremento de la aplicación web que vaya a desarrollarse. En la mayoría de los casos, un modelo de diseño incluye algunos de los aspectos mencionados, pero no todos. Las siguientes tareas y cuestiones relacionadas pueden ser de ayuda a la hora de decidir cómo desarrollar un modelo de diseño:

- **Diseñar la interfaz**. ¿De qué manera van a representarse las tareas y subtareas de interacción como parte de la interfaz? ¿Qué mecanismos de control de la interfaz (por ejemplo, enlaces, botones, menús...) son necesarios? ¿Cómo se posicionan los mecanismos de control en la página web? ¿Están todos los escenarios de uso contemplados por el diseño?

- **Diseñar la estética de la aplicación web**[7]. ¿Cómo se implementará la distribución de la página web? ¿Variarán el color y la forma dependiendo del contexto? ¿Cómo se distribuirán y representarán los mecanismos de navegación? ¿Están creados y disponibles todos los logotipos, gráficos, imágenes y fondos? ¿El diseño estético es consistente a lo largo de los diferentes incrementos?

- **Diseñar el esquema de navegación**. ¿Qué nodos y enlaces de navegación son necesarios? ¿Qué convenciones y ayudas de navegación van a emplearse? ¿Está definido por completo el flujo de navegación? ¿Casan los mecanismos de navegación con los requisitos y diseño de interfaz? ¿Se ha optimizado la navegación para distintas clases de usuario? ¿Concuerda la semántica de navegación con todos los escenarios de uso?

- **Diseñar la arquitectura de la aplicación web**. ¿Qué estilo(s) arquitectónico(s) van a usarse para el contenido y la funcionalidad?

- **Diseñar el contenido y la estructura que lo sustente**. ¿Qué contenidos deben diseñarse como parte del incremento? ¿Qué estructuras de datos y bases de datos son necesarias para mostrar los contenidos y para implementar la funcionalidad? ¿Están definidas a nivel de diseño las interfaces a las bases de datos existentes?

- **Diseñar componentes funcionales**. ¿Qué componentes deben desarrollarse? ¿Han sido definidos todos los algoritmos? ¿Hay disponibles contenidos adecuados para cuando sea necesario su procesamiento?

- **Seleccionar patrones de diseño apropiados**. ¿Qué patrones arquitectónicos son apropiados para el espacio de información? ¿Existen patrones que puedan ayudar en el diseño de la navegación? ¿Pueden usarse patrones de interacción para el diseño de la interfaz? ¿Hay patrones de presentación que puedan servir para los contenidos? ¿Pueden llevarse a cabo los flujos de trabajo, comportamientos, procesamiento y comunicaciones mediante patrones funcionales?

[7] Normalmente, el diseño estético para aplicaciones web con requisitos de calidad comerciales debe ser realizado por profesionales específicos (grafistas).

- **Diseñar mecanismos de integridad**. ¿Qué nivel de seguridad se exige para que los usuarios accedan al sistema? ¿Qué nivel de integridad de los datos se requiere para proteger de accesos no autorizados los contenidos y funcionalidad tanto en la parte cliente como en la parte servidor?

- **Revisar el diseño**. ¿Se ajusta el diseño a los requisitos de usuario? ¿Puede implementarse el diseño según la planificación desarrollada para el incremento?

En muchos casos, el diseño se realiza con las mismas herramientas que se usarán para la construcción. Estas herramientas permiten a un ingeniero web: distribuir la estructura arquitectónica y de navegación, situar y especificar contenidos; distribuir entradas mediante formularios; hacer un prototipo de la estética de la aplicación web. Con este enfoque, la transición del diseño a la construcción resulta muy fluida.

3.4.4 Construcción

Si las anteriores actividades han ido bien, se habrá identificado lo que los participantes quieren de la aplicación web y se habrá desarrollado un diseño que servirá como base para la actividad de construcción. A medida que progresa la construcción se llevarán a cabo dos acciones: generación de código y prueba. Las siguientes tareas y cuestiones relacionadas pueden ser de ayuda a la hora de planificar la acción de generación de código:

- **Elaborar y/o conseguir todos los contenidos e integrarlos en la arquitectura de la aplicación web**. ¿Qué tecnologías y herramientas de ingeniería web van a emplearse para construir los contenidos y componentes funcionales? ¿Qué formularios, plantillas y patrones preexistentes pueden usarse para la construcción?

- **Seleccionar el conjunto de herramientas apropiadas para la generación de código HTML**. ¿Puede generarse todo el código HTML con el conjunto de herramientas o será necesario codificar a mano algunos elementos?

- **Implementar todas las distribuciones de página, funciones, formularios y mecanismos de navegación**. ¿Están disponibles todos los contenidos para ser integrados en cada una de las páginas web del incremento?

- **Implementar todas las funciones de cálculo**. ¿Qué formularios, *scripts* e interfaces de bases de datos deben implementarse? ¿Han sido correctamente diseñados todos los algoritmos de cálculo? ¿La funcionalidad se despliega en el lado del cliente o en el del servidor?

- **Resolver cuestiones de configuración**. ¿Qué navegadores, complementos (*plug-ins*) y sistemas operativos podrán usarse tanto en la parte cliente como en la parte servidor?

Una vez que la aplicación web haya siso construida hay que probarla. La prueba comienza centrándose en aspectos concretos (grano fino) y va ampliando su foco de atención (grano cada vez más grueso) para abarcar una visión más global de la

aplicación web. Las siguientes tareas y cuestiones relacionadas pueden ser de ayuda a la hora de planificar la acción de prueba:

- **Probar todos los componentes de la aplicación web (contenido y funcionalidad)**. ¿Qué componentes deben probarse en el contexto de tareas de usuario? ¿Se han diseñado los casos de prueba de forma que permitan probar todas las posibilidades de funcionalidad?

- **Probar la navegación**. ¿Qué enlaces deben probarse en el contexto de las tareas de usuario? ¿Qué escenarios de uso son aplicables al incremento para desarrollar pruebas de navegación adecuadas? ¿Se han diseñado los casos de prueba de forma que permitan probar completamente la estructura de navegación?

- **Probar la facilidad de uso**. ¿Qué mecanismos relativos a la facilidad de uso deben probarse? ¿Qué escenarios de uso son aplicables al incremento para desarrollar pruebas de facilidad de uso adecuadas? ¿Han sido los casos de prueba diseñados de manera que permitan asegurar que todos los escenarios de uso pueden realizarse?

- **Probar la integridad y el rendimiento**. ¿Cómo se aplican todos los filtros de seguridad del incremento? ¿Cómo se prueba el rendimiento general del incremento? ¿Se han diseñado los casos de prueba de forma que quede asegurada la integridad tanto en la parte cliente como en la parte servidor?

- **Probar el incremento de la aplicación web para diferentes configuraciones**. ¿Ha sido desarrollada una lista con todas las configuraciones técnicas? ¿Han sido los casos de prueba diseñados para probar el incremento dentro de todas las configuraciones de funcionamiento?

Es probable que todas o la mayoría de estas tareas de construcción se apliquen a cada uno de los incrementos que se desarrollen. No obstante, el equipo de ingeniería web puede combinar o racionalizar varias de estas tareas (y quizá algunas otras adicionales) si la situación lo justifica.

3.4.5 Despliegue

Una vez completado un incremento de la aplicación web, éste está listo para ser entregado a los usuarios finales. Los objetivos son: (1) proporcionar funcionalidad a una o varias clases de usuarios, (2) conocer la reacción de los usuarios finales para saber si se han satisfecho los requisitos para el incremento y (3) establecer una base para las modificaciones que ocurrirán necesariamente como consecuencia del despliegue. Las siguientes tareas y cuestiones relacionadas pueden ser de ayuda a la hora de desplegar el incremento de la aplicación web:

- **Instalar el incremento de la aplicación web en un servidor en un dominio predefinido**. ¿Se han seguido todas las convenciones sobre nombres de ficheros y directorios y sobre referencias a enlaces? ¿Se les ha proporcionado a los usuarios información de acceso? ¿Están dispuestos y funcionando los elementos de seguridad adecuados (por ejemplo, controles de contraseñas)?

- **Establecer un mecanismo en línea para obtener la reacción de los usuarios**. ¿Se ha implementado un formulario de comentarios acompañando a la entrega del primer incremento? ¿El formulario es de formato libre o tiene una lista de preguntas multirrespuesta específicas? ¿Es posible evaluar los datos de reacción de forma cuantitativa?

- **Evaluar la interacción del usuario final**. ¿Cómo interactúa con el sistema el usuario? ¿Qué partes de la interacción están poco claras, son ambiguas o faltan? ¿Qué contenido o funcionalidad es incorrecto o falta?

- **Evaluar lo aprendido y considerar todas las reacciones de usuarios finales**. ¿Qué cambios son necesarios según la reacción de los usuarios? ¿Deben hacerse los cambios inmediatamente o como parte del siguiente incremento a desarrollar?

- **Hacer al incremento las modificaciones requeridas**. ¿Qué modificaciones hay que hacer al incremento actual? ¿Qué cambios (a los requisitos y al diseño) hay que hacer a incrementos posteriores?

Las actividades, acciones y tareas mencionadas en estas secciones están apoyadas por una amplia colección de principios, directivas y métodos (de planificación, análisis, diseño, construcción y prueba).

3.5 Actividades sombrilla

A medida que se van aplicando cada una de las actividades genéricas que acabamos de ver, también se van llevando a cabo una serie de actividades sombrilla, que son igualmente importantes para el éxito de un proyecto, por lo que deben ser consideradas de forma explícita por todo equipo de ingeniería web.

Aunque hay muchas posibles actividades sombrilla que pueden definirse, hay cuatro que son cruciales para el éxito de un proyecto de ingeniería web:

- **Control del cambio**. Gestiona los efectos del cambio a medida que se desarrolla cada incremento, apoyándose en herramientas que ayudan a administrar todo el contenido de la aplicación web.

- **Garantía de calidad**. Define y lleva a cabo las tareas destinadas a que cada entregable y el incremento cuenten con el nivel de calidad esperado.

- **Gestión del riesgo**. Considera los riesgos técnicos y del proyecto a medida que se desarrolla un incremento.

- **Gestión del proyecto**. Sigue y monitoriza el progreso a medida que se desarrolla un incremento.

Destacamos a continuación los aspectos más importantes de las actividades sombrilla.

3.5.1 Control del cambio

El cambio puede ser una fuente de caos en cualquier proyecto. Altera el flujo normal del proceso (por ejemplo, el equipo está listo para desplegar un incremento, pero entonces debe posponerlo para adaptar dicho incremento a un cambio sobrevenido), absorbe recursos (la gente responde al cambio, abandonando temporalmente las tareas que se supone que tenían que llevar a cabo) y enturbia el objetivo del equipo (el destino ya no está tan claro). Pero, como sabemos, el cambio es inevitable, y puede ser constructivo. Entonces, ¿cómo afrontarlo?

La estrategia incremental que se recomienda para la ingeniería web ayuda a un equipo a manejar el cambio. Las solicitudes de cambio, procedentes de cualquier tipo de participante, pueden aparecer en cualquier momento, pero una vez que se está trabajando en un incremento de la aplicación web, los cambios son puestos en cola para que, una vez desplegado el incremento, puedan ser evaluados y hacer lo que proceda tras la mencionada evaluación. Esto reduce la naturaleza perturbadora de las solicitudes de cambio, permite al equipo continuar con su trabajo pero aún así se permite que los cambios sean evaluados de forma relativamente rápida, ya que el desarrollo de incrementos suele ser rápido (del orden de semanas).

3.5.2 Garantía de calidad

Todo equipo de ingeniería web se esfuerza por producir una aplicación web de calidad. Pero, ¿qué se entiende por calidad en este contexto? ¿Qué directivas y tareas específicas hay disponibles para alcanzar esa calidad?

La calidad de una aplicación web se consigue gracias a un diseño sólido, cuyos principios son firmeza, eficacia y deleite, como vimos en la sección 3.4.3.2. El primer atributo de calidad, la firmeza, se logra cuando una aplicación web es fiable y presenta tanto contenido como funcionalidad libres de error. El segundo atributo de calidad, la eficiencia, se logra cuando la aplicación web satisface las necesidades de todos los participantes. El tercer atributo de calidad, el deleite, se logra cuando la aplicación web supera las expectativas tanto técnicas como de usuario.

La actividad sombrilla de garantía de calidad se centra en cada entregable que se produce a medida que se construye cada incremento. El objetivo es asegurar que el incremento a desplegar exhibirá firmeza, eficacia y deleite.

3.5.3 Gestión del riesgo

"¿Qué podría ir mal?". Ésta es una pregunta tan obvia que muchas veces nos olvidamos de formularla de manera explícita. La gestión del riesgo se encarga de responder a esta pregunta. Los riesgos son eventos o circunstancias que pueden hacer que un proyecto vaya mal. Es necesario identificar los riesgos para entonces poder gestionarlos.

La identificación de riesgos ocurre a lo largo de todo el proceso de ingeniería web. Siempre que cualquier participante (entre los que se incluyen los ingenieros web) identifica algo que podría ir mal, se registra para ser discutido y, si fuera necesario,

llevar a cabo las acciones necesarias. Se describe el riesgo potencial y se anota su impacto potencial.

Es necesario gestionar los riesgos de impacto alto, lo cual puede hacerse de dos maneras: de forma reactiva o de forma proactiva. El enfoque reactivo para la gestión de riesgos consiste en esperar a que ocurra algún problema para entonces tratar de solucionarlo. En el mejor de los casos, se monitoriza el proyecto vigilando los riesgos probables y se reservan recursos para aplicarlos en caso de que se materializara alguno de esos posibles riesgos. En el momento en que alguno de esos riesgos potenciales se materializara, entonces se llevarían a cabo las acciones necesarias para tratar de solucionarlo.

Suele ser más recomendable la gestión proactiva del riesgo, consistente en identificar los riesgos potenciales, evaluar su probabilidad e impacto para ordenarlos según su importancia y entonces trazar un plan para gestionar cada uno de esos riesgos.

3.5.4 Gestión del proyecto

Lo ideal es que un equipo de ingeniería web se autogestione. Cada miembro del equipo se compromete con las tareas a llevar a cabo y trabaja de forma diligente para completarlas dentro de los plazos establecidos. El equipo se organiza y supervisa a sí mismo.

No obstante, a medida que aumenta la complejidad de los incrementos, las interdependencias entre tareas suelen requerir del seguimiento y control de un líder de equipo o administrador. Después de todo, es importante saber si las cosas se están atrasando para poder reaccionar rápidamente para remediarlo.

La gestión de proyectos es una actividad sombrilla que ocurre como consecuencia de la planificación.

4 Construcción y despliegue

4.1 Introducción

Independientemente del tipo de aplicación web a desarrollar y del proceso (o ausencia de proceso) que se emplee para su desarrollo, en algún momento, el equipo de ingeniería web tendrá que construir una aplicación web basándose en el diseño y utilizando algunas de las tecnologías y herramientas existentes. Además, el equipo tendrá que desplegar la aplicación web en todos los entornos en los que se le exija. Según el modelo de proceso propuesto en estos apuntes, una aplicación web se desarrolla en base a una serie de incrementos. Las dos últimas actividades del proceso de ingeniería web son:

Construcción. Se usan tecnologías y herramientas de ingeniería web para implementar la aplicación web que ha sido modelada. Una vez construido un incremento de la aplicación web, se llevan a cabo una serie de pruebas rápidas para tratar de descubrir fallos de diseño, ya sea a nivel de contenido, funcionalidad, arquitectura, interfaz o navegación. También se llevan a cabo algunas otras pruebas para cubrir otras características de la aplicación web.

Despliegue. Se configura la aplicación web para su entorno operativo y entonces se pone disponible para los usuarios finales, comenzando entonces un periodo de evaluación, que lleva a presentar las correspondientes reacciones al equipo de ingeniería web, y el incremento se modifica según estas reacciones.

Aunque las actividades de construcción y despliegue son conceptualmente diferentes, la naturaleza distribuida de la web, combinada con la naturaleza incremental del proceso de ingeniería web, hacen que ambas actividades puedan intercalarse. Si un incremento comprende el desarrollo de varios componentes, cada uno de ellos puede desplegarse inmediatamente en cuanto haya sido construido y probado de manera independiente, proporcionando beneficios a los participantes cuanto antes. Esta posibilidad puede ofrecer una flexibilidad muy ventajosa, pero sólo si se gestiona con mucho cuidado. Si no se hace correctamente puede dar lugar a situaciones que seguramente todos hemos experimentado como usuarios de ciertas aplicaciones web en las que encontramos numerosos callejones sin salida etiquetados como "en construcción" o páginas descuidadas que han sido desplegadas antes de ser comprobadas adecuadamente.

En aras de la agilidad y flexibilidad, es incluso posible que la actividad de construcción se intercale un poco con la de modelado, ya que a veces se opta por no llevar a cabo el diseño detallado a nivel de componentes durante el diseño de la información ni durante el diseño funcional, sino que se pospone hasta la construcción, momento en el que se lleva a cabo un diseño de componentes más informal de forma conjunta con la generación de código y las pruebas.

4.2 Construcción y despliegue dentro del proceso de ingeniería web

La Figura 4-1 representa el proceso de ingeniería web destacando las actividades de construcción y despliegue. Se muestran las principales acciones asociadas a estas dos actividades. La figura ha sido ligeramente modificada con respecto a la que se ha mostrado en temas anteriores con el objeto de destacar la relación entre construcción y despliegue.

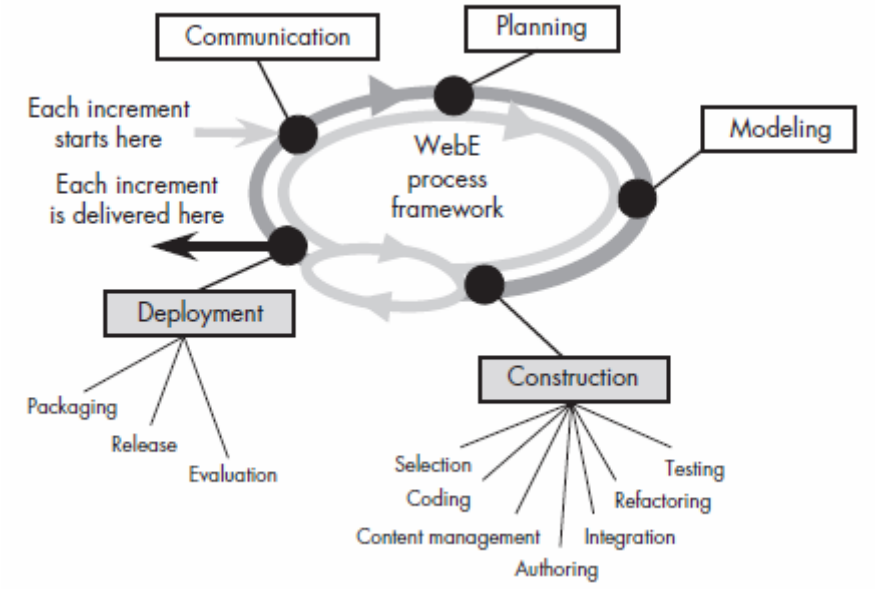

Figura 4-1. Construcción y despliegue dentro del proceso de ingeniería web

4.2.1 Relación entre construcción y despliegue

Aunque el ciclo general del proceso se repite para cada incremento de una aplicación web, existe además una significativa relación e iteración entre construcción y despliegue dentro de cada incremento. Además, hay ocasiones en las que se localiza alguna cuestión menor como consecuencia del trabajo realizado en construcción o despliegue, lo cual conlleva a un rápido repaso de alguna otra actividad (por ejemplo, modelado). En algunos casos, esta iteración interna será intencional y planeada. Otras veces será algo que surja sobre la marcha como consecuencia de las tareas llevadas a cabo.

Por ejemplo, el incremento 2 de *CasaSegura.com* ofrece al usuario información sobre productos, permitiendo buscar información de algún sensor determinado. Este incremento también contempla la personalización de la información para determinadas clases de usuarios. La arquitectura funcional incluye componentes para la generación de contenidos de productos (es decir, la extracción de los contenidos necesarios para una determinada página) y compilación dinámica de página (es decir, componer de manera dinámica el contenido, cabeceras, pies de página, menús y scripts de interacción de apoyo en una página). Aunque el diseño para el incremento 2 se haya hecho como algo completo y cohesionado, la construcción puede comenzar creando una versión de la generación de contenidos que ofrezca contenidos relevantes pero que no permita aún la adaptación de los mismos para diferentes clases de usuarios. Si esta versión limitada de

la funcionalidad del incremento es aceptable para los participantes, entonces puede desplegarse esta construcción intermedia mientras se va implementando la funcionalidad que falta.

Considérese ahora una situación en la que sea necesario volver atrás en el proceso de ingeniería web. Supongamos que el incremento 2 ha sido desplegado en su totalidad y se está llevando a cabo una revisión final del incremento en la que intervienen todos los participantes (o, al menos, representantes de todos los tipos de participantes). Se detecta lo siguiente:

1. Un usuario está buscando información de producto en una determinada categoría.

2. La categoría en cuestión no contiene ningún producto que esté disponible para el usuario.

No está claro si esta es una funcionalidad deseada (CPI desea que el usuario sepa que existe una determinada categoría aunque los productos de esa categoría no estén disponibles para ese tipo de cliente) o si es un descuido. Entonces se produce una vuelta a la actividad de comunicación para hacer una rápida comprobación con el participante adecuado, que indica que es correcto que el usuario conozca la existencia de esa categoría aunque ninguno de sus productos esté disponible para él, pero que debería mostrarse un texto dentro de esa categoría para indicar al usuario que únicamente los clientes de *CasaSegura.com* tienen acceso a esos productos. Entonces el equipo de ingeniería web comprueba si este nuevo requisito es consistente con el diseño actual (retomando brevemente la actividad de modelado). Una vez comprobado que es así, el equipo de ingeniería web implementa el cambio y despliega la nueva versión que incorpora dicho cambio. Todo esto habría ocurrido en un periodo de unas cuantas horas.

Estos ejemplos ilustran cómo la interrelación y entrelazamiento entre la construcción y el despliegue y otras actividades pueden permitir un proceso de ingeniería web más ágil y flexible.

4.2.2 Entornos de trabajo

En el caso de aplicaciones web sencillas, puede ser aceptable que un único desarrollador cree contenidos de manera local, por ejemplo, editando contenidos en una máquina local sin que los contenidos se alojen en un servidor aparte. Estos contenidos se publican entonces directamente al servidor de producción para que puedan ser accedidos inmediatamente por los usuarios. Este tipo de situación es el que se representa en la Figura 4-2.

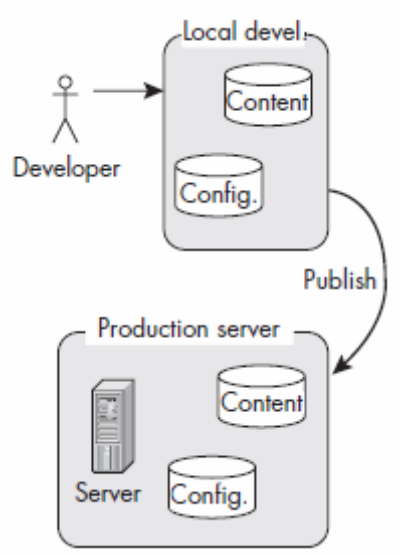

Figura 4-2. Desarrollo de aplicación web simple con un único desarrollador

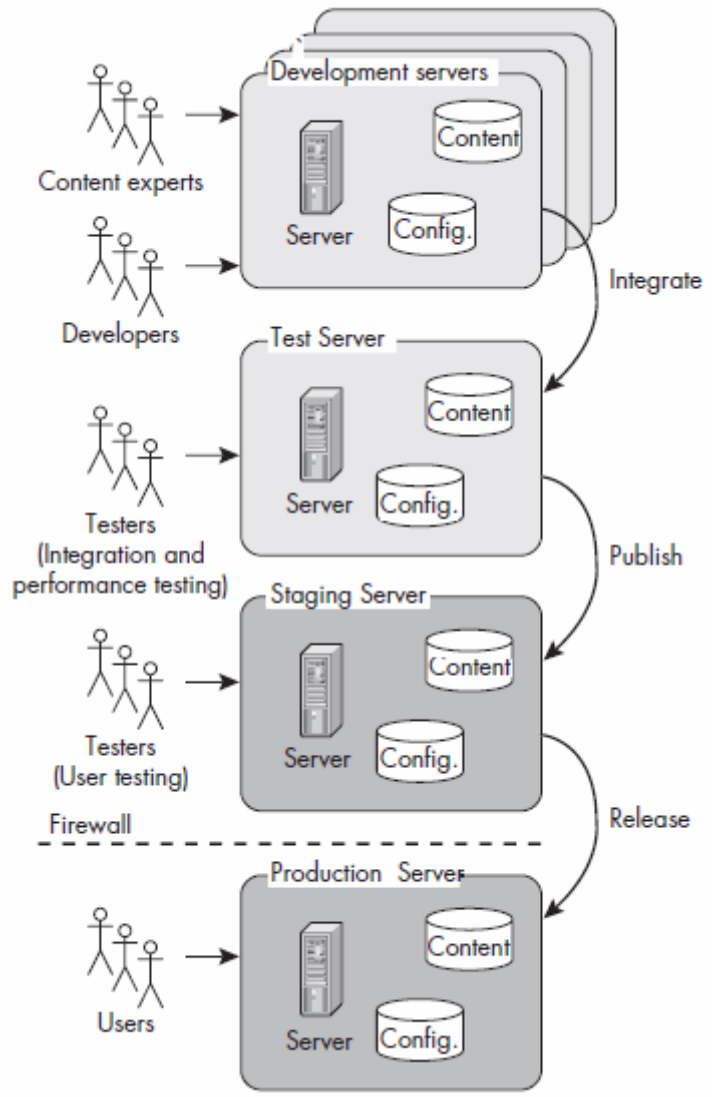

Figura 4-3. Desarrollo de aplicación web compleja con varios desarrolladores

En el caso de aplicaciones web más complejas (sobre todo aquellas en las que resulta crítico garantizar la calidad antes de desplegar) puede ser conveniente utilizar un entorno de trabajo más elaborado. En la Figura 4-3 se muestra un posible entorno de trabajo en el que intervienen cuatro tipos de servidores:

Servidor de desarrollo. Los desarrolladores utilizan estos servidores para llevar a cabo todo el desarrollo y pruebas unitarias[8]. Estos servidores constituyen un entorno controlado en el que los desarrolladores pueden crear, manipular y probar los componentes de sus aplicaciones web. En algunos casos, cada desarrollador puede tener su propio entorno personal con una copia de todos o algunos de los componentes de la aplicación web, junto con las herramientas apropiadas (por ejemplo, editores, entornos de desarrollo integrados (IDEs), material de referencia, etc.). Normalmente, la configuración de desarrollo no será la misma que la de los otros servidores.

Servidor de pruebas. Una vez que los desarrolladores han concluido la prueba unitaria de sus componentes, pasan a integrarlos para su verificación dentro del entorno del incremento completo. Normalmente, los servidores de pruebas no tienen instaladas herramientas de desarrollo porque podrían interferir con las pruebas. Sobre este tipo de servidores se realizan pruebas tanto de integración como de rendimiento.

Servidor de ensayo (*staging server*). El servidor de ensayo (o de escenificación) permite llevar a cabo una prueba exhaustiva del incremento por usuarios pero sin publicar el incremento en el servidor de producción y sin hacerlo disponible por tanto a toda la población de usuarios. Este servidor debe parecerse lo máximo posible al servidor de producción en todas sus características, como sistema operativo, herramientas de apoyo, hardware, controladores software (*drivers*), etc.

Servidor de producción. El incremento se instala en este servidor una vez que está listo para ser utilizado por todos los usuarios. Es fundamental que los cambios no sean desplegados en este servidor hasta que hayan sido concienzudamente probados en el servidor de ensayo.

El entorno concreto a adoptar para un proyecto determinado de aplicación web depende de la escala del proyecto, del número de desarrolladores y participantes y de la importancia de la aplicación web para la organización cliente. Hay casos en los que los cuatro tipos de servidores descritos se combinan en tres servidores (desarrollo, ensayo y producción) o incluso dos (desarrollo y producción). Hay casos en los que el equipo de ingeniería web puede utilizar un único entorno de desarrollo en red, mientras que en otros casos puede que distintos ingenieros web trabajen en diferentes redes. Puede haber casos en los que el servidor de producción (y por lo tanto también el servidor de ensayo) sea en realidad un *cluster* de ordenadores en red preparados para llevar a cabo un reparto de carga. Otra situación bastante habitual es la de tener un servidor de desarrollo y un servidor de prueba que son distintos desde el punto de vista lógico pero que funcionen ambos sobre un mismo servidor físico. Independientemente de estas u otras variaciones, siempre deben cumplirse los principios básicos:

- Mantener separados los entornos de desarrollo y de producción. No se debe desarrollar en servidores que sean accesibles para los usuarios.

[8] En este contexto, la prueba de unidad se refiere al intento organizado de descubrir errores en un componente informativo o funcional antes de su publicación como parte de un incremento.

- Proporcionar a los desarrolladores un entorno que facilite su productividad.

- Siempre que sea posible, llevar a cabo las pruebas en un entorno igual al entorno al que accederán los usuarios.

4.3 Construcción

La actividad de construcción abarca una serie de acciones de selección, codificación, creación (*authoring*), integración, refactorización y prueba que permiten obtener un incremento operativo que está listo para su despliegue a los usuarios finales. La selección se refiere a la búsqueda de componentes (u objetos contenidos en componentes) preexistentes relevantes que puedan ser reutilizados en el desarrollo actual. La codificación consiste en la creación de nuevos componentes o la adaptación de componentes preexistentes, lo cual puede requerir la creación de código HTML o en algún lenguaje de *scripting* o la generación automática de código a partir de alguna representación de diseño. La creación se refiere a la integración de contenidos dentro del diseño gráfico y la correspondencia de contenidos dentro de pantallas y páginas web. También puede incluir la creación de hojas de estilo. La integración es la unión de código, contenido y presentación para formar los componentes finales a desplegar. La refactorización es una acción iterativa que pule los componentes implementados para mejorar su estructura y claridad y para eliminar código redundante. Para terminar, la prueba consiste en verificar que los distintos componentes y objetos son correctos.

4.4 Despliegue

La actividad de despliegue comprende tres acciones: empaquetado, publicación y evaluación.

Para el software convencional, el empaquetado suele incluir la creación de una funcionalidad de instalación que puede ser distribuida por distintos medios a aquellos que quieran instalar la aplicación en cuestión. En el caso de las aplicaciones web esto no es aplicable, y el empaquetado se refiere a decidir los componentes a instalar en el servidor de producción y la manera de hacer esto (por ejemplo, ¿es mejor dejar el servidor de producción fuera de línea mientras se instalan los nuevos componentes o es mejor hacerlo con el servidor en línea en algún intervalo en el que se prevea una carga de trabajo reducida?).

Todo nuevo contenido o funcionalidad estará disponible inmediatamente para los usuarios, por lo que la acción de publicación debe tener en cuenta cuestiones como por ejemplo cómo comunicar la publicación de nuevos componentes a los usuarios. La publicación de componentes puede hacerse siguiendo un enfoque de grano fino, publicando componentes nuevos desde el servidor de ensayo al servidor de producción una vez que hayan sido probados. No obstante, si se abusa de este enfoque y se van produciendo numerosos y continuos pequeños cambios sobre una aplicación web, los usuarios pueden sentirse confundidos y puede aumentar la probabilidad de errores de integración y debidos a efectos laterales. Esto puede hacer que sea preferible empaquetar un conjunto de componentes y publicarlos en bloque (excepto cuando se trate de corregir algún error detectado).

Cada ciclo de empaquetado y publicación proporciona a los usuarios finales un incremento operativo que añade características de utilidad y que permite que tenga lugar una acción de evaluación de dichas características por parte de los usuarios. La reacción de los usuarios al nuevo incremento proporciona al equipo de ingeniería web una información muy importante que puede conducir a modificaciones sobre el contenido, funcionalidad y características de la aplicación web y también sobre el enfoque a seguir para el desarrollo del siguiente incremento.