

TITULO: Una adaptación del Proceso Unificado de Desarrollo para la creación de portales basados en Joomla.

TITLE: An Unified Process adaptation for the development of Joomla based web applications.

AUTORES:

Lic. Enrique José Leyva Miranda.
Ing. Mileidys González Prieto.

PAÍS: Cuba

RESUMEN:

Presenta una adaptación del Proceso Unificado (UP) para la creación de portales basados en Joomla, un sistema de gestión de contenidos (CMS) de código libre con características de plataforma para desarrollo de portales basado en componentes. En la modelación del proceso se ha empleado SPEM, un estándar basado en UML propuesto por el "Object Management Group" (OMG). Se proponen y describen cinco disciplinas e igual número de roles que se consideran adecuados a las características de este tipo de proyecto.

PALABRAS CLAVES:

INGENIERIA DEL SOFTWARE, MODELADO DE PROCESOS, PROCESO UNIFICADO, CMS, JOOMLA, SPEM

ABSTRACT:

Presents an Unified Process (UP) adaptation for the development of Joomla based web applications, which is an open source content management system (CMS) that becomes a framework for the component based web applications development. For the process modeling has been used SPEM, an UML based standard proposed by the "Object Management Group" (OMG). Five disciplines and the same number of roles that are considered convenient to this kind of projects are presented and described.

KEY WORDS:

SOFTWARE ENGINEERING, PROCESS MODELING, UNIFIED PROCESS, CMS, JOOMLA, SPEM

INTRODUCCIÓN:

El proceso de desarrollo de software establece las actividades requeridas para obtener, a partir de las necesidades de los usuarios, productos de software que las satisfagan. La definición de un proceso, describe los actores que ejecutan cada una de las actividades así como los resultados que producen (Acuña, 2001).

Un modelo del proceso es una representación abstracta del mismo, debe contener el conocimiento organizacional acerca del propio proceso. El modelado del proceso facilita su estudio y comprensión, la comunicación entre los actores involucrados, la gestión del proceso y hasta su propia evolución. Los modelos describen al proceso desde diversas perspectivas, entre las que se destacan (Curtis, 1992):

- Funcional. Describe que elementos del proceso se llevan a cabo y que flujos de información requieren dichos elementos.
- De comportamiento. Describe en que momento y bajo que condiciones se llevan a cabo los elementos del proceso.
- Organizacional. Describe donde y por quién se llevan a cabo los elementos del proceso.
- Informativa. Describe las entidades que genera o manipula el proceso, así como sus relaciones y estructura.

Muchos de los procesos de Desarrollo de Software más utilizados en la actualidad se basan en el Proceso Unificado; este ofrece, más que un proceso a seguir al pie de la letra, un marco a partir del cual pueden crearse procesos, por lo que cada organización debe decidir que roles, actividades y productos utilizar. UP se estructura como una matriz de dos dimensiones, donde las fases (columnas) representan los cuatro estados básicos por los que transita el proyecto, mientras las disciplinas (filas) establecen las actividades que han de llevarse a cabo. Sus características distintivas son: basado en casos de uso, centrado en la arquitectura, iterativo e incremental (Jacobson, 2000).

Al implementar un proceso de desarrollo basado en UP, decidir que roles, actividades y productos utilizar, no es algo trivial. De hecho es una tarea que requiere bastante conocimiento sobre el tipo de proyecto a desarrollar y el propio UP, lo que en la práctica la convierte en una barrera difícil de franquear para equipos carentes de la experiencia suficiente. El proceso que se describe en este trabajo es resultado de la experiencia de los autores en el desarrollo de portales basados en Joomla. No se pretende con esto ofrecer una receta mágica e incuestionable, pero se espera que pueda servir como guía a aquellos que se inicien en el desarrollo de este tipo de aplicaciones.

Joomla en pocas palabras

Un CMS es una aplicación que permite la creación, publicación y actualización de "contenidos", garantizando además la gestión de usuarios y permisos. Por contenidos, generalmente se entiende documentos, aunque muchos CMS (los más exitosos) han sido concebidos con una visión más abarcadora. Estos incluyen, además de la gestión de documentos, la posibilidad de extender el sistema para gestionar cualquier tipo de información.

Entre los CMS de este tipo, existen varios que son además de código libre, constituyendo plataformas muy útiles para el desarrollo de aplicaciones Web. El uso de estas herramientas libera al desarrollador de la necesidad de enfrentarse a componentes "clásicos" del sistema como la gestión de usuarios, contenidos estáticos, noticias, apariencia, etc. Permitiendo que concentre sus

esfuerzos en las especificidades de la aplicación que pretende desarrollar (más sobre CMS en Cuerda, 2004).

Joomla es un CMS de código libre, desarrollado en PHP, que se distribuye como un núcleo de funcionalidades básicas e incluye la gestión de: contenidos estáticos, noticias, plantillas, grupos de usuarios fijos, vínculos a otros sitios entre otros. Presenta cuatro tipos de elementos básicos:

1. Plantillas (templates). Definen la apariencia del sitio, en ellas se incorporan los elementos generales de diseño.
2. Módulos. Son fragmentos de la página donde se muestra determinada información o se brinda alguna funcionalidad.
3. Componentes. Al igual que los módulos están especializados en una tarea específica, pero a diferencia de estos proveen el contenido fundamental de cada página.
4. Mambots. Son una especie de controladores de eventos que ejecutan una acción determinada ante un evento como puede ser mostrar contenido, buscar, etc.

Este CMS está diseñado de forma tal que se le pueden incorporar muy fácilmente tanto nuevas plantillas como módulos, componentes y mambots. Lo que lo dota de una gran flexibilidad, permitiendo que se adapte a disímiles situaciones. Cuenta con una activa comunidad de desarrolladores que constantemente están suministrando nuevos elementos para extender las funcionalidades del sistema. A la hora de desarrollar una aplicación web basada en este CMS es muy probable encontrar gran parte de las funcionalidades requeridas implementadas total o parcialmente, lo que reduce enormemente el tiempo de desarrollo y abarata los costos.

MATERIALES Y METODOS:

Básicamente se ha empleado la **modelación**, para obtener una representación del proceso. Con este fin se ha hecho uso de **SPEM**, un metamodelo basado en **UML** para definir (modelar) procesos. El que se sustenta en la concepción de estos como colaboraciones entre entidades abstractas llamadas roles, que llevan a cabo operaciones llamadas actividades sobre entidades concretas llamadas productos (SPEM, 2005).

Se ha llevado a cabo una **revisión bibliográfica** sobre el tema del modelado de procesos para adecuar el trabajo a las tendencias actuales.

Otros métodos utilizados han sido: **análisis histórico – lógico, análisis y síntesis.**

RESULTADOS DEL TRABAJO:

Las disciplinas, según SPEM, permiten agrupar los elementos del proceso de acuerdo a temas comunes. UP propone nueve disciplinas: Modelamiento del Negocio, Gestión de Requerimientos, Análisis y Diseño, Implementación, Prueba, Implantación, Gestión de Proyecto, Gestión de Configuración y

Cambio, y Entorno. La adaptación que se propone establece cinco disciplinas: Gestión de Proyecto, Gestión de Requerimientos, Diseño, Implementación y Prueba. En este epígrafe se describen los roles, actividades y artefactos que se proponen para cada una de las disciplinas, así como la secuencia en que se considera que las actividades deben ser llevadas a cabo.

Gestión del Proyecto

La gestión agrupa actividades relacionadas con la planificación, control y evaluación de la ejecución del proyecto. Crea las condiciones necesarias para la correcta ejecución de las restantes disciplinas. En el proceso que se propone, un sólo rol ejecuta las actividades de gestión, el jefe de proyecto, que puede ser una sola persona o un pequeño equipo de dirección (ver figura 1).

La gestión comienza con la planificación de la fase de inicio. En este punto se tiene sólo una vaga noción de lo que será sistema, bien sea por una petición de un cliente o una idea surgida en el propio equipo de desarrollo. No se cuenta con elementos de juicio suficientes como para comprender el alcance del proyecto, ni para evaluar su viabilidad, por lo que el objetivo fundamental de esta fase será capturar la información necesaria para determinar si se desarrollará o no el sistema, y negociar con el cliente el alcance de este.

Se recomienda que se lleve a cabo la fase de inicio a través de una sola iteración, y se trabaje sólo en la captura de requerimientos (que se describe más adelante). El plan de esta fase debe contemplar fundamentalmente: el personal responsable de la captura de requerimientos, las acciones previstas para el seguimiento (reuniones, partes, etc.) y los criterios de medida para dar por terminada la fase, es poco realista concebir fecha de culminación al planificar la fase, aunque debe contemplarse su inclusión, cuando ya esta haya avanzado lo necesario.

No es aconsejable destinar muchos recursos humanos a esta etapa, pues generalmente las fuentes de información (clientes, usuarios finales, beneficiarios, etc.) están poco disponibles, lo que hace que el trabajo no sea intenso y sí extenso. Esto provoca que un equipo grande desperdicie mucho tiempo y dispare innecesariamente los costos, se recomienda emplear no más de dos o tres personas, entre las que estará incluido el propio jefe de proyecto.

La fase de inicio termina con la evaluación de la viabilidad del proyecto. Si se determina que no es viable, se procede a su cancelación, documentando las causas que motivaron tal decisión. En caso de determinarse como viable el proyecto, se lleva a cabo su contratación; donde debe dejarse claro el alcance del proyecto, los objetivos generales, el tiempo aproximado de desarrollo y el costo estimado.

Una vez contratado, sobre esta base se estructura un plan general del proyecto que guiará todo el proceso de desarrollo en lo adelante. En este plan deben especificarse, entre otros aspectos, la cantidad de iteraciones previstas, la duración promedio de estas y los responsables de los roles dentro del proceso.

A partir de este momento la gestión transcurre a lo largo de las iteraciones del proyecto comenzando cada una con una planificación, controlando luego la ejecución del plan y finalmente evaluando los resultados.

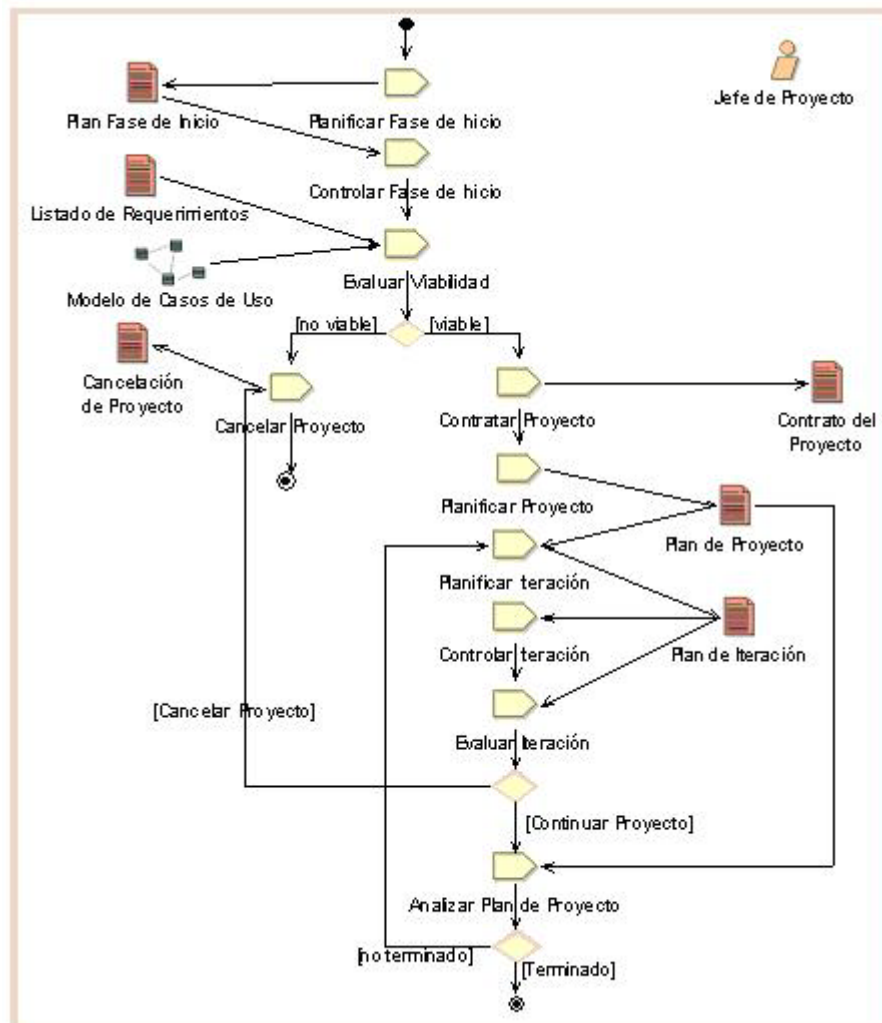


Figura 1 Secuencia de actividades de la Gestión del Proyecto

En el plan de iteración deben quedar especificados: los objetivos de la iteración (casos de uso a especificar, casos de uso a implementar, componentes a los que se dará mantenimiento, etc.), el tiempo previsto para alcanzar cada objetivo, los recursos con que se contará (estaciones de trabajo, recursos humanos, etc.), los responsables de cada objetivo y el modo en que será controlada la ejecución del plan.

Al evaluar la iteración, se debe analizar: la medida en que fueron logrados los objetivos, los imprevistos que afectaron la ejecución de la iteración y cualquier factor que atente contra la viabilidad del proyecto. De existir algún factor de este tipo se estudian los posibles modos de atenuar sus efectos, decidiéndose cancelar el proyecto en caso de no encontrarse manera alguna.

Gestión de Requerimientos

Los requerimientos son funcionalidades que debe brindar el sistema (requerimientos funcionales) u otro tipo de características que debe poseer (requerimientos no funcionales). La gestión de requerimientos agrupa un conjunto de actividades que tienen como objetivos: determinar los requerimientos del sistema, obtener casos de uso a partir de los requerimientos, determinar el orden en que los casos de uso serán implementados y propagar hacia el sistema, de manera controlada, los cambios en los requerimientos. Los roles que este trabajo propone para esta disciplina son el analista de sistema y el jefe de proyecto (ver figura 2).

La gestión de requerimientos comienza con la determinación, por el analista de sistema, de un conjunto inicial de estos. Para lograrlo se apoyará en: entrevistas con clientes y usuarios, criterios de expertos y estudio de proyectos similares, entre otras técnicas y herramientas. Siempre teniendo en cuenta que el cliente es quien en última instancia decide qué requerimientos debe satisfacer el sistema. Esta actividad arroja un listado de requerimientos clasificados en funcionales y no funcionales, pero estos no estarán detallados, ni serán los definitivos, sino sólo una primera aproximación que irá evolucionando en las diferentes iteraciones del proceso. Si bien en las primeras iteraciones, la actividad “determinar requerimientos” arrojará muchos nuevos requerimientos, a medida que el proceso avance, el resultado fundamental estará dado por cambios en requerimientos ya identificados, que pueden aparecer como respuesta a solicitudes del cliente o a propuestas del propio equipo de desarrollo.

En cualquier caso esta actividad dará paso a una de las siguientes:

1. Actualizar Caso de Uso. En caso de existir un requerimiento funcional que ya ha dado lugar a un caso de uso y luego ha cambiado, se debe redefinir el caso de uso, actualizándolo de modo que responda nuevamente al requerimiento que lo originó. Puede suceder incluso, aunque es poco frecuente, que el requerimiento desaparezca, lo que haría necesaria la eliminación del caso de uso. La actualización puede estar motivada también por un cambio en los requerimientos no funcionales, en este caso la propagación suele afectar a varios casos de uso.
2. Definir Caso de Uso. Si existe un requerimiento funcional nuevo, o lo que es lo mismo sin su correspondiente caso de uso, se impone definirlo. Este nuevo caso de uso no estará detallado, será solo un esbozo.
3. Detallar Caso de Uso de mayor prioridad. Si no se cumple ninguna de las situaciones anteriormente descritas, entonces se pasa directamente a especificar en detalles el caso de uso que esté definido como el de mayor prioridad entre los no detallados. Esta actividad debe producir una descripción textual, a través de una tabla de eventos de cada uno de los escenarios que formen el caso de uso, así como los esbozos de las interfaces de usuario correspondientes, artefactos que servirán de guía a la hora de diseñar e implementar el caso de uso.

Así en un ciclo constante, a lo largo de todo el proceso, el analista de sistema va determinando requerimientos, actualizando el modelo de casos de uso y detallando casos de uso. Como complemento, el jefe de proyecto, en cada iteración analizará el listado de casos de uso no detallados, para ordenarlos según la prioridad que le confiera, para este fin tendrá como criterios fundamentales: el riesgo y la relevancia arquitectónica.

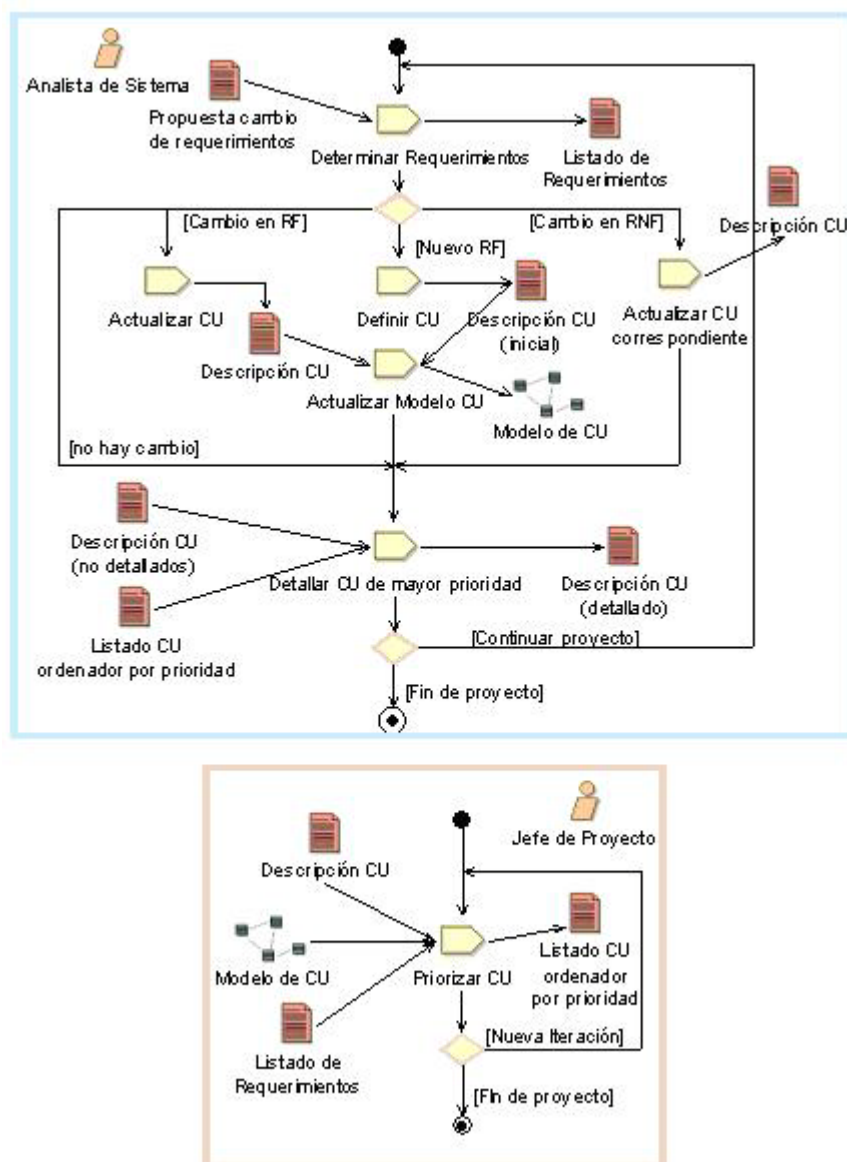


Figura 2 Secuencias de actividades de la Gestión de Requerimientos

Diseño

El diseño debe transformar los casos de uso en especificaciones del sistema formuladas como diagramas de clases y de componentes. En la disciplina de diseño que se propone intervienen los roles de: Ingeniero de casos de uso, Jefe de proyecto e Ingeniero de componentes. La descripción que se presenta (ver figura 3) está conducida por un caso de uso y se circunscribe a una iteración, partiendo del principio de que en cada iteración se hace, como parte

de la gestión del proceso, una planificación de los casos de uso en que se trabajará.

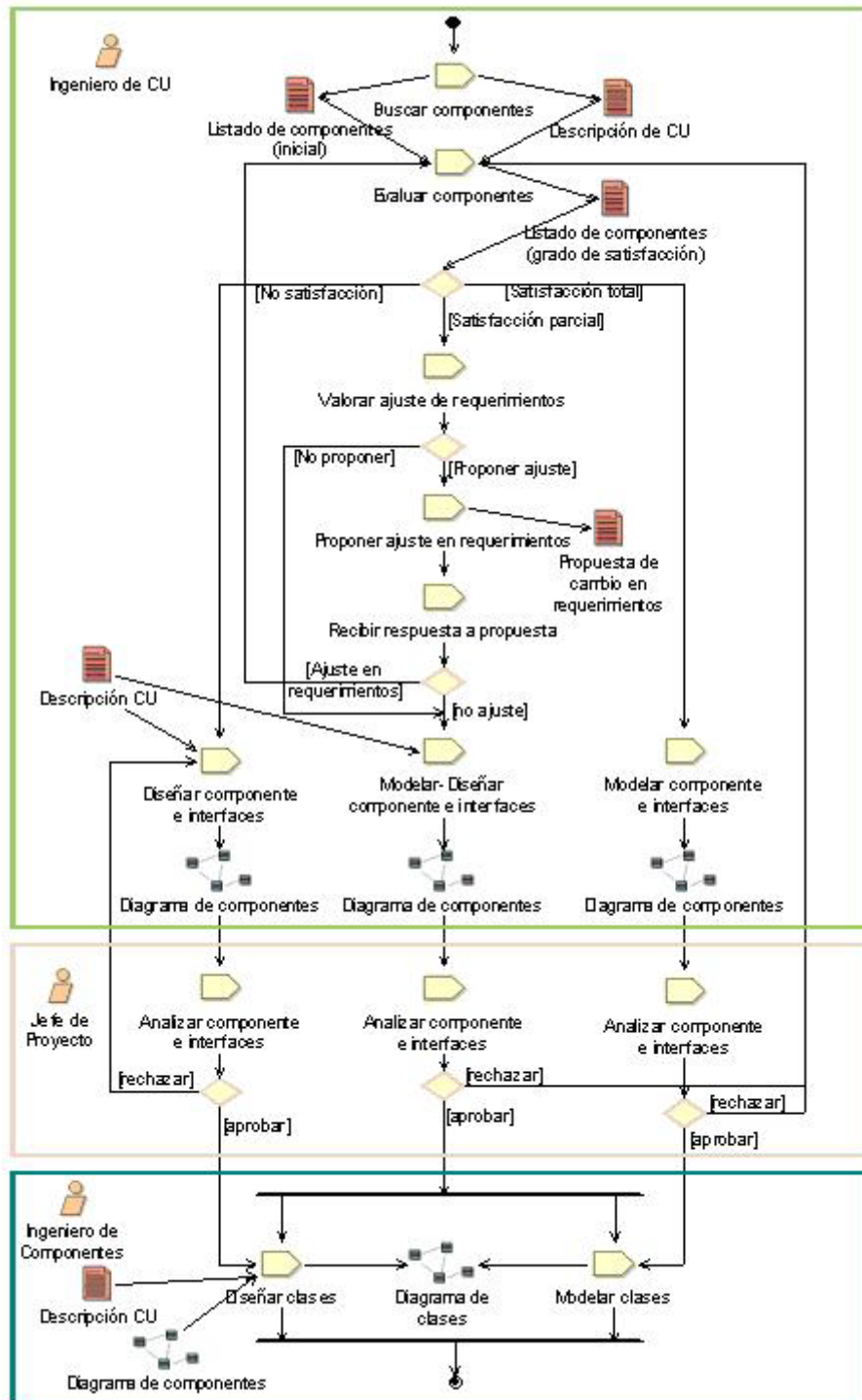


Figura 3 Secuencia de actividades de la disciplina Diseño.

Para cada caso de uso, el diseño comienza con una búsqueda de componentes que satisfagan las especificaciones del mismo, esta la realiza el Ingeniero de casos de uso correspondiente y debe comenzar por aquellos que ya estén siendo usados en el sistema, continuar por los que se hayan

desarrollado por el propio equipo para otros sistemas y finalizar con los disponibles en Internet desarrollados por terceros. Es aconsejable no invertir mucho esfuerzo evaluando cada componente, sólo se pretende en esta actividad conseguir un conjunto de candidatos que aparentemente puedan servir para el caso de uso.

Una vez que se tiene un conjunto de componentes lo suficientemente prometedor, o se ha invertido demasiado esfuerzo en la búsqueda como para seguir en ella, estos deben ser evaluados minuciosamente en cuanto al grado en que satisfagan la especificación del caso de uso. Esta evaluación puede arrojar tres situaciones:

1. Ningún componente satisface la especificación del caso de uso con un grado aceptable (superior a un umbral mínimo que fije el equipo).
2. Un componente satisface parcialmente la especificación del caso de uso con un grado superior al umbral mínimo fijado.
3. Un componente satisface totalmente la especificación del caso de uso.

En el primer caso, se procede a desarrollar el diseño de un componente para dar respuesta al caso de uso. Este será un diseño con un nivel alto de abstracción, enfocado fundamentalmente hacia la especificación de la estructura de ficheros necesaria, un esbozo del modelo de datos, la identificación de dependencias hacia otros componentes y algunas de las API que ofrece Joomla como plataforma, así como las API que debe ofrecer si es que fuera necesario. Es muy útil diseñar componentes lo más parametrizables y flexibles posible, pues aunque esto implica un esfuerzo adicional, aumenta las posibilidades de reutilización, tanto en el propio proyecto como en otros posteriores, elevando a la larga la eficiencia del equipo de desarrollo.

Este diseño debe ser analizado por el jefe de proyecto, quien desde una perspectiva más global decide finalmente si es arquitectónicamente adecuado o no. De ser rechazado, el Ingeniero de casos de uso debe adecuarlo para satisfacer las exigencias que se hayan especificado, repitiéndose este proceso hasta que el jefe de proyecto lo apruebe, momento a partir del cual la responsabilidad pasa al Ingeniero de Componentes.

En el segundo caso, se debe valorar la posibilidad de negociar cambios en los requerimientos para hacer un buen uso del componente disponible. (Stevens, 2002). De decidirse esta opción, ha de redactarse una propuesta de cambio de requerimientos, donde se fundamente la conveniencia de tal proceder, la que se remitirá al Analista de Sistema, quien como parte de la gestión de requerimientos le dará curso con el cliente. Si la respuesta es positiva, se impone una reevaluación de los componentes; en caso contrario, se procede a modelar el componente seleccionado de acuerdo a sus características actuales, incorporándose a este modelo elementos de diseño que describan las adaptaciones que se hace necesario realizarle. Al igual que en el caso anterior el Jefe de Proyecto debe aprobar el modelo, la diferencia fundamental estriba en que de ser rechazado no se rediseña, sino que se descarta y se vuelve a la evaluación de componentes. Esto se debe a que generalmente es poco factible

realizar cambios radicales en el diseño de un componente previamente desarrollado.

En el tercer caso, se modela el componente seleccionado de acuerdo a sus características actuales, sin incorporar ningún elemento nuevo de diseño, debido a que la intención es utilizarlo tal y como está. Este modelo será analizado por el Jefe de Proyecto, quien finalmente aprobará o rechazará el uso del componente.

En los tres casos, una vez aprobado el diseño o modelo del componente, le corresponde al Ingeniero de Componentes, diseñar o modelar las clases que lo integran. Este es ya un diseño más refinado, donde se incorporan todos los elementos con un alto nivel de detalle, siempre respetando todas las especificaciones definidas por el Ingeniero de Caso de Uso. Es importante tener en cuenta, como se verá más adelante, que el propio Ingeniero de Componentes será el responsable de la implementación de cada una de estas clases; por lo que en este punto, las disciplinas de diseño e implementación están estrechamente relacionadas.

Implementación

La implementación debe producir a partir de los elementos del diseño, componentes funcionales del sistema, en este caso archivos php, Javascript y html fundamentalmente. Se propone que la implementación de cada componente sea llevada a cabo por el propio Ingeniero de Componentes que trabaje en su diseño, esto reduce la necesidad de comunicaciones y documentaciones que muchas veces no perduran más allá de la propia iteración y hasta entorpecen su normal desarrollo (ver figura 4).

En la actividad de implementar clases, se trabaja con dos elementos de entrada fundamentales: el diagrama de clases, donde aparece el diseño; y los reportes de errores en caso de estar sometiendo las clases a mantenimiento. Es difícil describir la disciplina de implementación en un diagrama de actividades, pues tiene un carácter muy dinámico, muy personal e interactúa mucho con otras disciplinas, fundamentalmente el diseño y la prueba.

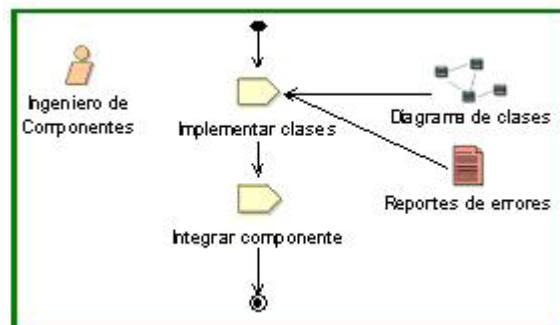


Figura 4 Secuencia de actividades de la disciplina Implementación.

Prueba

El objetivo de la prueba es garantizar la calidad del producto, y esto no se logra solo evaluando el producto final, sino que debe ser un proceso continuo a lo largo de todo el ciclo de vida. El principal criterio de calidad es que el producto satisfaga las necesidades del cliente, y como estas están expresadas a través de los casos de uso, son estos entonces una guía idónea para dirigir la prueba (ver figura 5).

Dentro de cada iteración, una vez que un caso de uso es implementado, corresponde al Ingeniero de Caso de Uso realizar las pruebas necesarias para verificar que la implementación satisface la especificación. Si se encuentran errores, se genera un reporte donde se describe el contexto en que estos ocurrieron y se detiene temporalmente la prueba hasta tanto sean subsanados los errores.

Una vez que la implementación alcanza una calidad tal que el Ingeniero de Caso de Uso no detecta ningún error, entonces está lista para ser distribuida a los usuarios en una versión Beta. Estos a su vez la someterán a las pruebas que estimen conveniente y serán los que finalmente acepten o no la versión como estable.

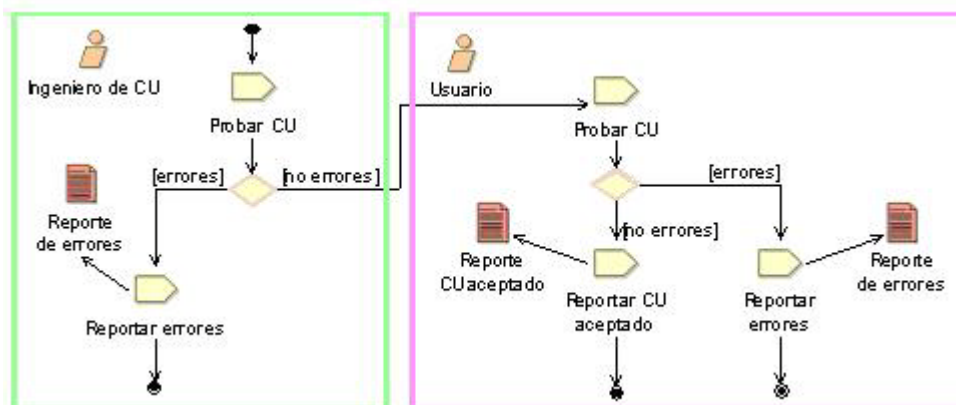


Figura 5 Secuencia de actividades de la disciplina Prueba

CONCLUSIONES:

Este trabajo ha presentando una propuesta de adaptación del Proceso Unificado de desarrollo de software para proyectos basados en el CMS Joomla. Se describieron las cinco disciplinas y cinco roles que se consideran adecuados a las características de este tipo de proyecto. Se ha hecho énfasis en el proceso, no así en los modelos, que salen del ámbito de los objetivos de este trabajo.

Aunque no está entre los objetivos iniciales se considera que es posible extender esta propuesta para diversas modalidades de sistemas Web de código libre basados en componentes.

RECOMENDACIONES

Se considera que los resultados de este trabajo dejan abiertas las siguientes direcciones de trabajo:

- Presentar una propuesta complementaria, que enfatice en los artefactos que el proceso debe producir.
- Estudiar la factibilidad del proceso propuesto para diversas modalidades de sistemas Web de código libre basados en componentes.

BIBLIOGRAFÍA:

1. World Multiconference on Systemics, Cibernetics and Informatics: Proceedings (5.: 2001: Orlando , Florida , US). Software Process Modelling / Silvia T. Acuña, Xavier Ferré. p 1-6.
2. Service- and Component-based Development: Using Select Perspective™ and UML / Hedley Apperly... [et al.] US: Ed. Addison Wesley, 2003. 240 p.
3. Booch, Grady. El Lenguaje unificado de modelado / Grady Booch, James Rumbaugh , Ivar Jacobson. New York : Ed. Prentice Hall, 2003. 457 p.
4. Crnkovic, Ivica. Building Reliable Component-Based Software Systems / Ivica Crnkovic, Magnus Larsson. US: Ed. Artech House, Inc, 2002. 386 p.
5. Cuerda García, Xavier. Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto / Xavier Cuerda García, Julià Minguillón Alfonso. **Mosaic** 2004, 36 dic. [seriada en línea] www.uoc.edu/mosaic/articulos/cms1204.html [consultado: 22 mar. 2006].
6. Curtis, Kellner. Over, Process modelling. **Communications of the ACM** 35 (9), 75-90. sep. 1992.
7. Jacobson, Ivar. El Proceso Unificado de Desarrollo de Software / Ivar Jacobson, Grady Booch, James Rumbaugh. New York : Ed. Addison Wesley, 2000. 304 p.
8. Kroll, Per. The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP / Per Kroll, Philippe Kruchten. US: Ed. Addison Wesley. 2003. 464 p.
9. Kruchten, Philippe. The Rational Unified Process An Introduction. Massachusetts , US: Ed. Addison Wesley. 2000. 320 p.
10. Lonchamp, Jacques. Open Source Software Development Process Modelling. **En** Software Process Modelling. New York : Ed. Springer, 2005. p 1-35.
11. Pressman, Roger S. Ingeniería del Software, un Enfoque Práctico. Madrid : Ed. Mac Graw Hill, 2001. 601 p.
12. SPEM: Software Process Engineering Metamodel. Version 1.1. US; OMG, 2005. 99 h.
13. Stevens, Perdita. Utilización de UML en Ingeniería del Software con objetos y componentes / Perdita Stevens, Rob Pooley . Madrid: Ed. Addison-Wesley Iberoamericana, 2002. 291 p.

DATOS DE LOS AUTORES:

Nombre:

Lic. Enrique José Leyva Miranda. Profesor Instructor
Ing. Mileidys González Prieto. Profesora Instructora

Correo:

eleyvam@facinf.uho.edu.cu
milo@facinf.uho.edu.cu

Centro de trabajo:

Facultad de Informática y Matemática, Universidad de Holguín “Oscar Lucero Moya”