Proceso de Construcción de Aplicaciones de Software Libre

Fase de Análisis y Diseño

Esta fase comprende la definición de la arquitectura del software, la especificación de los datos persistentes y el diseño de la interfaz de usuario. En el caso de la arquitectura es conveniente destacar que ésta constituye un aspecto a considerar, con mayor importancia, para aplicaciones de software que resulten complejas en términos del número de requerimientos y/o el impacto de los mismos (Hofmeister et al., 2000).

La arquitectura del software, la especificación de datos persistentes y el diseño de interfaz de usuario pueden construirse de forma incremental en cada iteración de desarrollo, con base a los requerimientos funcionales y no funcionales que se aborden en cada iteración.

A continuación se indican las actividades y tareas que componen esta fase.

Actividad: Especificación de datos persistentes

Tarea: Modelar los datos persistentes que maneja el software.

Recomendaciones:

- Para modelar los datos persistentes se debe seleccionar el tipo de modelo en el que se realizará el almacén de datos. Entre los tipos de modelos mas utilizados se encuentran: orientado a objetos, entidad-relación, basado en documentos. Es posible realizar transformaciones entre estos modelos.
- Los objetos o entidades que maneja el software se pueden identificar en la descripción textual de los casos de uso.
- Los diagramas de clases elaborados como parte de la arquitectura del software constituyen en sí un modelo de datos, en caso de utilizar el modelo de datos orientado objetos.
- En los casos en los que se trabaje con el modelo de datos relacional se requiere modelar los datos del software en base al modelo entidad-relación.
- En la medida de lo posible, se recomienda utilizar tipos básicos para los campos de las relaciones, clases o documentos, de tal manera que la aplicación de software admita el uso de diversos gestores de datos, que se ajusten a las necesidades del despliegue.
- Cuando se utilizan herramientas (editores gráficos) para modelar los datos es necesario tener en cuenta algunas pautas tales como configurar

la salida de los scripts en SQL estándar, y mantener actualizado el modelo cuando ocurren cambios en los scripts de creación de la base de datos.

- Se debe tener como premisa que la aplicación podrá cambiar de gestor de datos, incluso de modelo de datos, por lo tanto, es conveniente ceñirse a estándares y patrones de base de datos, como por ejemplo, los definidos en SQL, en el Modelo-Vista-Controlador o en ORM (Object-Relational Mapping).

Herramientas:

- Para elaborar diagramas de entidad-relación se pueden utilizar herramientas como: DIA, ER Visual, BD Designer Fork (principalmente para trabajar con base de datos MySQUL), Druid.
- El modelo de datos del software puede registrarse en la planilla Modelo de datos persistentes (Base de datos).

Productos:

- Modelo de datos persistentes.

Responsables:

- Analistas y/o el Arquitecto de Software.

Tarea: Especificar los datos a intercambiar con otras aplicaciones de software.

Recomendaciones:

- La interoperabilidad entre aplicaciones de software constituye un atributo de calidad a considerar en el desarrollo de aplicaciones que requieran el intercambio de datos con otras aplicaciones, razón por la cual es necesario especificar los datos a intercambiar, incluyendo en ello la definición de los formatos a utilizar para realizar dicho intercambio.

Herramientas:

- La especificación de datos a intercambiar puede registrarse en en la planilla Modelo de datos persistentes (Base de datos).

Productos:

- Especificación de datos a intercambiar.

Responsables:

- Analistas y/o el Arquitecto de Software.

Actividad: Definición de la arquitectura del software

Tarea: Definir la arquitectura del software.

Recomendaciones:

- Al momento de definir la arquitectura del software, es decir, sus componentes y la interacción (comunicación) entre éstos, se requiere tener en cuenta no solo las funcionalidades del software y las limitaciones tecnológicas existentes, sino también los atributos de calidad asociados al software, pues la arquitectura que se defina puede inhibir o facilitar el cumplimiento de dichos atributos (Bass et al., 1998). Por ejemplo, el atributo de calidad respectivo al desempeño del software depende de los componentes de se definan para éste y de su ubicación en los procesadores, así como de los caminos de comunicación entre estos componentes, etc.
- La descripción textual de los casos de uso constituyen el insumo base para identificar los componentes del software.
- Para el desarrollo de aplicaciones de software complejas y de gran escala se recomienda utilizar patrones y/o estilos arquitectónicos, así como patrones de diseño, pues éstos permiten mejorar la calidad de estas aplicaciones. Para la selección de patrones y estilos es necesario tener presente que éstos pueden facilitar el cumplimiento de ciertos atributos de calidad en un software, pero a su vez pueden inhibir el cumplimiento de otros atributos. Por ejemplo, el patrón arquitectónico Modelo-Vista-Controlador facilita el cumplimiento de atributos de calidad asociados a la funcionalidad y la mantenibilidad, pero dificulta el cumplimiento de atributos de calidad asociados al desempeño y a la portabilidad del software (Buschmann et al., 1996).
- Es importante que se estudien varias alternativas de arquitecturas, de modo que se pueda seleccionar de éstas aquella que permita cumplir en mayor medida con los requerimientos funcionales y los atributos de calidad establecidos para el software, que facilite futuras modificaciones al mismo y que sea factible conforme las limitaciones tecnológicas que puedan existir.
- La arquitectura de un software puede ser representada a través de diferentes diagramas o vistas arquitectónicas, las cuales constituyen las diferentes perspectivas del diseño de una aplicación (Kruchten, 1999). Entre los diagramas utilizados para representar vistas arquitectónicas se encuentran: diagramas de clases, diagramas de secuencia, diagramas de estado, diagramas de componentes, entre otros. Cabe destacar que no es obligatorio diseñar todas las vistas arquitectónicas de un software, basta con plantear aquellas que se consideren pertinentes según la complejidad y alcance del software, que permitan contar con una visión de éste que sirva de apoyo en las diferentes fases de construcción del mismo, así como en sus procesos de mantenimiento.

Herramientas:

- Existen varias herramientas para elaborar los diagramas de la arquitectura de un software, entre ellas: Dia, Umbrello, Bonita, CASEUML, ArgoUML, BOUML, entre otras.
- En el caso de la Fundación Cenditel se plantea utilizar para elaborar los diagramas de arquitectura de un software la herramienta Platuml contenida en el plugin de la plataforma Trac desarrollado para la metodología. Entre los diagramas que se pueden elaborar con esta herramienta se encuentran: diagramas de clases, de secuencia, de estado, entre otros.
- Los diagramas en base a los cuales se represente la arquitectura del software pueden registrarse en la plantilla Arquitectura de software.

Productos:

- Diagramas que representan la arquitectura del software.

Responsables:

- Arquitecto de Software.

Colaboradores:

- Analistas, otros integrantes del Equipo de Desarrollo.

Actividad: Diseño del prototipo no funcional de la interfaz de usuario

Tarea: Diseñar las pantallas no funcionales correspondientes a las interfaces gráficas del software.

Recomendaciones:

- Por lo general toda interfaz gráfica se compone de los siguientes elementos (Baéz, Castañeda, Castañeda, 2005): a) Encabezado, éste se refiere a un logo o imagen de identificación del software. Se recomienda utilizar frames para que el encabezado se cargue solo una vez en las pantallas de

interfaz del software. b) Menú, en éste se muestra el listado de las funcionalidades que ofrece el software, se recomienda que el menú se ubique, de ser posible, en varios lugares de las pantallas de interfaz. c) Zona de contenido, esta muestra las operaciones que puede ejecutar el usuario conformes a las funcionalidades que ofrece el software. d) Zona de mensajes, en esta se muestran los mensajes de tipo informativo, de error y de éxito en una operación del software.

- Se recomienda utilizar un diagrama de navegabilidad entre las pantallas diseñadas para representar la interfaz de usuario, con lo cual se podrá mostrar el paso de una pantalla a otra.
- Teniendo en cuenta la importancia que adquiere la interfaz de usuario, en términos de usabilidad, se plantean a continuación algunas recomendaciones a considerar para el diseño de la interfaz gráfica: a) La interfaz de las operaciones que ejecuta el software debe mantener un estándar visual. Por ejemplo, de ser posible, las funciones para modificar o eliminar información deben seguir un mismo esquema de presentación para las distintas funciones u operaciones del software en las cuales se requieran éstas. b) La estructura de iconos o botones que sirven de enlace a las funciones que ejecuta el software debe ser lo más sencilla posible, es decir, se debe evitar la ejecución de varios pasos a efectuar en el software para acceder a alguna de sus funciones. c) La interfaz debe mantener una estandarización en relación al formato de los iconos o botones mostrados. d) Los botones o iconos utilizados en la interfaz pueden mostrar textos en los cuales se indique que función cumple cada uno de éstos. Para ello se recomienda hacer uso de los tool tips. e) Los tipos y tamaños de las letras utilizadas en las pantallas deben facilitar la visualización de los textos o frases que se presentan en la interfaz. f) Los colores utilizados en cada pantalla deben ser contrastantes entre sí, a fin de facilitar la lectura de la información mostrada en la interfaz. g) La interfaz debe mantener una estandarización en relación al idioma de las personas que usarán el software.

Herramientas:

- La herramienta Pencil se puede utilizar para elaborar las pantallas del prototipo no funcional de la interfaz de usuario.
- Este prototipo puede registrarse en el plantilla Prototipo no funcional de la interfaz de usuario.

Productos:

- Prototipo no funcional de la interfaz de usuario.

Responsables:

- Diseñador Gráfico.

Colaboradores:

- Equipo de Desarrollo, Usuarios.

Tarea: Validar el prototipo no funcional de la interfaz con los usuarios.

Productos:

- Prototipo no funcional de la interfaz validada por los Usuarios.

Responsables:

- Diseñador Gráfico y Usuarios.

volver a metodología