



Instituto Nacional
de Tecnologías
de la Comunicación

GUÍA AVANZADA DE GESTIÓN DE REQUISITOS

LNCS

AVISO LEGAL

- CMMI® es una marca registrada en la Oficina de Marcas y Patentes de EEUU por la Universidad Carnegie Mellon.
- Las distintas normas ISO mencionadas han sido desarrolladas por la International Organization for Standardization.
- SWEBOK® (Software Engineering Body of Knowledge) es una marca registrada por el IEEE (Institute of Electrical and Electronics Engineers).

Todas las demás marcas registradas que se mencionan, usan o citan en la presente guía son propiedad de los respectivos titulares.

INTECO cita estas marcas porque se consideran referentes en los temas que se tratan, buscando únicamente fines puramente divulgativos. En ningún momento INTECO busca con su mención el uso interesado de estas marcas ni manifestar cualquier participación y/o autoría de las mismas.

Nada de lo contenido en este documento debe ser entendido como concesión, por implicación o de otra forma, y cualquier licencia o derecho para las Marcas Registradas deben tener una autorización escrita de los terceros propietarios de la marca.

Por otro lado, INTECO renuncia expresamente a asumir cualquier responsabilidad relacionada con la publicación de las Marcas Registradas en este documento en cuanto al uso de ninguna en particular y se eximen de la responsabilidad de la utilización de dichas Marcas por terceros.

El carácter de todas las guías editadas por INTECO es únicamente formativo, buscando en todo momento facilitar a los lectores la comprensión, adaptación y divulgación de las disciplinas, metodologías, estándares y normas presentes en el ámbito de la calidad del software.

- **Llevar a cabo una demostración:** Proporcionar al usuario final una visión general del sistema que al que pertenece el prototipo y la demostración del prototipo.
- **Observar cómo los participantes usan el prototipo:** Observar a los participantes puede cubrir problemas de aprendizaje de los que un usuario final podría no informar en una entrevista. Funciona de forma más efectiva cuando se simulan tareas reales y se involucra a los usuarios finales.

7.5. HERRAMIENTAS DE PROTOTIPADO

Una herramienta de prototipado debería:

- Dar soporte de desarrollo basado en componentes, por ejemplo permitir la definición y soporte de objetos de código reutilizables
- Proporcionar un entorno integrado para el desarrollo basado en repositorio que soporta el desarrollo de modelos de análisis y diseño del código del programa actual
- Ser flexible, intuitiva y fácil de usar
- Proporcionar un entorno de programación intuitivo y visual
- Incluir soporte para depuración y pruebas
- Permitir realizar cambios en los prototipos de forma rápida y fácil
- Proporcionar herramientas de diseño de base de datos que soporten el desarrollo interactivo de prototipos de base de datos
- Soportar el desarrollo interactivo para acceso a datos (p.ej.: SQL)
- Soportar el volumen requerido de usuarios y transacciones con tiempos de respuesta razonables
- Concordar con la infraestructura técnica
- Proporcionar soporte para la generación automática para múltiples plataformas hardware, si es necesario

Ejemplos de herramientas de prototipado:

- Sistema de gestión de base de datos relacional (RDBMS)
- Generadores de pantallas
- Generadores de menús
- Programación visual
- Lenguajes de 4ª generación

7.6. BENEFICIOS Y DIFICULTADES

Una vez que se ha visto cómo desarrollar un prototipo, a continuación se van a describir cuáles son algunos de los principales beneficios y dificultades en el uso de prototipos.

Beneficios

El uso de prototipos se debería usar para todos los sistemas interactivos. Hay que considerar que una revisión seria por los usuarios finales puede resultar casi siempre en un cambio. Algunos de los beneficios que se obtienen son:

- Los prototipos fundamentalmente **ayudan en la comunicación** entre el usuario final y las personas encargadas del desarrollo; la mejora en la comunicación resulta en una **exactitud mayor** y en un **descenso de los errores** en las partes posteriores del proyecto, cuando son más caros de resolver. La participación temprana otorga poderes a los usuarios finales y facilita la aceptación y entendimiento del sistema.
- La generación de prototipos es un medio efectivo de comunicación del entendimiento del analista de los requisitos del sistema al usuario final. El usuario final consigue un **mejor entendimiento** del sistema propuesto que el que obtendría de una documentación escrita.
- La generación de prototipos es un medio efectivo de comunicación de los requisitos de sistema y el diseño a los programadores. Es **más fácil** para el programador **construir** un sistema desde un modelo de funcionamiento que desde un documento de diseño.
- Algunas actividades de diseño, como el diseño de la interfaz, se realizan en etapas tempranas del ciclo de vida cuando se desarrolla el prototipo. Los usuarios finales son capaces de probar el diseño y proporcionar sus entradas antes de que el tiempo y el dinero se hayan expandido. Si se usan prototipos horizontales durante el análisis, una primera parte de los componentes externos se definen en ese momento. Si se desarrolla un prototipo vertical, se puede diseñar una parte de la base de datos durante el análisis.
- El uso de prototipos puede **reducir enormemente el alcance y tamaño** de las actividades de diseño. Las actividades de diseño siguen teniendo lugar pero en otras etapas del ciclo de vida del sistema. Dependiendo de la extensión del prototipo, se pueden reducir los esfuerzos de diseño para incluir sólo tareas como preparación del plan de pruebas, diseño de ayudas de usuario, diseño de conversión y optimización de base de datos.
- Los **usuarios** finales se pueden **involucrar más** en el proceso de desarrollo del sistema cuando se usan prototipos. Esto ayuda a asegurar que los requisitos de usuario se cumplen.
- Los prototipos hacen a un **sistema** ser **más intuitivo** y **reduce** la cantidad de **documentación** escrita requerida.
- Los prototipos **animan** y requieren la **participación activa** de los usuarios finales.

Dificultades

- La estructura de los prototipos se corrompe por cambios constantes. De ahí que sea difícil una **evaluación a largo plazo**.
- Los prototipos **requieren una participación activa** de los usuarios finales.
- Un prototipo **no puede sustituir completamente** una especificación en papel. Los prototipos deberían complementar, no reemplazar, otras metodologías.
- Numerosas cuestiones de diseño no son y no pueden ser documentadas de forma adecuada mediante el uso de prototipos; estas cuestiones se pueden olvidar sin querer.
- Los prototipos con frecuencia conducen a un **acuerdo prematuro del diseño físico**.

8. GLOSARIO

- **Analista de requisitos:** El analista de requisitos es un rol. Es la persona que se encarga de capturar los requisitos en modelos de caso de uso, casos de uso y especificaciones suplementarias. Facilita la definición del alcance y los detalles de los requisitos, describe las relaciones y dependencias entre requisitos y proporciona un vocabulario común.
- **Atributos de calidad:** Exposiciones que indican cómo de bien realiza el sistema algunos comportamientos. Son un tipo de requisitos no funcionales. Pueden ser características deseables como: rápido, fácil, intuitivo, robusto, fiable, seguro y eficiente. Hay que trabajar con los usuarios para definir de forma precisa que quieren decir con este tipo de términos que suelen ser ambiguos y subjetivos.
- **Brainstorming:** El término 'brainstorming' hace referencia a un proceso de pensamiento lateral y es una excelente forma de desarrollar distintas soluciones creativas para un problema. Consisten en centrarse en un problema y presentar todas las soluciones posibles. Se diseña para ayudar a romper con unos patrones de pensamiento y encontrar nuevas formas de mirar a ese problema. Intenta abrir posibilidades y desechar suposiciones erróneas sobre el límite del problema y no contempla críticas a las ideas propuestas. Todas las ideas se evalúan una vez que la sesión de brainstorming ha finalizado y estas soluciones se pueden explorar o analizar usando enfoques convencionales.
- **Capacidad de prueba:** Es un aspecto medible de los requisitos para comprobar si la solución se corresponde con el requisito original.
- **Casos de uso:** Son declaraciones generales de objetivos de usuario o tareas de negocio que son necesarios para realizar el sistema, mientras que las descripciones de tareas específicas representan escenarios de uso.
- **Control de cambios:** El control de cambios es un proceso formal que se utiliza para asegurar que un producto, servicio o proceso sólo se modifica de acuerdo a un cambio necesario identificado.
- **Desarrollo de requisitos:** El desarrollo de requisitos implica entender los requisitos de negocio, obtener los requisitos de usuario y trasladar los requisitos de negocio y de usuario a requisitos software/de sistema.
- **Escenarios:** Un escenario es una breve descripción de un evento. Se usa para comunicar una idea de un producto o experiencia que implica interactividad. Un escenario es también un informe o una sinopsis de una acción, evento o situación en curso. El desarrollo de escenarios se usa en la planificación y obtención de requisitos.
- **Gestión de cambios:** La gestión de cambios es el proceso de desarrollar un proceso de petición de cambio planificado en una organización. Típicamente el objetivo es

maximizar los esfuerzos colectivos de toda la gente implicada en el cambio y minimizar el riesgo de fallo a la hora de implementar el cambio.

- **Gestión de requisitos:** La gestión de requisitos implica gestionar los cambios de los requisitos y mantener la consistencia entre los requisitos y otros productos de trabajo del proyecto. Es el proceso de encontrar, documentar, organizar y hacer un seguimiento de los cambios de los requisitos de un sistema.
- **Línea base:** una instantánea o un estado específico de un conjunto de productos de trabajo que han sido formalmente revisados y aprobados. Aunque el estado se actualice más adelante, siempre a través de un procedimiento de control de cambios, la línea base permanece constante y disponible como referencia del estado original para poder compararlo con el estado actual.
- **Obtención de requisitos:** El proceso de identificar las necesidades y salvar las disparidades entre las comunidades involucradas en el propósito de definir y destilar los requisitos para cumplir con las restricciones de estas comunidades.
- **Priorización de requisitos:** La priorización de requisitos es el proceso de asignar prioridades a los requisitos, basándose en sus beneficios esperados y en la importancia que tienen para la gente implicada en el proyecto, de tal forma que los que tengan prioridades mayores puedan implementarse primero, como parte de un ciclo de desarrollo incremental, iterativo...
- **Prototipos:** Son simulaciones de una aplicación que permite a los usuarios visualizar la aplicación que no está aun construida. Los prototipos ayudan a los usuarios a tener una idea de cómo va a ser el sistema y facilita la toma de decisiones relacionadas con el diseño sin esperar a que éste se haya construido. Los prototipos proporcionan a los desarrolladores de software un modelo de trabajo y pueden usarse por los clientes, analistas de negocio o gerentes para confirmar o realizar cambios en los requisitos, ayudar a definir interfaces, desarrollar componentes de colaboración y proporcionar unos mejores acuerdos.
- **Puerta de calidad:** es un punto por el que pasa cada uno de los requisitos antes de formar parte de la especificación de requisitos. Las puertas de calidad se aseguran de que cada requisito cumple con el criterio que tiene asignado.
- **Requisito:** Un requisito es una condición o capacidad con la que ha de cumplir el sistema. Es decir, algo que el producto debe hacer, o una propiedad que el producto debe tener, y que es necesaria para las personas involucradas en el negocio. Hay varios tipos de requisitos como pueden ser funcionales, de usabilidad, de fiabilidad, de rendimiento, de mantenimiento...
- **Requisitos de fiabilidad:** Requisitos que describen la fiabilidad y robustez del sistema.
- **Requisitos de negocio:** una descripción a alto nivel de lo que el sistema debe hacer. Representan los objetivos, la base del negocio, estrategias, visión, alcance y el valor esperado del desarrollo del software. Los requisitos de negocio proporcionan

la dirección que ha de seguir el proyecto y forman la base de los requisitos de usuario.

- **Requisitos de rendimiento:** Incluyen tiempos de respuesta, requisitos de precisión, de capacidad, de escalabilidad, de seguridad...Especifican quién tiene acceso autorizado al producto, y bajo qué circunstancias se concede ese acceso y a qué partes del sistema/aplicación.
- **Requisitos de sistema:** contienen los requisitos funcionales y no funcionales del sistema a un alto nivel de arquitectura. Definen las funcionalidades y características que hay que construir en el sistema/software para satisfacer los requisitos de negocio y de usuario. Esto sirve como fuente para una arquitectura, diseño y planes de pruebas detallados.
- **Requisitos de usabilidad:** Incluyen facilidad de uso, personalización e internacionalización, requisitos de accesibilidad, requisitos de usuario... Proporcionan más detalle de los requisitos de negocio; son la descripción de las tareas para que sean ejecutadas de forma correcta por el software cuando se trabaja con él. Estos requisitos describen la funcionalidad necesaria para satisfacer tareas específicas, necesidades operativas y grupos de usuario.
- **Requisitos de usuario:** entran en más detalle en los requisitos de negocio. Son una descripción de las tareas que ha de ejecutar de forma adecuada el software cuando el usuario opera con él. Estos requisitos describen la funcionalidad necesaria para satisfacer tareas específicas, necesidades operacionales y grupos de usuarios.
- **Requisitos del cliente:** Definen e identifican los requisitos del cliente, para incluir la voz del cliente, los datos, las expectativas convertidas en expresiones medibles que se usan para asegurar que se cumple con las necesidades del cliente.
- **Requisitos funcionales:** Los requisitos funcionales describen el comportamiento observable que el sistema exhibirá, con frecuencia en el contexto de una secuencia de acción del actor – respuesta del sistema. Los requisitos funcionales definen lo que el sistema hará; forman la mayor parte de la especificación de requisitos.
- **Requisitos no Funcionales:** Los requisitos no funcionales son requisitos que especifican criterios que pueden ser usados para describir la operación de un sistema más que el comportamiento específico. Los requisitos no funcionales son aquellos que describen las características globales. Los requisitos no funcionales típicos son la fiabilidad, escalabilidad, etc.
- **Requisitos tecnológicos:** un requisito que es necesario sólo debido a la tecnología elegida. Su objetivo no es satisfacer una necesidad directa del cliente.
- **Restricciones:** Las restricciones son condiciones que limitan las elecciones disponibles al diseñador o programador. Son requisitos globales. Pueden ser restricciones del propio proyecto o del diseño del producto.
- **Trazabilidad:** La trazabilidad es la capacidad de enlazar un elemento del proyecto con otro elemento del proyecto relacionado, especialmente los relacionados con los requisitos. Los elementos del proyecto involucrados en la trazabilidad se denominan



Instituto Nacional
de Tecnologías
de la Comunicación

elementos de trazabilidad. Típicamente los elementos de trazabilidad incluyen diferentes tipos de requisitos, elementos de análisis y diseño, artefactos de prueba (baterías de pruebas, casos de pruebas, etc.), así como documentación de soporte y material de formación.

9. REFERENCIAS

A. Abran, J.W. Moore, P. Bourque, R. Dupuis, *Guide to the Software Engineering Body of Knowledge*, IEEE Computer Society, 2004.

Gilb, Tom. *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using PLanguage*. Butterworth-Heinemann, 2005

K.E. Emam, J.N. Drouin, W. Melo, *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, IEEE Computer Society Press, 1998. Leffingwell, Dean, and Don Widrig. *Managing Software Requirements: A Use Case Approach*. Second edition. Addison-Wesley, 2003

M.B. Chrissis, M. Konrad, S. Shrum, *CMMI® Second Edition. Guidelines for Process Integration and Product Improvement*, Addison-Wesley, 2007.

R.S. Pressman, *Software Engineering: A Practitioner's Approach*, Sixth ed., McGraw-Hill, 2004.

Sommerville, Ian and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, 1998

Suzanne Robertson y James Robertson, *"Mastering the Requirements Process"*, Segunda edición (2006)