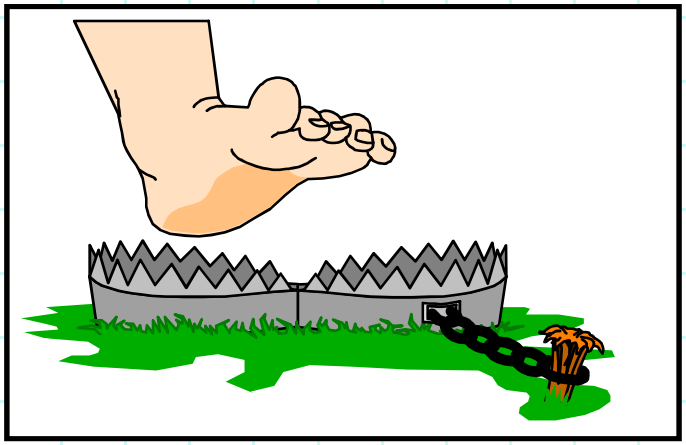
Requerimientos del Software

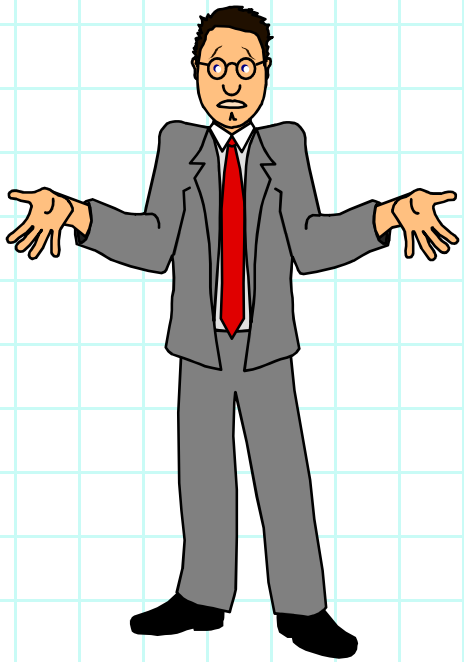
8 Trampas a evitar



Trampa #1: Confusión sobre Requerimientos

Síntomas:

- Los stakeholders discuten los "requerimientos" sin adjetivos precisos.
- El sponsor del proyecto presenta un concepto de alto nivel como "los requerimientos".
- Las pantallas de interface con usuario son vistas como "los requerimientos".
- Los usuarios proporcionan los "requerimientos" pero los desarrolladores aun no saben que construir.
- Los requerimientos se enfocan solo en la funcionalidad.



Trampa #1: Confusión sobre Requerimientos

Soluciones:

- Adoptar templates para los tres niveles de requerimientos.
  - Requerimientos del negocio (Visión & Scope Document).
  - Requerimientos del usuario (Use Case Document).
  - Requerimientos funcionales (Software Requir. Spec.).
- Distinguir requerimientos funcionales y no funcionales.
  - Atributos de calidad, restricciones, requerimientos de interfases externas, reglas del negocio.
- Clasificar input del cliente en las 3 diferentes categorías.
- Distinguir ideas de solución, de los requerimientos en si.

Trampa #2: Participación inadecuada del Cliente

Síntomas:

- Algunas clases de usuarios son ignorados.
- Algunas clases de usuarios no tienen representante.
- Reemplazantes de usuarios pretenden hablar por ellos.
  - Gerentes/jefes usuarios.
  - Marketing.
  - Desarrolladores.
- Los desarrolladores tienen que tomar muchas decisiones.
- Los clientes rechazan el producto la 1a. vez que lo ven.

Trampa #2: Participación inadecuada del Cliente

Soluciones:

- Identificar las diferentes clases de usuarios.
- Identificar campeones de producto y tomarlos como usuarios representativos.
- Conducir/realizar focus groups.
- Identificar a los tomadores de decisión.
- Hacer que los usuarios evalúen el prototipo.
- Hacer que los usuarios representativos revisen el SRS.

Trampa #3: Requerimientos vagos/ambiguos

Síntomas:

- Los lectores interpretan un mismo requerimiento de varias maneras distintas.
- Los requerimientos omiten informacion que los desarrolladores necesitan.
- Los requerimientos no son verificables.
- El desarrollador tiene que hacer demasiadas preguntas.
- El desarrollador tiene que adivinar muchos puntos.

Trampa #3: Requerimientos vagos/ambiguos

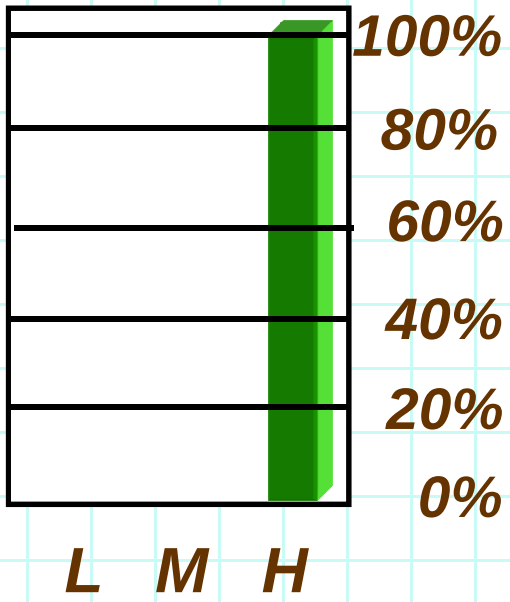
Soluciones:

- ☑ Inspeccionar formalmente los docum. de requerimientos.
- ☑ Escribir casos de prueba conceptuales contra los req.
- ☑ Modelar req. para encontrar brechas de conocimiento.
- ☑ Usar prototipos para hacer los req. mas tangibles.
- ☑ Definir los términos en un glosario.
- ☑ Evitar palabras ambiguas :
  - minimizar, maximizar, optimizar, rápido, amigable, simple, intuitivo, robusto, estado-del-arte, mejora, eficiente, flexible, varios, y/o, etc., incluir, soportar.

Trampa #4: Requerimientos no priorizados

Síntomas:

- ¡Todos los requerimientos son críticos!
- Diferentes stakeholders interpretan "alta" prioridad de manera diferente.
- ¡Después de la priorización el 95% sigue siendo alto!
- Los desarrolladores no quieren admitir que no pueden hacerlo todo a la vez.
- No esta claro que requerimientos diferir durante la "fase rápida de reducción de alcance".



Trampa #4: Requerimientos no priorizados

Soluciones:

- Alinear los requerimientos funcionales con los requerimientos del negocio.
- Alinear los requerimientos funcionales con casos de uso de alta prioridad.
  - Frecuencia de uso, clases prioritarias de usuario.
  - Core process del negocio, demandas regulatorias.
- Definir en forma no ambigua las categorías prioritarias.
- Asignar requerimientos o características a versiones.
- Priorizar analíticamente requerimientos discrecionales.

Trampa #5: Parálisis del Análisis

Síntomas:

- El desarrollo de los req. parece proseguir para siempre.
- Se emiten continuamente nuevas versiones del SRS.
- Nunca se fija una línea base para los requerimientos.
- Todos los req. se modelan nuevamente a diario.
- El diseño y la codificación no se pueden empezar hasta que el SRS sea perfecto.

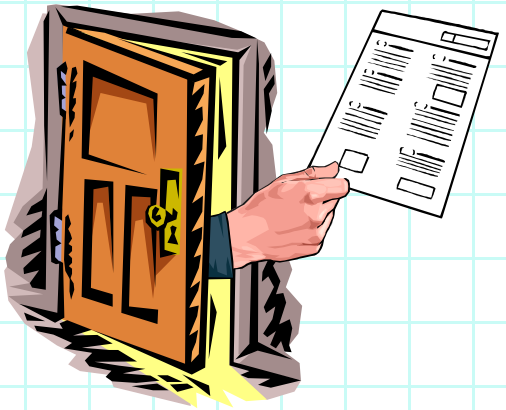
Trampa #5: Parálisis del Análisis

Soluciones:

- Recuerde : el producto es el software, no el SRS.
- Seleccione un apropiado ciclo de vida de desarrollo.
  - Versiones periódicas, prototipeo evolutivo, time-boxing.
- Decidir en que momento los requerimientos son suficientemente buenos.
  - Riesgo aceptable de proseguir con la construccion.
  - Revisión por analistas, desarrolladores, probadores, etc.
- Modelar solo las partes complejas o inciertas.
- No incluya los diseños de las interfases de usuario final en el SRS.

Trampa #6: Ampliación paulatina del alcance

Síntomas:



- Continuamente se añaden nuevos requer.
  - El schedule no cambia.
  - No se asignan recursos adicionales.
- El alcance del producto nunca se define claramente.
- Cambios a requer. se filtran a través de la puerta falsa.
- Requerimientos propuestos vienen, van, y regresan.
- Temas relativos al alcance del proyecto se debaten durante las revisiones al SRS.
- El término de esta fase no se toma en serio.

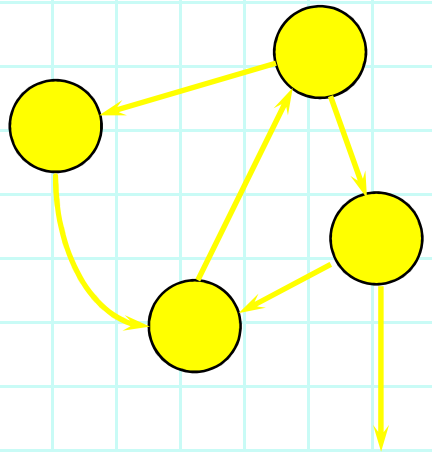
Trampa #6: Ampliación paulatina del alcance

Soluciones:

- Determine causas raíces de la ampliacion de alcance.
- Documente visión y alcance del producto.
- Defina limites e interfases del sistema.
- Siga el proceso de control de cambios para ***todos*** los cambios.
- Mejore los métodos de elaboración de requerimientos.
- Siga un importante y serio proceso de baselining.
- Renegocie compromisos cuando cambien los requerimientos.

Trampa #7: Inadecuado proceso de Cambio

Síntomas:



- No se define un proceso de cambio.
- Algunas personas se saltan el proceso de cambio.
  - Conversaciones con amigos al interior del proyecto.
  - Implementación de cambios que son rechazados.
  - Trabajo en cambios propuestos antes de su aprobación.
- Nueva funcionalidad se hace evidente durante la prueba.
- Estado poco claro de los cambios solicitados.
- Cambios no son comunicados a todos los afectados.
- No esta claro quien toma las decisiones de cambio.

Trampa #7: Inadecuado proceso de Cambio

Soluciones:

- ☑ Defina un proceso practico de control del cambios.
- ☑ Conforme un Comité de Control de Cambios.
  - Grupo diverso o multifuncional.
  - Tomar decisiones de cambio validas e inevitables.
- ☑ Utilice una herramienta para recolectar, rastrear y comunicar cambios.
  - Herramientas de rastreo de problemas, son útiles.
  - ¡Una herramienta no es un proceso!
- ☑ Establezca y refuerze políticas de control de cambios.
- ☑ Compare prioridades contra otros requer. pendientes.

Trampa #8: Inadecuado Control de Versiones

Síntomas:

- Cambios aceptados no se incorporan en el SRS.
- No se distinguen las diferentes versiones del SRS.
  - Diferentes versiones tienen la misma fecha.
  - Documentos idénticos tienen fechas diferentes.
- Las personas trabajan con diferentes versiones del SRS.
  - Implementan características anuladas o desechadas.
  - Pruebas contra requerimientos erróneos o no vigentes.
- Historia del cambio y de versiones previas de los documentos, se pierden.

Trampa #8: Inadecuado Control de Versiones

Soluciones:

- Incluya los cambios en el SRS.
- Adopte un esquema de versiones para los documentos.
- Coloque los docum. de requer. bajo control de versiones.
  - Restrinja el acceso de lectura/escritura.
  - Proporcione acceso solo lectura a todos.
- Comunicar las revisiones a todos los afectados.
- Use una herramienta de administración de requerimientos.
  - Registre la historia completa de cada requer. de cambio.
  - Convierta el SRS en un reporte mas de la base de datos.

Referencias

- Gause, Donald C., and Gerald M. Weinberg. *Exploring Requirements: Quality Before Design*. New York: Dorset House Publishing, 1989.
- Gottesdiener, Ellen. *Requirements by Collaboration: Workshops for Defining Needs.* Boston: Addison-Wesley, 2002.
- IEEE Std. 830-1998, "Recommended Practice for Software Requirements Specifications." Los Alamitos, Ca.: IEEE Computer Society Press, 1998.
- Leffingwell, Dean, and Don Widrig. *Managing Software Requirements*. Reading, Mass.: Addison Wesley Longman, 2000.
- Robertson, Suzanne, and James Robertson. *Mastering the Requirements Process*. Harlow, England: Addison-Wesley, 1999.
- Sommerville, Ian, and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. New York: John Wiley & Sons, 1997.
- Wiegers, Karl E. *Creating a Software Engineering Culture*. New York: Dorset House Publishing, 1996.
- Wiegers, Karl E. *Software Requirements*. Redmond, Wash.: Microsoft Press, 1999.

DHARMA CONSULTING

Especialistas en Project Management

Project Management Institute

R.E.P.

CONSULTORÍA Y CAPACITACIÓN EN...

SOFT SKILLS

- Habilidades Blandas para la Gestión de Proyectos.
- Habilidades de Team Building y Team Work para el Gestor de Proyectos.
- Habilidades de Gestión de Equipos para el Gestor de Proyectos.
- Habilidades de Facilitación para el Gestor de Proyectos.
- Habilidades de Liderazgo para el Gestor de Proyectos.
- Gestión del Cambio Organizacional e Individual.

GESTIÓN DE PROYECTOS, PROGRAMAS Y PORTAFOLIOS

- Gestión de Proyectos (Guía del PMBOK®).
- Estándares para la Gestión de Proyectos.
- Taller Práctico de Gestión de Proyectos.
- Taller de Preparación para la Certificación PMP®
- Gestión de Programas (PPgM).
- Gestión de Portafolios (PPfM).

GESTIÓN ORGANIZACIONAL DE PROYECTOS

- Gestión Organizacional de Proyectos (EPM).
- Oficina de Proyectos (PMO).
- Modelo de Madurez de la Gestión Organizacional de Proyectos (OPM3)®.

HERRAMIENTAS

- Curso Taller de MS Project 2003.
- Curso Taller de MS Project 2007.
- Curso Taller de MS Project 2010.
- MS Project Server 2007.
- Herramientas avanzadas para la Gestión de Proyectos.

Página Web: www.dharmacon.net , contacto: informes@dharma-consulting.com