U-tad
CENTRO UNIVERSITARIO
DE TECNOLOGÍA
Y ARTE DIGITAL

Verificación del Software

U-tad
CENTRO UNIVERSITARIO
DE TECNOLOGÍA Y ARTE DIGITAL

CENTRO UNIVERSITARIO
DE TECNOLOGÍA
Y ARTE DIGITAL

Verificación del Software

5. Gestión de pruebas

Alonso Álvarez García
alonso.alvarez@ext.live.u-tad.com

Verificación del Software

Tema 1. Fundamentos de Calidad del Sofware
Tema 2. QA en el SDLC
Tema 3. Revisión y pruebas
Tema 4. Técnicas y herramientas
Tema 5. Gestión de las pruebas

Gestión de las pruebas

5.1. Los roles del testing
5.2. Reporting y control. Trazabilidad

Verificación del Software

5.1 Los roles del testing

Roles en el testing

La calidad es cada vez más una **responsabilidad compartida**. Eso significa que las personas que participen en la construcción de un producto o servicio son **corresponsables** de su calidad, y aportan desde su rol particular.
Pero siguen siendo necesarios **perfiles específicos** relacionados con la calidad:

- Para la QA (*Quality Assurance*) es necesario personas que conozcan cómo organizar y gestionar los procesos.
- Para la QC (*Quality Control*) hacen falta personas que diseñen escenarios de prueba, que entiendan cómo organizarlas y seleccionar la modalidad más apropiada, que manejen las técnicas usadas en el testing, que cuenten con el conocimiento experto necesario para hacer pruebas exploratorias, sanity, smoke, …

Esta última parte se concreta en dos funciones específicas:

- Gestor/a, manager, líder de pruebas o **Test Manager**
- Experto/a en testing, ingeniero/a de pruebas, Testing Engineer o **Tester**

Cada organización, cada departamento, a veces cada proyecto, decide de qué forma implementar esas funciones, con roles o posiciones, o como responsabilidades dentro de otros.

Test manager

El/La Test Manager se responsabiliza:

- Del proceso de testing (definición, organización, seguimiento, …)
- De liderar las actividades de testing, lo que, en un entorno ágil, significa adoptar un rol de *servant-leadership*, fomentando el crecimiento de las personas y liderando con el ejemplo.

Los Test Managers son figuras específicas en grandes proyectos en los que hay múltiples equipos, y donde puede haber equipos especializados en la calidad del producto o servicio. También pueden ser líderes de disciplinas o *chapters*, estructuras especializadas de determinadas empresas.
En productos o servicios pequeños no suele haber líderes. Puede haber personas con un mayor grado de conocimiento experto, pero los equipos se suelen autoorganizar sin la necesidad de crear subequipos con sus propios gestores.

Puede ocurrir que el/la Test Manager se convierta en un **Testing Coach**, encargado de aconsejar y orientar a los equipos, sin necesidad de gestionarlos.

Actividades del Test manager (1)

- Definir, revisar y supervisar la estrategia y políticas de pruebas
- Planificación de actividades:
  - Comprensión del contexto y riesgos en los que se desarrollará la actividad
  - Elección de la mejor aproximación y técnicas
  - Estimaciones (tiempo, esfuerzos, …)
  - Definición de objetivos, niveles, …
  - Gestión de defectos
  - Y adaptar los planes en función de los resultados
- Escribir los test plan (selección del subconjunto de pruebas más adecuado en cada momento)
- Coordinación con otras actividades y managers
- Iniciar y supervisar el análisis, diseño, implementación, ejecución
- Monitorización del proceso y toma de decisiones, como cuándo terminar las pruebas
- Elaboración de informes

Actividades del Test manager (2)

- Definición y aplicación de métricas
- Supervisar la calidad del testing y sus productos
- Tomar decisiones sobre herramientas
- Definir una estrategia de entornos
- Y, sobre todo, preocuparse por la personas:
  - Participar en los procesos de selección
  - Hacer de coach y mentor
  - Fomentar la formación y crecimiento personal
  - Ayudar en la resolución de conflictos internos y con otras áreas/equipos

Actividades del Tester (1)

- Contribución activa en el desarrollo y revisión de planes de pruebas
- Revisión y análisis de inputs como requisitos, especificaciones, modelos, documentos, análisis, diseños, …
- Diseño y construcción de test cases, y procesos de prueba
- Preocuparse de la trazabilidad de test cases y objetos de prueba
- Diseño y configuración de entornos (generalmente con ayuda de personas de sistemas)
- Diseño de datasets y obtención de datos
- Ejecutar tests manuales, analizar resultados, identificar y notificar defectos
- Validar la resolución de defectos
- Registrar el grado de avance y desviaciones
- Usar las herramientas seleccionadas
- Automatizar tests (podría contar con apoyo)
- Revisar tests de otras personas
- Trabajar con NFRs (*Non Functional Requierements*) y lanzarlos: suele ser una labor especializada

Actividades del Tester (2)

La independencia del tester ayuda a evitar sesgos y condicionamientos. Hay varios grados de independencia:

- Probar tu propio trabajo (por ejemplo, developers que prueban su código): no hay independencia en las pruebas
- Una persona del mismo perfil prueba tu trabajo (por ejemplo, developer que prueba el código de otro developer)
- Una persona con distinto perfil en un mismo equipo (tester del mismo equipo prueba el código)
- Equipo independiente de pruebas dentro del área técnica
- Equipo independiente de pruebas que reporta a Negocio
- Equipo externo de pruebas

A medida que se gana en independencia también se pierde contexto, fluidez y alineamiento.

El testing independiente supone riesgos de aislamiento, falta de colaboración, de comunicación, de objetivos comunes, y puede llevar a formar silos con distintos objetivos y fricciones.

Aquí influye mucho el modelo de organización y de ciclo de vida adoptado.

Contribución al testing

El resto de personas (PO, desarrolladores, operaciones, Negocio, …) tiene una contribución destacable en la calidad y el testing:

- Developers:
  - Pruebas de componentes e integración
  - Automatización de tests unitarios
  - Soporte al testing y la automatización a niveles superiores
  - Revisión de tests
  - Y POR ENCIMA DE TODO, trabajar con las personas de testing como parte del mismo equipo, y no como terceros
- Negocio, usuarios, clientes, analistas de negocio o SMEs: pruebas UAT y soporte a la calidad y el testing
- Resto de roles: contribución aportando necesidades y requisitos y participando en en pruebas específicas (aceptación de operaciones, de seguridad, de UX, …)
- Dirección de la organización: soporte activo a la calidad y el testing

Verificación del Software

5.2 Organización de las pruebas

Planificación

Se materializa en el Test Plan, un mecanismo que estructura y orienta las pruebas y que obliga a definir cómo se van a hacer, con qué profundidad, con qué recursos, con qué herramientas, en qué tiempo, …
El Test Plan enumera qué se quiere probar y cómo hacerlo. También indica qué resultados se esperan y cómo se comunican (documentos, informes) y cómo se integra con el ciclo de vida del producto o servicio.

El hecho de tener un plan previo es una guía que permite anticipar problemas y recibir feedback, aunque como plan deba estar preparado para adaptaciones y cambios.

Un Test Plan debe estar alineado con la estrategia y políticas de la organización.
De un Test Plan de nivel superior (programa, solución, feature) se derivan otros más específicos (proyecto, equipo, release).

Criterios de entrada y salida

Son la forma de determinar el comienzo y final del proceso. Ayudan a mantener el control sobre la calidad del proceso y del producto. Puede ser de distinto tipo en función del nivel y naturaleza de las pruebas.

- **Criterios de entrada**: pasos necesarios para iniciar la pruebas.
  - Código completado y en qué condiciones: DoD cumplido
  - Disponibilidad de entornos, documentación, …
  - Pruebas diseñadas, disponibilidad de datos y herramientas de prueba, …
  - Etc, …
- **Criterios de salida**: condiciones necesarias para determinar que se han concluido las pruebas.
  - Porcentaje de plan de pruebas ejecutado
  - Cobertura de código validado
  - Número y severidad de bugs detectados pendientes
  - Pruebas no funcionales completadas
  - Etc, …

Monitorización y control

El proceso de testing debe tener un seguimiento para conocer cómo se está desarrollando y tomar decisiones al respecto. Esto se traduce en recopilar métricas y presentarla de forma que se puedan tomar decisiones sobre ella.
Esas métricas se fijarán en:

- El grado de ejecución del test plan
- Los resultados de esa ejecución
- El grado de cobertura del código y/o funcionalidades validados
- Los defectos encontrados, su severidad y estado

Esto nos permitirá conocer cómo se está desarrollando el proceso y anticipar si es necesario introducir cambios, hasta qué punto está siendo efectivo el testing, o la calidad del objeto probado.

Reporting

Es la forma de compartir información con la organización (especialmente el management) acerca del proceso.

Los informes deben emitirse regularmente (mejor si están siempre disponibles y actualizados) para conocer el avance, y al final del proceso para conocer el resultado.

Contenidos habituales:

- Resumen y estado del proceso de pruebas
- Eventos significativos
- Desviaciones
- Bloqueos e impedimentos
- Métricas y sus valores

Gestión de la configuración

Uno de los grandes problemas de la calidad del software es el desalineamiento de versiones y entornos. Esto supone:

- No poder reproducir defectos en distintos entornos
- No poder recuperar versiones específicas
- No tener la certeza de estar aplicando el test adecuado a una versión determinada del software
- Hacer cambios (fixing) sobre versiones que no muestran defectos
- Y en general pérdida de control sobre las funcionalidades definidas, el código que las implementa, el entorno sobre el que corren, y las pruebas que las verifican

El control de configuración es además la herramienta que garantiza la trazabilidad de las pruebas.

Con respecto a las pruebas en sí, el control de configuración ayuda a tener controlado cada test y su entorno: precondiciones, proceso, dataset y postcondiciones. Una misma prueba puede tener distintas versiones en función de cómo evolucione la definición e implementación de cada funcionalidad.

Gestión de riesgos

En el testing, los riesgos son posibles eventos cuyo impacto afecta al desarrollo de las pruebas. Los riesgos se categorizan en función de su impacto y probabilidad.
Hay dos tipos de riesgos:

- De proyecto (o actividad), relacionado con la forma en la que se desarrolla, por ejemplo retrasos en entregas o en la disponibilidad de entornos
- De producto, relacionado con el resultado en sí, como carencias, defectos, bajo rendimiento, etc

Los riesgos se anticipan y gestionan, es decir, se catalogan y buscan forma de reducir su posible impacto:

- Mitigación: plantear la condiciones para reducir el impacto y frecuencia del riesgo, o estar preparados para cuando aparezca
- Transferencia: si es posible hacer que la responsabilidad sobre el riesgo y su impacto pase a otro agente, área, organización, …
- Aceptación: cuando no es posible hacer nada de antemano, sólo reconocer su existencia

Gestión de defectos (1)

Como ya sabemos, los defectos son deficiencias del producto o desalineamientos con respecto a los requisitos. A veces son también el resultado de pruebas que no se corresponden con los valores previstos y que se señalan para una investigación más en profundidad.

Los defectos se procesan de distinta forma en función del modelo de ciclo de vida, el nivel de test en el que se han detectado, y la forma en la que se esté gestionando el testing. Por ejemplo, se puede haber acordado el convenio de que un defecto encontrado en el nivel de componente no se registra formalmente, o que los defectos se comunican inmediatamente, o sólo al terminar un ciclo de pruebas. Estos son aspectos que deben estar acordados y aceptados de antemano por todos los participantes en el proceso (management, Negocio, cliente, desarrollo, testing, …).

Gestión de defectos (2)

Todo fallo detectado **no es automáticamente** un defecto en el código:

- Puede haber condiciones del entorno que puedan haberlo provocado (caída de servidor, servicio o comunicaciones no relacionado, …).
- El test puede ser incorrecto, y no estar diseñado adecuadamente.
- El dataset del test puede ser incorrecto
- La propia ejecución del test –especialmente si es manual- puede dar lugar a falsos positivos.

Los defectos detectados se registran para:

- Resolverlos, lo que significa empezar por replicar el fallo.
- Obtener métricas e indicadores de la calidad del producto, por ejemplo comparando los defectos detectados con los resueltos.
- Mejora continua:
  - Análisis del proceso y su efectividad
  - Comparar distintos proyectos y productos, equipos, estrategias, herramientas, …

Además, las métricas pueden ayudar a anticipar la vida del producto o servicio una vez entregado.

Gestión de defectos (3)

Los defectos pueden aparecer en cualquier momento:

- Diseño
- Construcción
- Validación
- Producción

y usando cualquier técnica:

- Pruebas estáticas
- Pruebas dinámicas (caja negra, caja blanca)
- Uso por parte de los clientes y usuarios finales (que es el peor momento posible pero es inevitable que algunos defectos se filtren hasta aquí)

Gestión de defectos (4)

Los defectos no son únicamente problemas en el código. Otras fuentes de defectos son:

- Requisitos
- Historias de Usuario
- Criterios de aceptación
- Diseños
- Documentación
- Manuales
- Casos de prueba y datasets
- Configuraciones

Aunque normalmente los que se registran y procesan son los problemas en el código, o los de los productos de la construcción (manuales, documentación), todos los elementos mencionados son fuente de posibles defectos. En el caso de requisitos y diseños, los defectos tienen una relevancia especial porque solucionarlos reduce considerablemente el coste asociado a la (falta de) calidad.

Gestión de defectos (5)

La información que acompaña un defecto debe ser suficientemente detallada como para entender su naturaleza y poder reproducirlo, especialmente cuando se refiere al producto en sí. Datos que pueden formar parte de un informe de defecto pueden ser:

- Un identificador único que asegure la trazabilidad
- Momento en el que se detectó
- Quien lo hizo (si es manual)
- El objeto probado claramente identificado (versión, fecha)
- Entorno en el que se ha detectado, y las precondiciones que llevan a él
- Descripción de los pasos que llevan a la detección del fallo
- Dataset o datos introducidos que han provocado el fallo
- Evidencias del fallo: textos, capturas de pantalla, videos, ficheros de log, registros, …
- Severidad del defecto
- Prioridad del defecto
- No es estrictamente necesario pero a veces se puede aventurar una hipótesis sobre su causa

Además, habría información sobre la resolución: si se pudo replicar, qué lo causaba, cómo se ha solucionado, si se ha validado la solución, etc.

Gestión de defectos (6)

La severidad o criticidad de un defecto se fija en dos aspectos importantes:

- Frecuencia con la que ocurre
- Impacto

De ese modo, un defecto que **bloquee** una aplicación pero que suceda en condiciones muy **infrecuentes**, podría tener una criticidad menor que otro que **impida siempre** utilizar una función del programa.

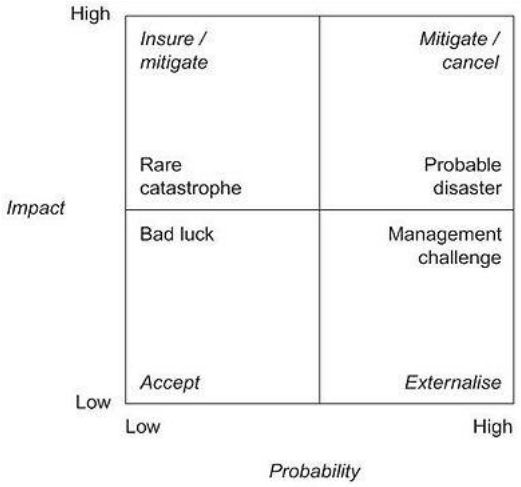
La escala de criticidad es un valor arbitrario que se fija dentro del equipo o de la organización. Su significado debe quedar claro para todos los implicados. Una posible escala podría ser:

1. Bloqueante: el defecto impide el uso del software
2. Crítico: hay funciones básicas que no están operativas
3. Mayor: fallo importante en algunas funciones
4. Menor: problemas de poca envergadura que no impiden usar el software
5. Triviales: defectos sin apenas impacto que pueden pasar desapercibidos

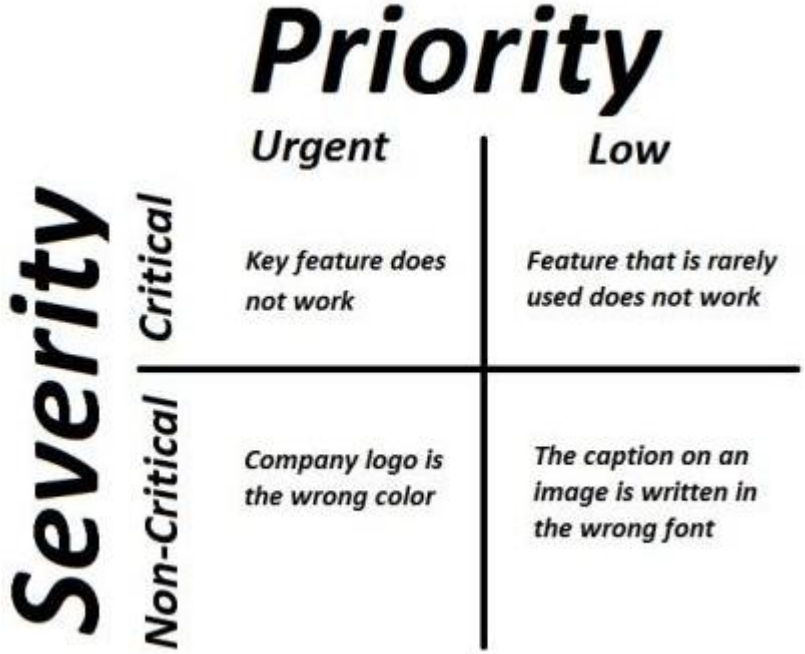
La prioridad, por su parte, también debe acordarse y puede hacer referencia, por ejemplo, a la urgencia para resolver el defecto.

Gestión de defectos (7)

No es lo mismo **severidad** que **prioridad**.



Severidad como combinación de impacto y probabilidad



Severidad frente a prioridad.

Ejemplo de gestión de defectos

En la web del curso encontrarás una página Web con un juego. Piensa un plan de pruebas (no hace falta que lo formalices, bastará un guión o una checklist). Si al ejecutarlo encuentras un defecto, ¿Cómo lo reflejarías? ¿Qué datos registrarías?

Ejemplo de gestión de defectos

SOLUCIÓN

En la web del curso encontrarás una página Web con un juego. Piensa un plan de pruebas (no hace falta que lo formalices, bastará un guión o una checklist). Si al ejecutarlo encuentras un defecto, ¿Cómo lo reflejarías? ¿Qué datos registrarías?

Identificador del fallo.
Pasos para reproducir el fallo (acciones, contexto, …)
Evidencias: descripción del fallo y, si es posible, adjuntar imágenes o vídeos.
Valores esperados y valores obtenidos.
Identificación (incluyendo versión) del sw probado y el entorno.
Severidad
Prioridad
Persona que lo ha identificado
Otros datos de interés