Pruebas de Requisitos, Diseño, Sistemas y *Hardware*

Pruebas de Sistemas

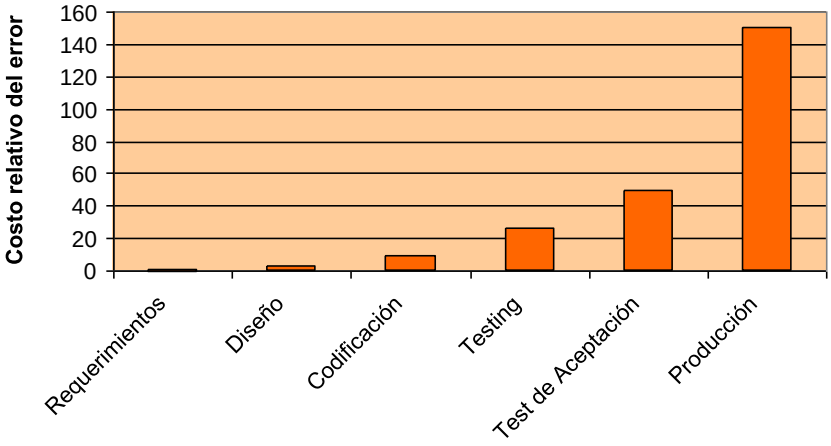
UNIVERSIDAD SIGLO 21

Prueba de requisitos

Históricamente los documentos de requisitos son los productos de trabajo menos probados/revisados en el ciclo de desarrollo. Pero con el tiempo y haciendo un análisis de los defectos que se detectan en los sistemas, esto ha ido cambiando. Se sabe que:

- Los defectos detectados en etapas finales del ciclo de vida de desarrollo, tienen un alto costo de reparación.
- Un defecto no detectado en fases tempranas tiene un efecto multiplicador, ya que se propaga y se va multiplicando a lo largo del ciclo de vida. Seguramente ese defecto no detectado, genera una cantidad exponencial de fallas en el producto final.
- Si se analiza la causa raíz de los defectos detectados en etapas finales, se tendría la sorpresa que un alto porcentaje de ellos tienen su origen en la etapa de requerimientos. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Figura 1: Costo relativo del error de acuerdo a la etapa



Fuente: elaboración propia.

La gráfica mostrada no debe ser tenida en cuenta de manera rigurosa en cuanto a los números, lo que tiene que advertirse es el costo relativo creciente de reparar el error en función de la etapa en la que se lo encuentra.

Según Hetzel (1988), revisar los documentos de requisitos involucra responder a dos preguntas básicas, con varios cuestionamientos relacionados:

1. ¿Está faltando algún requerimiento?
2. ¿Existe algún requerimiento que pueda ser simplificado o eliminado?

Las técnicas generales para alcanzar revisiones efectivas se vieron en el punto anterior, pero ahora se desarrollarán las técnicas particulares para *probar los requerimientos*.

Probar requerimientos a través de comprender lo que se debe probar

Obtener una declaración de los requerimientos para un sistema-*software* resulta simple como posible, sin embargo, obtener aquellos que cubran las necesidades "deseadas" ha demostrado ser excepcionalmente difícil. Es básicamente una tarea que Hetzel (1988) denomina *definición de problema*.

Como probadores tenemos que determinar si el enunciado del problema en esta fase ha sido formulado correctamente. Para ello, el procedimiento sería:

1. tomar la declaración del requerimiento;
2. ir al *back end*[1] de la solución;
3. sobre el back end de la solución, plantear cómo probaríamos para determinar que la solución propuesta actualmente soluciona la necesidad, es decir, cubre el requerimiento.

La experiencia demuestra que para alcanzar una buena declaración de requerimientos se deben especificar las pruebas para la aceptación de la solución junto con la declaración de los requerimientos. Cuando esto se hace a la par (el listado de requerimiento y el listado de casos de pruebas que probarán esos requerimientos), desaparecen muchos de los problemas asociados con malas interpretaciones o especificaciones poco claras.

[1] Se empleará back end sin traducción para indicar la parte final de la solución. Podría pensarse en un maquetado, por ejemplo, de lo que sería la interfaz del sistema.

Probar requerimientos a través del diseño de casos de pruebas basado en requerimientos

"Se llaman *casos de pruebas basados en requerimientos* a aquellos que se diseñan solamente a partir de las especificaciones externas o de los requerimientos del sistema y en el momento en que los requerimientos son definidos" (Uninotas, 2017, https://bit.ly/2KGaC4d).

Se ha reconocido que con esta práctica los requerimientos se vuelven más claros y las fallas en ellos se reconocen más fácilmente al entender cómo probarlos.

"Se pueden presentar las siguientes situaciones:

1. Que el requerimiento no esté completo: en ese caso, se emplea un caso de prueba que focalice la información que está faltando. Por ejemplo, si no está especificada una entrada o una funcionalidad" (Uninotas, 2017, https://bit.ly/2KGaC4d).
   ¿Debe diseñarse un caso de prueba con esa entrada y preguntar qué debería hacer el sistema en ese caso? Es mucho más efectiva esta pregunta que un comentario de que el requerimiento está incompleto.

2. "Que el requerimiento no sea preciso: de manera parecida se debe construir un caso de prueba y preguntar si ese es el resultado esperado para esa situación" (Uninotas, 2017, https://bit.ly/2KGaC4d). Así se focalizará la atención en la respuesta o resultado impreciso y se examinará más cuidadosamente.

Todo esto está basado en la convicción de que, tomando el tiempo y el esfuerzo en desarrollar un test completamente desde los requerimientos, se paga solo, se alcanzan muchas veces requerimientos mejorados y validados antes de que el diseño de sistemas y la programación hayan comenzado.

Matriz de validación de requerimientos

"Esta matriz lista cada requerimiento y hace referencia, para cada uno de ellos, los casos de pruebas o situaciones que se han creado para probarlos" (Uninotas, 2017, https://bit.ly/2KGaC4d).

Los casos de pruebas asociados a cada requerimiento deben probar exitosa y completamente cada requerimiento. Además, un caso de prueba asociado a un requerimiento puede estar asociado a otro también.

"Esta matriz se usa en varias instancias consecutivas: una, al crearla para la asociación entre requerimiento y caso de prueba, dos, al momento de la ejecución, por lo que registra el estado (corrido, paso/fallo), y así consecutivamente ejecución por ejecución" (Uninotas, 2017, https://bit.ly/2KGaC4d).

¿Qué columnas tendría esa matriz?

> Dentro de las columnas que ya puedes estar intuyendo a partir de lo descripto, debe ir una denominada status, con el uso que cada organización quiera darle. Puede ser: estado de especificación del requerimiento, estado de los casos de pruebas asociados al requerimiento, estado del requerimiento en función de los resultados de los casos de pruebas, número de defectos detectados para ese requerimiento, número identificador de defecto/los defectos levantados para ese requerimiento, entre otros usos. (Uninotas, 2017, https://bit.ly/2KGaC4d)

> El utilizar una Matriz de validación de requerimientos trae muchos beneficios:
>
> - Asegura que los requerimientos son listados, e identifica los casos de pruebas asociados a cada requerimiento.
> - Facilita las actividades de revisión, tanto de requerimientos como de pruebas.
> - Facilita el seguimiento de las actividades de diseño de casos, de ejecución de casos, y la medición del progreso de cada una de estas actividades.
> - Permite medir la cobertura de las pruebas (¿cuánto he abarcado con las pruebas?), medida en cobertura funcional.
> - El beneficio fundamental de utilizar esta matriz, es que la actividad de diseño de casos de pruebas no se pasa por alto y está forzada a que los casos de pruebas son diseñados para cada requerimiento. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Se facilita así la revisión formal, tanto de los requerimientos como de los casos de pruebas.

Prueba de requerimientos a través de prototipos o modelos

“Esta estrategia de pruebas consiste en construir un modelo o prototipo del sistema, no para ser usado, sino para probarlo y confirmar que se entiende el requerimiento”. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Esto bajo la premisa de que un gráfico, es mejor que miles de palabras, y un ejemplo es mejor que muchos gráficos.

Son rentables cuando se tiene un escaso entendimiento del requerimiento y se pretende alcanzar experiencia con el modelo de trabajo.

Lo que actualmente se conoce como desarrollo incremental es una extensión del concepto de prototipo.

> El desarrollo incremental consiste en el proceso de establecer los requerimientos y diseñarlos, construir y testear un sistema construido de a partes. En lugar de intentar resolver el sistema total todo de una vez, se selecciona una parte, se diseña, construye e implementa una parte. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Luego, en función de la experiencia con esa parte, se selecciona otra y así se continua. La ventaja es que los requerimientos para los incrementos de más adelante no tienen que ser especificados prematuramente y pueden ser ajustados con base en lo que va sucediendo.

Pruebas del diseño o modelos

Diseñar en el proceso de desarrollo es encontrar una solución que cumpla con los requerimientos.

Siempre hay varias alternativas de diseño que alcanzan la mayoría de los requerimientos. Nuestro objetivo como probadores es poder encontrar el diseño que se ajuste mejor a los requerimientos deseados. Cuando se dice "que mejor se ajuste", quiere decir que es el más simple o fácil de construir y el menos propenso a contener errores o fallas.

Al igual que en requerimientos, Hetzel (1988) dice que probar el diseño involucra responder a dos preguntas básicas, con varios cuestionamientos relacionados:

1. ¿Es esta solución elegida la correcta?
2. ¿La solución cumple los requerimientos?

"Nuevamente aquí la mejor técnica a aplicar sobre los diseños es la revisión formal, por lo que las mismas deben ser planeadas y desarrolladas durante la fase de diseño" (Uninotas, 2017, https://bit.ly/2KGaC4d).

Las revisiones deben determinar no solo cómo el diseño funciona, sino que la alternativa seleccionada es la mejor opción disponible.

Diseño de pruebas a través de análisis de alternativas

"El proceso de diseño involucra alternativas, por lo que la primera prueba a realizar es para confirmar que el enfoque que ha adoptado el diseñador, es la alternativa correcta" (Uninotas, 2017, https://bit.ly/2KGaC4d).

En la práctica, es poco frecuente el planteo de análisis alternativos, por lo que las revisiones de diseño, si se hacen, se centran en el cómo se espera que funcione el sistema, y así los diseñadores quedan encerrados en un enfoque en particular sin tener la seguridad de que sea el más adecuado.

Las alternativas de análisis deben ser examinadas de forma anticipada en el proceso de diseño, es decir en el momento en que son planteadas, permitiendo cualquier cambio para luego seguir con otro curso. Esto implica la necesidad de una revisión de diseño de alto nivel, concentrada en las alternativas y en la manera de evaluar y comparar esas alternativas.

El procedimiento de revisión de diseño a través del análisis de alternativas propuesto por Hetzel (1988) es dividir la revisión en dos partes:

1. Los revisores escuchan las alternativas de diseño propuestas por los diseñadores y repasan las ventajas y desventajas de cada una. Se suspende la revisión.
2. Cada revisor debe generar al menos una alternativa que no haya sido tenida en cuenta aún y así, en la segunda sesión son evaluadas y se reconsidera nuevamente la decisión del diseño completo. (Uninotas, 2017, https://bit.ly/2KGaC4d)

No hay ninguna duda de la rentabilidad que da la prueba del diseño tempranamente en el proceso de desarrollo. Es simple: o se lo hace bien desde el principio o se vive para siempre con las consecuencias.

Pruebas de diseño a través de modelos

Muchos aspectos críticos del diseño son adecuados para probar con un modelo de pruebas o simulación analítica. Consiste en construir una representación simple o modelo de alguna propiedad del diseño y, usando el modelo, probar así el diseño. El modelo permite que las pruebas tengan lugar tempranamente en la fase de diseño.

Estos modelos también se pueden extender para probar el diseño de configuraciones de la base de datos, secuencia de transacciones, tiempos de respuestas e interfaces de usuarios.

Pruebas de diseño a través de casos de pruebas basados en el diseño

De la misma manera que los requerimientos, fallas o errores de diseño pueden ser descubiertos, así también validaciones de diseños pueden garantizarse enfatizando en el diseño de casos de pruebas basados en diseño. Se llaman casos de pruebas basados en diseño.

Estos casos de pruebas hacen foco en el camino de los datos y los procesos dentro de la estructura del sistema. A través de la construcción de casos de pruebas especializados y analizando cómo el diseño los maneja se prueban la estructura interna, los procesos o caminos complejos, distintos escenarios, riesgos de diseño y áreas débiles, entre otros aspectos. (Uninotas, 2017, https://bit.ly/2KGaC4d)

El diseño de casos de pruebas basados en requerimientos y basados en diseño provee comprensión y es un recurso muy rico para las pruebas de diseño.

Pruebas de sistemas software

Acá no se refiere al nivel de pruebas denominado de la misma manera en el Módulo I,

> sino que se refiere a probar un sistema o programa realizando pruebas que lleven a validar el funcionamiento del programa. El desafío como probador es medir la calidad del código como así también asegurar que el programa ha sido completado correctamente. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Distinto a lo que propone Bill Hetzel (1988) en su libro *The complete guide of software testing*, dentro de las pruebas de sistemas quedan incluidas todas las estrategias de pruebas, ya sean para medir la lógica interna del programa como el funcionamiento sin tener en cuenta la lógica interna. Esto se llama pruebas de caja blanca y pruebas de caja negra, respectivamente. Las técnicas dentro de cada estrategia serán desarrolladas en la unidad siguiente.

> Estas pruebas pueden comenzar con el nivel de Pruebas de Componentes, ya que se puede comenzar probando pequeñas partes, unidades, funciones, rutinas, o subrutinas que operan individualmente. A esto se lo denomina "**in the small**"[2]. Se lo deja sin traducción para mejor comprensión. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Luego, se combinan estas partes con otras partes que también ya fueron probadas in the small y a estas pruebas se las llama *in the large*.

[2] *In the small*: en lo pequeño. *In the large*: en lo grande.

Alcanzar los objetivos de las pruebas denominadas in the small es responder a las siguientes preguntas principales:

1. La lógica del programa ¿funciona correctamente?
    a. El código ¿hace lo que tiene que hacer?
    b. El programa ¿puede fallar?

2. ¿Está toda la lógica presente?
    a. ¿Está faltando alguna función?
    b. El programa ¿hace todo lo especificado?

Estas preguntas tienen que conducir nuestra actividad de pruebas. El programador, por su parte, debe establecer que la lógica funciona adecuadamente y que no contiene errores o problemas latentes. Los probadores deben dar confianza sobre eso y determinar además que está presente toda la lógica necesaria o requerida. Entonces, es indispensable diseñar un set de casos de pruebas suficientes que ejercite esa lógica, que la ejecute, y así determinar si es correcto su funcionamiento.

Tipos de casos de pruebas

Es bueno tener una clasificación en categorías de los casos de pruebas. Una clasificación propuesta por Hetzel (1988) para las pruebas de sistemas es agruparlas en función del origen de los datos de pruebas:

Tabla 1: Clasificación de las pruebas de sistemas

Tipo de caso de prueba	Fuente	Ampliación
Basado en requerimiento	Especificaciones	Se derivan de las especificaciones o bien del entendimiento de lo que se supone que el sistema deberá hacer
Basados en Diseño	Lógica del sistema	Se derivan de la estructura lógica del sistema
Basados en el código	Estructura de datos y código	Se derivan de la lógica del programa y de cualquier elemento o archivo de datos usados en el programa
Aleatorios	Generador aleatorio	Se derivan a través de una técnica de muestreo y aleatoria, como los producidos por

		generadores de datos de pruebas dirigidos por parámetros
Extraídos	Archivos existentes o casos de prueba	Son aquellos que se extraen de archivos existentes, archivos vivos o de otros sistemas
Extremos	Condiciones límites	Aquellos que se derivan del esfuerzo de hacer fallar el sistema, condiciones límites y los peores casos de pruebas pensados

Fuente: Hetzel, 1988, p. 76-77.

Los pasos para seleccionar los casos de pruebas basados en requerimientos son:

1. Comenzar con los casos que naturalmente surgen a partir de las especificaciones y de la identificación de las funciones del sistema.
2. Crear un set mínimo que ejecute todas las entradas y salidas, que cubra la lista de funciones del sistema.
3. Descomponer cada condición compuesta en condiciones simples.
4. Examinar las combinaciones posibles de las condiciones simples y agregar los casos de pruebas para cada una de esas combinaciones posibles.
5. Depurar el listado eliminando todos los casos de pruebas que ya sean corridos por otros sets.
6. Revisar sistemática y cuidadosamente.

   Adicionalmente a los casos de pruebas basados en requerimientos, se deben generar casos de pruebas basados en diseño o en código. Éstos deben asegurar la prueba de cada segmento lógico de programa y que con ellos se cubra el diseño completo. (Uninotas, 2017, https://bit.ly/2KGaC4d)

Para lograr esto, los probadores están forzados a entrar en el código, determinar qué partes ya han sido probadas por los casos de pruebas basados en requerimientos, y construir tantos casos de pruebas como sean necesarios para las partes que no han sido probadas.

Tabla 2: Tabla comparativa de los tipos de casos de pruebas vistos

Pruebas basadas en requerimientos	Pruebas basadas en diseño o código
Independientes del diseño lógico	Dependientes del diseño lógico
Derivadas de las especificaciones	Derivadas de la estructura del programa
No pueden probar características extras	Pueden probar características extras

Fuente: elaboración propia.

Pruebas de hardware

Los fabricantes de productos de hardware luchan por encontrar nuevas maneras de acelerar la madurez de sus productos y por reducir su tiempo de salida al mercado. El uso de técnicas de simulación para evaluar la confiabilidad de un producto de hardware en condiciones de campo ha crecido en importancia.

Tradicionalmente, los procesos de desarrollo de hardware se basan en un proceso de diseñar, construir, probar y reparar (DBTF[3]). Con los tiempos de producción ajustados, este proceso puede demorar la producción si al momento de las pruebas se detectan deficiencias de diseño. Sumado a esto, está que es muy costoso y requiere tiempo fabricar y probar múltiples prototipos. “Con el ánimo de reducir el tiempo y el costo de desarrollar y mejorar la confiabilidad de los diseños, los ingenieros de hardware involucraron la calificación virtual y la verificación física en sus procesos de manufactura”. (Uninotas, 2017, https://bit.ly/2KGaC4d)

> Las *calificaciones virtuales* son la aplicación de técnicas de simulación para determinar la habilidad del sistema-hardware en alcanzar los objetivos deseados.
>
> La *verificación física* es el proceso de definir y conducir pruebas físicas sobre el producto, o sobre un producto representativo para demostrar el comportamiento con fallos inducidos. (Uninotas, 2017, https://bit.ly/2KGaC4d)

[3] DBTF: *design*, *build*, *test*, *fix*: proceso de diseñar, construir, probar, reparar.

“El proceso que se lleva adelante es:

1) Definir la información de entradas: la información de entradas para hacer las calificaciones virtuales se establece identificando fallas potenciales basadas en los materiales y las condiciones de carga” (Uninotas, 2017, https://bit.ly/2KGaC4d). Este procedimiento involucra revisar gráficos de diseño, lista de materiales y los documentos de especificaciones de fabricación para las partes y los materiales que serán usados en la creación del producto.

2) “Caracterización de carga: implica definir de manera anticipada los escenarios en los que el objeto de calificación virtual va a ser usado” (Uninotas, 2017, https://bit.ly/2KGaC4d). Debido al uso de distintos materiales en la fabricación de hardware, hay factores tales como la temperatura y la vibración que pueden producir lo que se conoce como estrés mecánico pudiendo generar así fallas de material.

3) “Análisis de los resultados y definición de rangos de valores aceptados: no todos los resultados pueden estar dentro de rangos no aceptados” (Uninotas, 2017, https://bit.ly/2KGaC4d). Lo que hay que determinar dentro de las calificaciones virtuales son los rangos de resultados que serán aceptados luego en la verificación física.

4) “Evaluación del riesgo de falla: conociendo los resultados del comportamiento con sus rangos de valores y según las condiciones de carga anticipadas, se deben establecer los mecanismos de fallas” (Uninotas, 2017, https://bit.ly/2KGaC4d).

Como se puede observar, este es un proceso general pero muy diferente al aplicado para las pruebas de sistemas de software. En cuanto a las fases de gestión, estas se pueden compartir y aplicar perfectamente. En cuanto a lo específico del hardware, también cabe aclarar que el proceso puede tomar características específicas según los componentes de hardware involucrados en la fabricación.

Referencias

Hetzel, B. (1988). *An introduction. The complete guide to software testing* (2.da ed.). Massachsetts, USA: QED Information Science.

Uninotas. (2017). *Clases de equivalencia PRUEBAS.* Recuperado de: https://www.uninotas.net/clases-de-equivalencia-pruebas