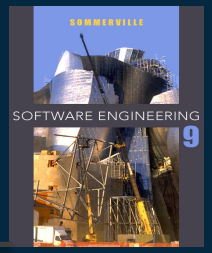
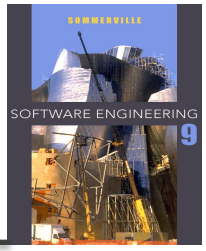
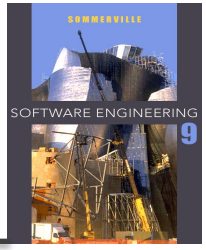




Capítulo 8 - Prueba de software

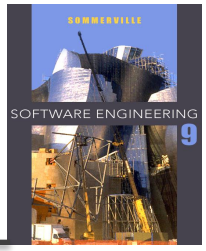
Conferencia 1

Temas cubiertos



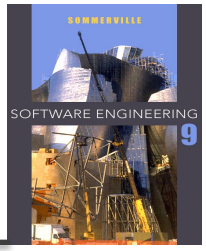
- Pruebas de desarrollo
- Desarrollo impulsado por pruebas
- Prueba de liberación
- Prueba de usuario

Pruebas del programa



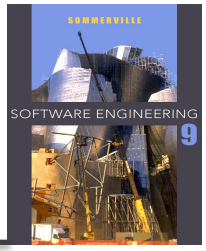
- Las pruebas tienen por objeto demostrar que un programa hace lo que se pretende y descubrir los defectos del programa antes de que se ponga en uso.
- Cuando se prueba el software, se ejecuta un programa usando datos artificiales.
- Se comprueba si los resultados de la prueba contienen errores, anomalías o información sobre los atributos no funcionales del programa.
- Puede revelar la presencia de errores NO su ausencia.
- Las pruebas forman parte de un proceso de verificación y validación más general, que también incluye técnicas de validación estáticas.

Los objetivos del programa de pruebas



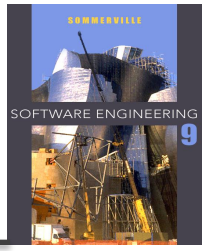
- Demostrar al desarrollador y al cliente que el software cumple con sus requisitos.
  - En el caso de los programas informáticos personalizados, esto significa que debe haber al menos una prueba para cada requisito del documento de requisitos. En el caso de los productos de software genérico, significa que debería haber pruebas para todas las características del sistema, además de las combinaciones de estas características, que se incorporarán en la versión del producto.
- Descubrir situaciones en las que el comportamiento del software es incorrecto, indeseable o no se ajusta a su especificación.
  - Las pruebas de defectos se ocupan de eliminar comportamientos indeseables del sistema, como caídas del sistema, interacciones indeseables con otros sistemas, cálculos

Validación y prueba de defectos



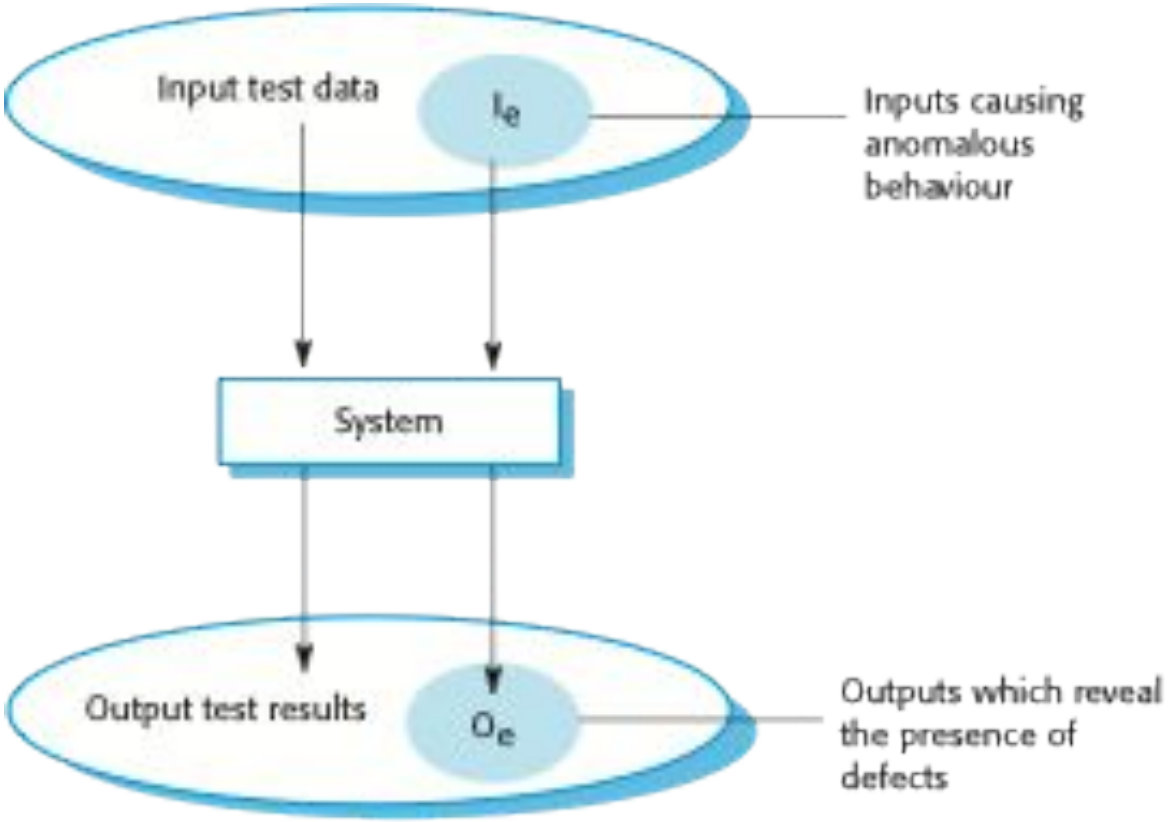
- El primer objetivo lleva a las pruebas de validación
  - Se espera que el sistema funcione correctamente utilizando un conjunto determinado de casos de prueba que reflejen el uso previsto del sistema.
- El segundo objetivo lleva a la prueba de defectos
  - Los casos de prueba están diseñados para exponer los defectos. Los casos de prueba en las pruebas de defectos pueden ser deliberadamente oscuros y no es necesario que reflejen la forma en que se utiliza normalmente el sistema.

Objetivos del proceso de prueba

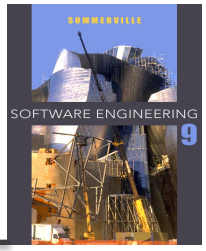


- Pruebas de validación
  - Demostrar al desarrollador y al cliente del sistema que el software cumple con sus requisitos
  - Una prueba exitosa muestra que el sistema funciona como se pretende.
- Prueba de defectos
  - Para descubrir fallos o defectos en el software cuando su comportamiento es incorrecto o no se ajusta a su especificación
  - Una prueba exitosa es una prueba que hace que el sistema funcione incorrectamente y así expone un defecto en el sistema.

Un modelo de entrada y salida de pruebas de programas

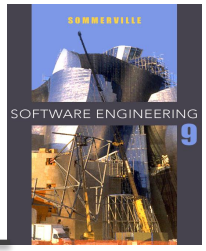


Verificación vs. validación



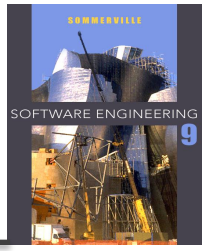
- Verificación:
  "¿Estamos construyendo bien el producto?"
- El software debe ajustarse a sus especificaciones.
- Validación:
  "¿Estamos construyendo el producto correcto?"
- El software debe hacer lo que el usuario realmente requiere.

V & V confianza



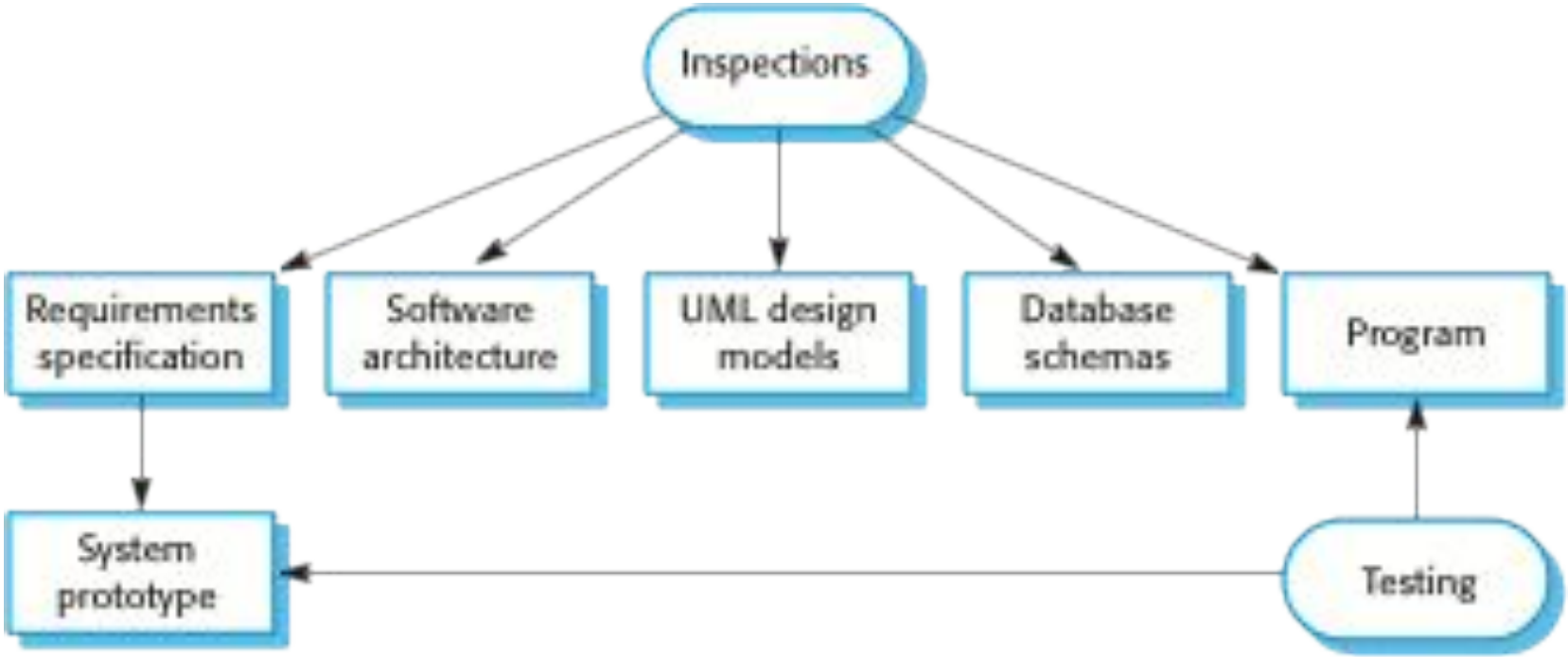
- El objetivo de V & V es establecer la confianza de que el sistema es "apto para el propósito".
- Depende del propósito del sistema, las expectativas de los usuarios y el entorno de comercialización
  - El propósito del software
    - El nivel de confianza depende de lo crítico que sea el software para una organización.
  - Las expectativas del usuario
    - Los usuarios pueden tener bajas expectativas de ciertos tipos de software.
  - Entorno de marketing
    - Llevar un producto al mercado antes de tiempo puede ser más importante que encontrar defectos en el programa.

Inspecciones y pruebas

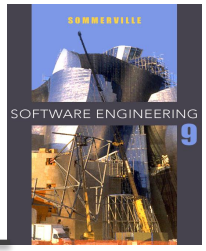


- Inspecciones de programas informáticos
  Preocupados por el análisis de
  la representación del sistema estático para descubrir problemas (verificación estática)
  - Puede complementarse con un análisis de documentos y códigos basado en herramientas.
  - Discutido en el capítulo 15.
- Pruebas de software Preocupados por el ejercicio y observar el comportamiento del producto (verificación dinámica)
  - El sistema se ejecuta con datos de prueba y se observa su comportamiento operacional.

Inspecciones y pruebas

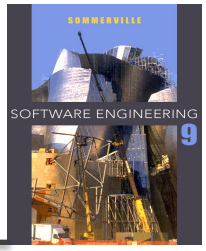


Inspecciones de software



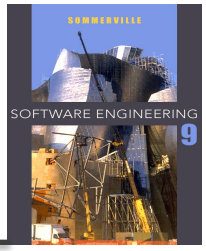
- Se trata de personas que examinan la representación de la fuente con el objetivo de descubrir anomalías y defectos.
- Las inspecciones no requieren la ejecución de un sistema, por lo que pueden utilizarse antes de su aplicación.
- Pueden aplicarse a cualquier representación del sistema (requisitos, diseño, datos de configuración, datos de prueba, etc.).
- Se ha demostrado que son una técnica efectiva para descubrir errores de programa.

Ventajas de las inspecciones



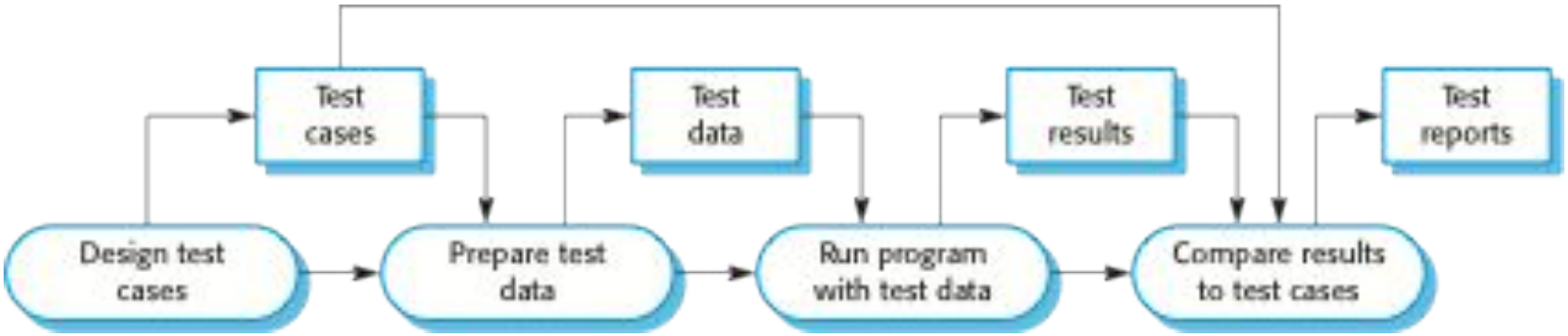
- Durante las pruebas, los errores pueden enmascarar (ocultar) otros errores. Dado que la inspección es un proceso estático, no hay que preocuparse por las interacciones entre los errores.
- Las versiones incompletas de un sistema pueden ser inspeccionadas sin costos adicionales. Si un programa está incompleto, entonces necesita desarrollar arneses de prueba especializados para probar las partes que están disponibles.
- Además de buscar defectos en el programa, una inspección también puede considerar atributos de calidad más amplios de un programa, como el cumplimiento de las normas, la portabilidad y la capacidad de mantenimiento.

Inspecciones y pruebas

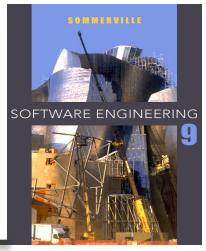


- Las inspecciones y pruebas son técnicas de verificación complementarias y no opuestas.
- Ambos deben ser usados durante el proceso de V & V.
- Las inspecciones pueden comprobar la conformidad con una especificación pero no con los requisitos reales del cliente.
- Las inspecciones no pueden comprobar características no funcionales como el rendimiento, la utilidad, etc.

Un modelo del proceso de prueba de software

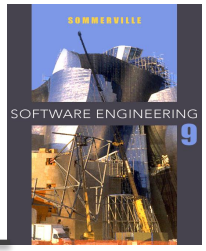


Etapas de las pruebas



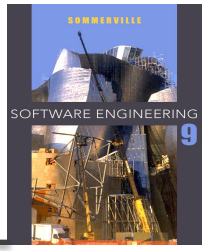
- Pruebas de desarrollo, en las que el sistema se prueba durante el desarrollo para descubrir fallos y defectos.
- Prueba de lanzamiento, donde un equipo de pruebas separado prueba una versión completa del sistema antes de que sea liberado a los usuarios.
- Prueba de usuario, en la que los usuarios o posibles usuarios de un sistema prueban el sistema en su propio entorno.

Pruebas de desarrollo



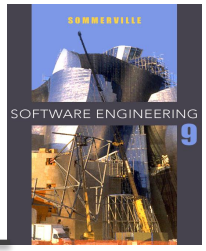
- Las pruebas de desarrollo incluyen todas las actividades de prueba que realiza el equipo que desarrolla el sistema.
  - Pruebas de unidades, donde se prueban las unidades de programas individuales o las clases de objetos. Las pruebas unitarias deben centrarse en probar la funcionalidad de los objetos o métodos.
  - Pruebas de componentes, donde varias unidades individuales se integran para crear componentes compuestos. Las pruebas de componentes deben centrarse en probar las interfaces de los componentes.
  - Prueba de sistema, donde algunos o todos los componentes de un sistema se integran y el sistema se prueba como un todo. Las pruebas del sistema deben centrarse en probar las interacciones de los componentes.

Prueba de la unidad



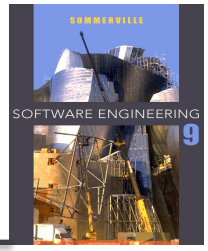
- La prueba de la unidad es el proceso de probar los componentes individuales de forma aislada.
- Es un proceso de prueba de defectos.
- Las unidades pueden serlo:
  - Funciones o métodos individuales dentro de un objeto
  - Clases de objetos con varios atributos y métodos
  - Componentes compuestos con interfaces definidos utilizados para acceder a su funcionalidad.

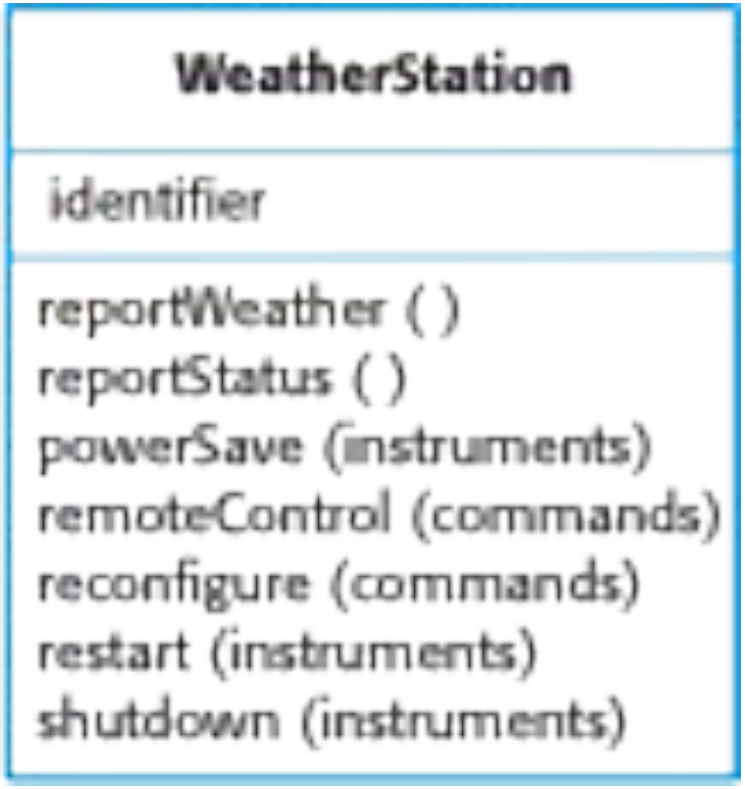
Prueba de clase de objeto



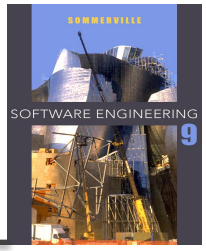
- La cobertura completa de la prueba de una clase implica
  - Comprobación de todas las operaciones asociadas a un objeto
  - Estableciendo e interrogando todos los atributos del objeto
  - Ejercitando el objeto en todos los estados posibles.
- La herencia dificulta el diseño de pruebas de clase de objetos, ya que la información a probar no está localizada.

La interfaz de objetos de la estación meteorológica



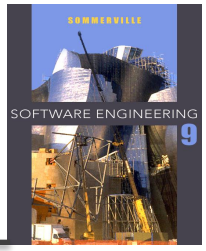


Pruebas de la estación meteorológica



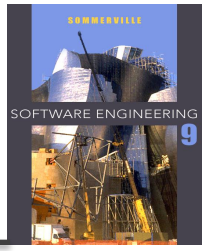
- Necesidad de definir casos de prueba para el informe de clima, calibración, prueba, arranque y apagado.
- Usando un modelo de estado, identificar las secuencias de transiciones de estado que se van a probar y las secuencias de eventos que causarán estas transiciones
- Por ejemplo:
  - Apagado -> Correr-> Apagado
  - Configurando-> Corriendo-> Probando-> Transmitiendo-> Corriendo
  - Correr-> Recoger-> Correr-> Resumir-> Transmitir-> Correr

Pruebas automatizadas



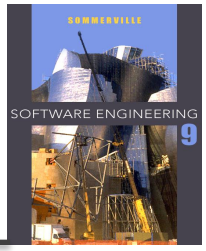
- Siempre que sea posible, las pruebas de la unidad deben automatizarse para que las pruebas se ejecuten y se comprueben sin intervención manual.
- En las pruebas de unidades automatizadas, se utiliza un marco de automatización de pruebas (como JUnit) para escribir y ejecutar las pruebas del programa.
- Los marcos de pruebas unitarias proporcionan clases de prueba genéricas que se extienden para crear casos de prueba específicos. Pueden entonces ejecutar todas las pruebas que usted ha implementado e informar, a menudo a través de alguna interfaz gráfica de usuario, sobre el éxito de las demás pruebas.

Componentes de prueba automatizados



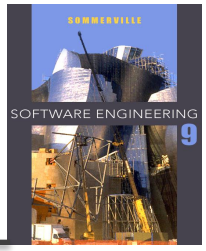
- Una parte de la configuración, donde se inicializa el sistema con el caso de prueba, es decir, las entradas y salidas esperadas.
- Una parte de la llamada, donde se llama al objeto o método a probar.
- Una parte de afirmación en la que se compara el resultado de la llamada con el resultado esperado. Si la afirmación se evalúa como verdadera, la prueba ha sido exitosa si es falsa, entonces ha fallado.

Eficacia de la prueba unitaria



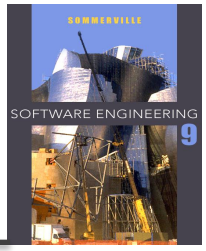
- Los casos de prueba deben mostrar que, cuando se usa como se espera, el componente que se está probando hace lo que se supone que debe hacer.
- Si hay defectos en el componente, éstos deben ser revelados por casos de prueba.
- Esto nos lleva a dos tipos de caso de prueba de la unidad:
  - El primero de ellos debe reflejar el funcionamiento normal de un programa y debe mostrar que el componente funciona como se espera.
  - El otro tipo de caso de prueba debe basarse en la experiencia de prueba de dónde surgen los problemas comunes. Debería utilizar entradas anormales para comprobar que éstas se procesan correctamente y no chocan con el componente.

Estrategias de prueba



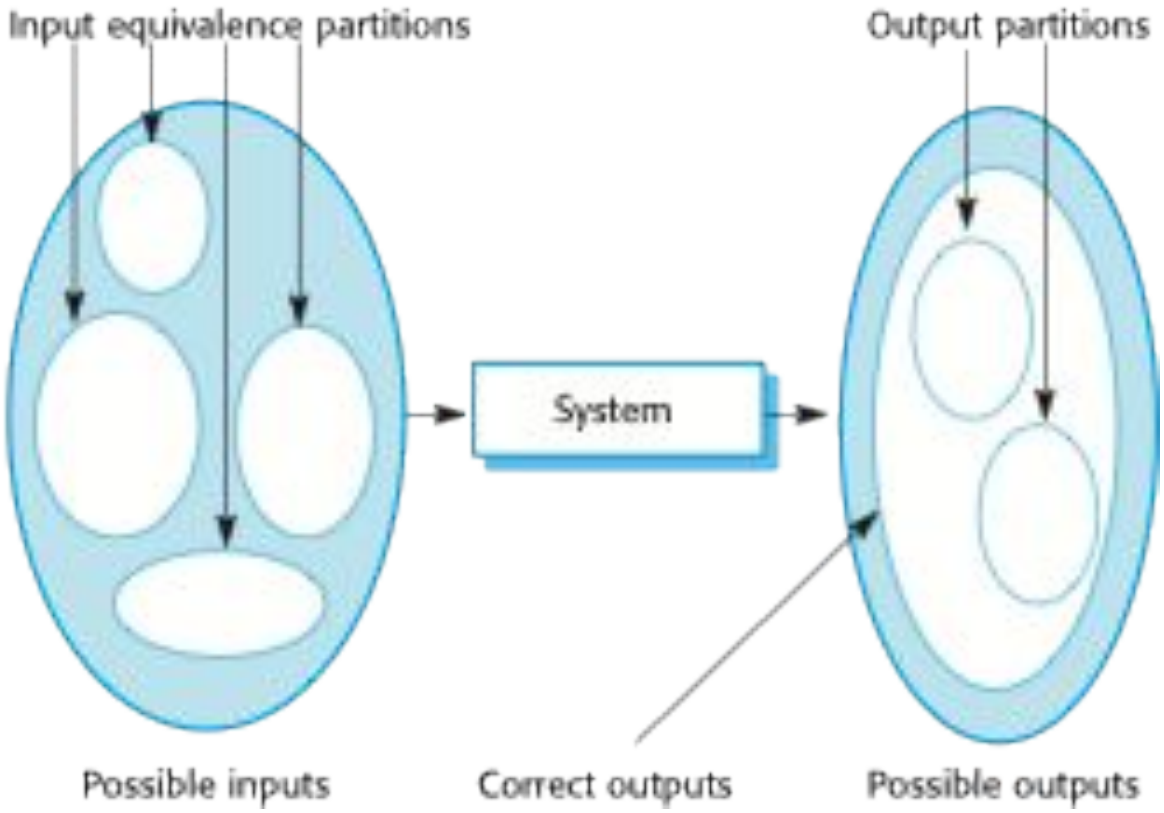
- Prueba de partición, donde se identifican grupos de insumos que tienen características comunes y deben ser procesados de la misma manera.
  - Debe elegir pruebas dentro de cada uno de estos grupos.
- Pruebas basadas en directrices, donde se usan las directrices de las pruebas para elegir los casos de prueba.
  - Estas directrices reflejan la experiencia previa de los tipos de errores que los programadores suelen cometer al desarrollar componentes.

Pruebas de partición

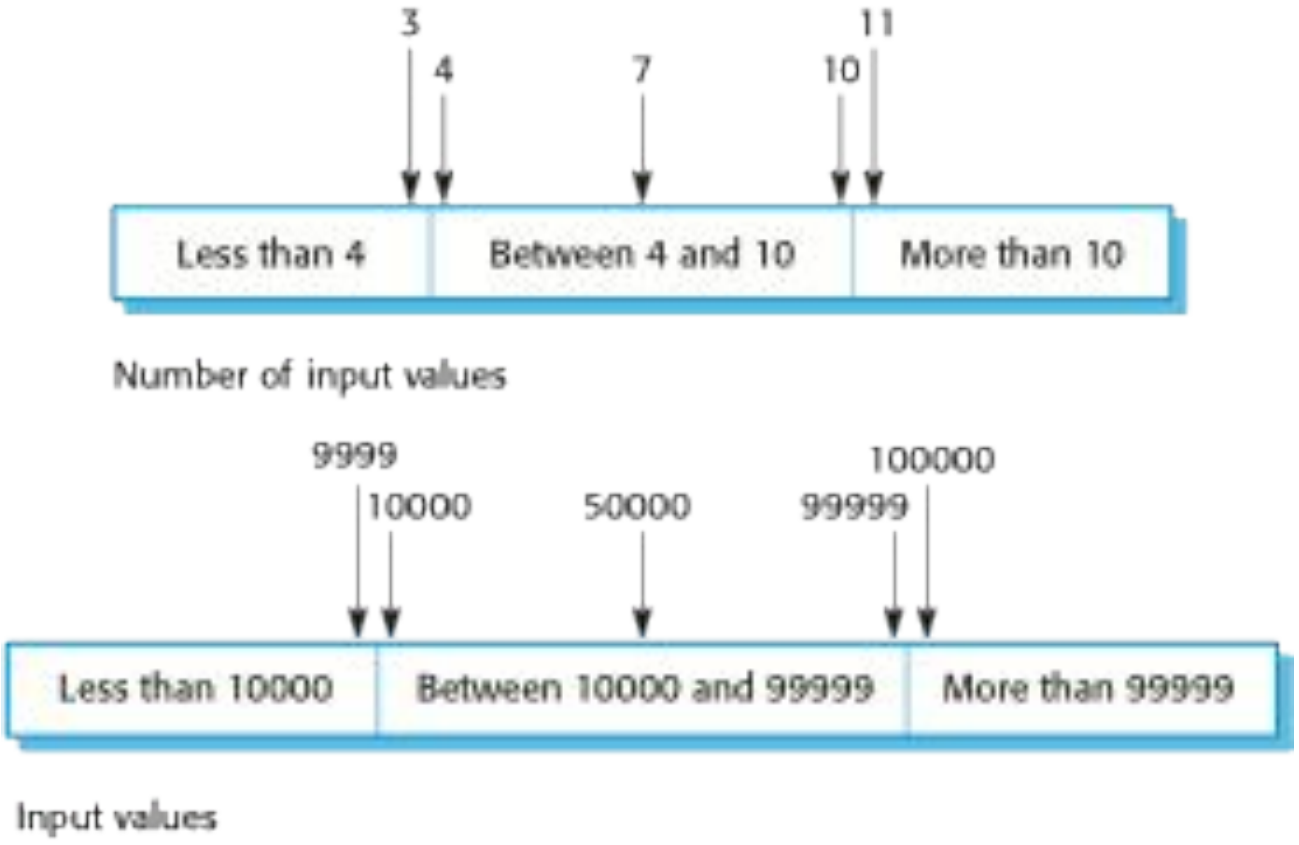


- Los datos de entrada y los resultados de salida a menudo se dividen en diferentes clases en las que todos los miembros de una clase están relacionados.
- Cada una de estas clases es una partición de equivalencia o dominio donde el programa se comporta de manera equivalente para cada miembro de la clase.
- Los casos de prueba deben elegirse de cada partición.

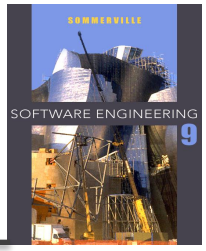
Equivalencia de la partición



Particiones de equivalencia

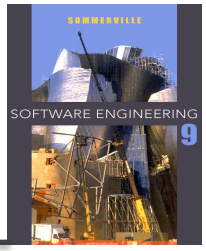


Pautas de prueba (secuencias)



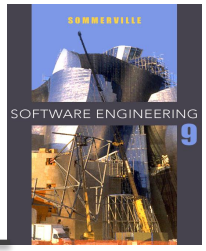
- Software de prueba con secuencias que tienen un solo valor.
- Usar secuencias de diferentes tamaños en diferentes pruebas.
- Derivar las pruebas para acceder al primer, medio y último elemento de la secuencia.
- Prueba con secuencias de longitud cero.

Pautas generales de pruebas

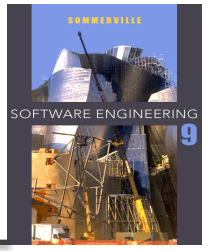


- Elegir entradas que obliguen al sistema a generar todos los mensajes de error
- Diseñar entradas que causen que los buffers de entrada se desborden
- Repita la misma entrada o serie de entradas varias veces
- Forzar la generación de salidas no válidas
- Forzar los resultados del cálculo a ser demasiado grandes o demasiado pequeños.

Puntos clave



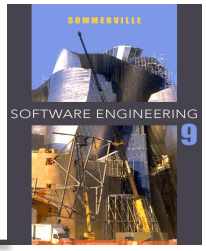
- Las pruebas sólo pueden mostrar la presencia de errores en un programa. No puede demostrar que no hay fallos restantes.
- Las pruebas de desarrollo son responsabilidad del equipo de desarrollo de software. Un equipo separado debe ser responsable de probar un sistema antes de que sea liberado a los clientes.
- Las pruebas de desarrollo incluyen pruebas de unidad, en las que se prueban objetos individuales y métodos pruebas de componentes en las que se prueban grupos de objetos relacionados y pruebas de sistemas, en las que se prueban sistemas parciales o completos.



Capítulo 8 - Prueba de software

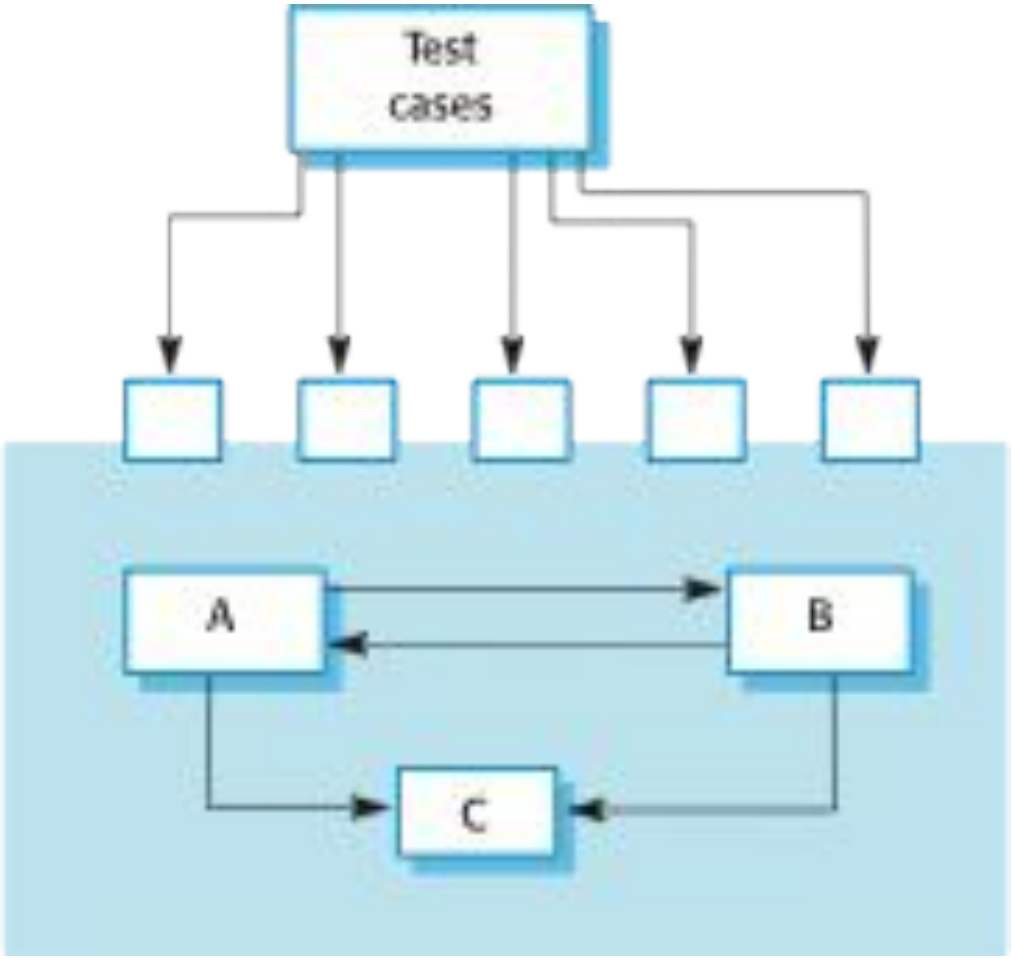
Conferencia 2

Pruebas de componentes

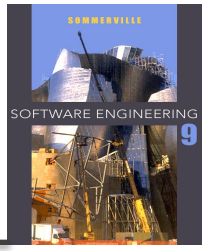


- Los componentes de software son a menudo componentes compuestos que están hechos de varios objetos que interactúan.
  - Por ejemplo, en el sistema de la estación meteorológica, el componente de reconfiguración incluye objetos que se ocupan de cada aspecto de la reconfiguración.
- Se accede a la funcionalidad de estos objetos a través de la interfaz de componentes definida.
- Por lo tanto, las pruebas de los componentes compuestos deben centrarse en demostrar que la interfaz del componente se comporta de acuerdo con su especificación.
  - Puede asumir que las pruebas de unidad en los objetos individuales dentro del componente han sido completadas.

Prueba de la interfaz

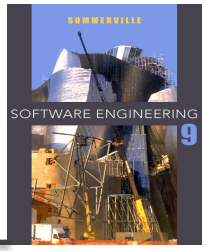


Prueba de la interfaz



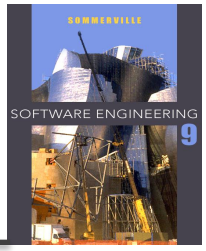
- Los objetivos son detectar fallos debidos a errores de interfaz o a suposiciones no válidas sobre las interfaces.
- Tipos de interfaz
  - Interfaces de parámetros Los datos pasan de un método o procedimiento a otro.
  - Interfaces de memoria compartida El bloque de memoria se comparte entre procedimientos o funciones.
  - Interfaces de procedimiento El subsistema encapsula un conjunto de procedimientos que serán llamados por otros subsistemas.
  - Interfaces de paso de mensajes Los subsistemas solicitan servicios de otros subsistemas

Errores de la interfaz



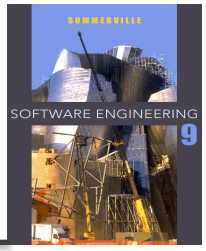
- El mal uso de la interfaz
  - Un componente que llama llama a otro componente y comete un error en el uso de su interfaz, por ejemplo, parámetros en un orden equivocado.
- Malentendido de la interfaz
  - Un componente de llamada incorpora suposiciones sobre el comportamiento del componente llamado que son incorrectas.
- Errores de tiempo
  - El componente de llamada y el de llamada funcionan a diferentes velocidades y se accede a información desactualizada.

Directrices para la prueba de la interfaz



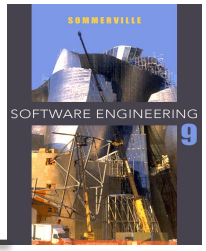
- Diseñar pruebas para que los parámetros de un procedimiento llamado estén en los extremos de sus rangos.
- Siempre pruebe los parámetros de los punteros con punteros nulos.
- Pruebas de diseño que hacen que el componente falle.
- Usar pruebas de estrés en los sistemas de paso de mensajes.
- En los sistemas de memoria compartida, varía el orden en que se activan los componentes.

Pruebas del sistema



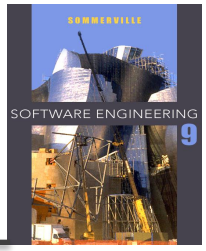
- Las pruebas del sistema durante el desarrollo implican la integración de componentes para crear una versión del sistema y luego probar el sistema integrado.
- El enfoque de las pruebas de sistemas es probar las interacciones entre los componentes.
- Las pruebas del sistema comprueban que los componentes son compatibles, interactúan correctamente y transfieren los datos correctos en el momento adecuado a través de sus interfaces.
- Las pruebas de sistemas comprueban el comportamiento emergente de un sistema.

Pruebas de sistemas y componentes



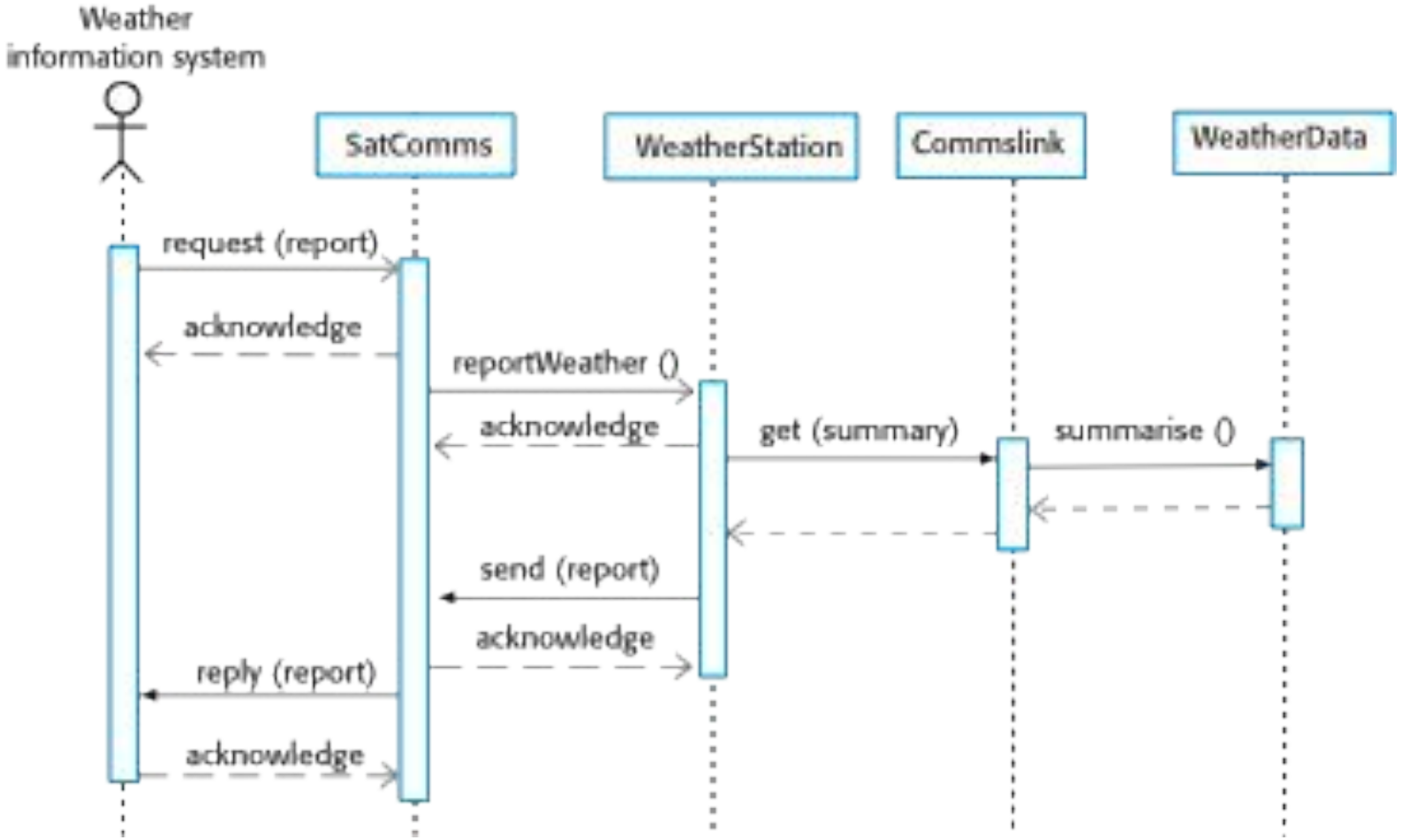
- Durante las pruebas de los sistemas, los componentes reutilizables que se han desarrollado por separado y los sistemas de venta al público pueden integrarse con los componentes de nuevo desarrollo. A continuación se prueba el sistema completo.
- Los componentes desarrollados por los diferentes miembros o subequipos del equipo pueden ser integrados en esta etapa. La prueba del sistema es un proceso colectivo más que individual.
  - En algunas empresas, las pruebas del sistema pueden implicar un equipo de pruebas separado sin la participación de diseñadores y programadores.

Pruebas de casos de uso

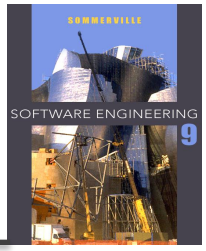


- Los casos de uso desarrollados para identificar las interacciones del sistema pueden utilizarse como base para las pruebas del sistema.
- Cada caso de uso suele implicar varios componentes del sistema, por lo que la prueba del caso de uso obliga a que se produzcan estas interacciones.
- Los diagramas de secuencia asociados al caso de uso documentan los componentes e interacciones que se están probando.

Recopilar la tabla de secuencia de datos meteorológicos

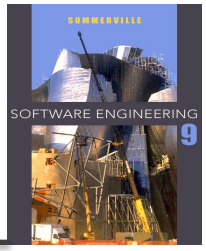


Políticas de pruebas



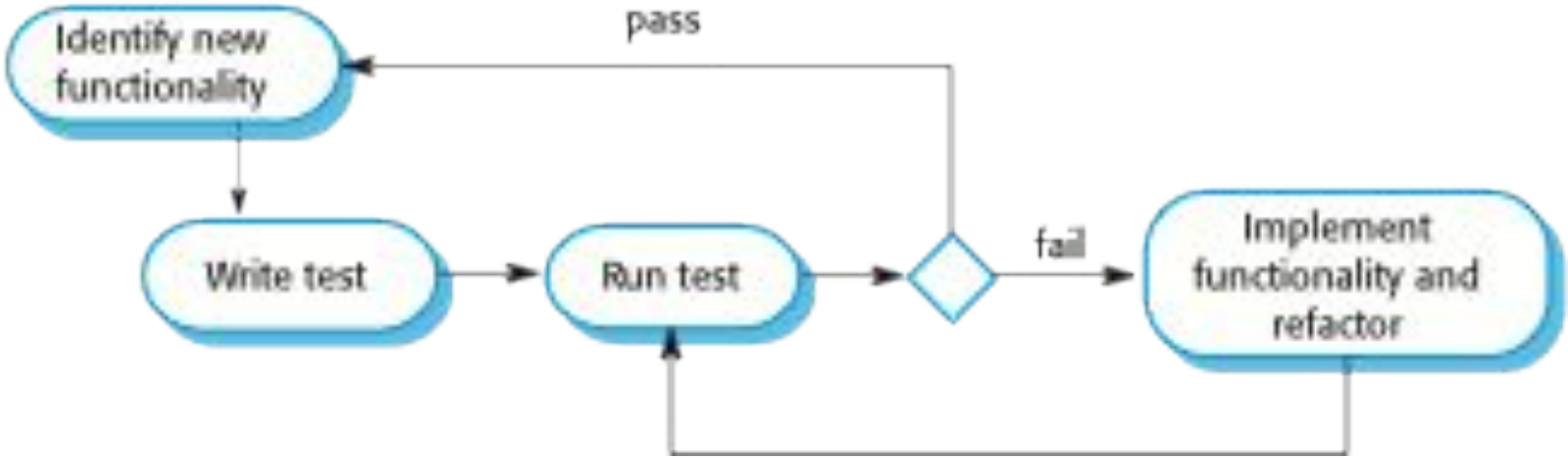
- Es imposible realizar pruebas exhaustivas del sistema, por lo que se pueden desarrollar políticas de pruebas que definan la cobertura de pruebas del sistema requerida.
- Ejemplos de políticas de pruebas:
  - Todas las funciones del sistema a las que se accede a través de los menús deben ser probadas.
  - Deben probarse las combinaciones de funciones (por ejemplo, el formato de texto) a las que se accede a través del mismo menú.
  - En los casos en que se proporcione la entrada del usuario, todas las funciones deben ser probadas tanto con la entrada correcta como con la incorrecta.

Desarrollo impulsado por pruebas

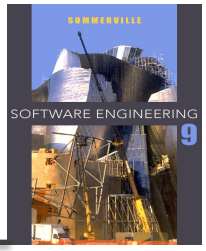


- El desarrollo dirigido por pruebas (TDD) es un enfoque del desarrollo de programas en el que se intercalan las pruebas y el desarrollo de código.
- Las pruebas se escriben antes que el código y "pasar" las pruebas es el motor crítico del desarrollo.
- Desarrolla el código de forma incremental, junto con una prueba para ese incremento. No pasas al siguiente incremento hasta que el código que has desarrollado pasa su prueba.
- TDD se introdujo como parte de métodos ágiles como la Programación Extrema. Sin embargo, también puede utilizarse en procesos de desarrollo basados en planes.

Desarrollo impulsado por pruebas

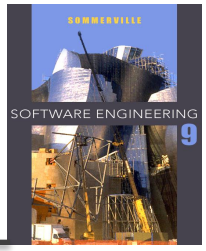


Actividades del proceso de TDD



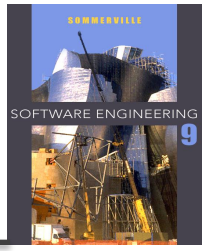
- Empiece por identificar el incremento de la funcionalidad que se requiere. Normalmente, esto debería ser pequeño y ejecutable en unas pocas líneas de código.
- Escriba una prueba para esta funcionalidad e impleméntela como una prueba automatizada.
- Ejecute la prueba, junto con todas las demás pruebas que se han implementado. Inicialmente, no ha implementado la funcionalidad, así que la nueva prueba fallará.
- Implementar la funcionalidad y volver a ejecutar la prueba.
- Una vez que todas las pruebas se realizan con éxito, se pasa a la implementación de la siguiente parte de la

Beneficios del desarrollo basado en pruebas



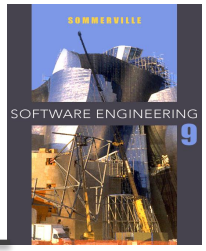
- Cobertura del código
  - Cada segmento de código que escribes tiene al menos una prueba asociada, así que todo código escrito tiene al menos una prueba.
- Prueba de regresión
  - Un conjunto de pruebas de regresión se desarrolla de forma incremental a medida que se desarrolla un programa.
- Depuración simplificada
  - Cuando una prueba falla, debería ser obvio dónde está el problema. El código recién escrito necesita ser revisado y modificado.
- Documentación del sistema
  - Las pruebas en sí son una forma de documentación que describe lo que el código debería hacer.

Prueba de regresión



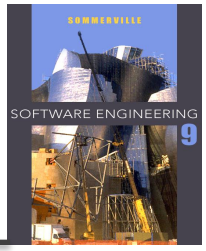
- Las pruebas de regresión están probando el sistema para comprobar que los cambios no han "roto" el código que funcionaba anteriormente.
- En un proceso de prueba manual, la prueba de regresión es costosa pero, con la prueba automatizada, es simple y directa. Todas las pruebas se repiten cada vez que se hace un cambio en el programa.
- Las pruebas deben realizarse "con éxito" antes de que se cometa el cambio.

Prueba de liberación



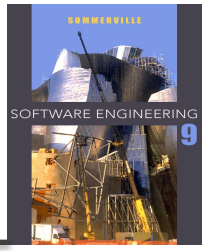
- La prueba de liberación es el proceso de probar una liberación particular de un sistema que está destinado a ser utilizado fuera del equipo de desarrollo.
- El objetivo principal del proceso de pruebas de liberación es convencer al proveedor del sistema de que es lo suficientemente bueno para su uso.
  - Las pruebas de lanzamiento, por lo tanto, tienen que demostrar que el sistema ofrece su funcionalidad, rendimiento y fiabilidad especificados, y que no falla durante el uso normal.
- Las pruebas de liberación son normalmente un proceso de pruebas de caja negra en el que las pruebas sólo se derivan de la especificación del sistema.

Pruebas de lanzamiento y pruebas del sistema



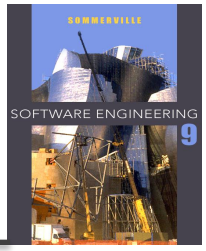
- La prueba de liberación es una forma de prueba del sistema.
- Diferencias importantes:
  - Un equipo separado, que no ha participado en el desarrollo del sistema, debería encargarse de las pruebas de lanzamiento.
  - Las pruebas del sistema por parte del equipo de desarrollo deberían centrarse en descubrir los fallos del sistema (pruebas de defectos). El objetivo de las pruebas de lanzamiento es comprobar que el sistema cumple sus requisitos y es lo suficientemente bueno para el uso externo (pruebas de validación).

Pruebas basadas en los requisitos



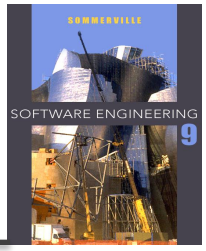
- Las pruebas basadas en los requisitos implican el examen de cada requisito y la elaboración de una o varias pruebas para el mismo.
- Requisitos del MHC-PMS:
  - Si se sabe que un paciente es alérgico a algún medicamento en particular, la prescripción de ese medicamento dará lugar a que se emita un mensaje de advertencia al usuario del sistema.
  - Si un prescriptor elige ignorar una advertencia de alergia, debe dar una razón por la que se ha ignorado.

Pruebas de requisitos



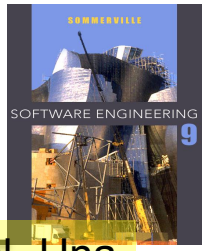
- Prepara un registro de pacientes sin alergias conocidas. Prescribir medicamentos para las alergias que se sabe que existen. Comprobar que el sistema no emita un mensaje de advertencia.
- Prepara un registro de pacientes con una alergia conocida. Prescribir la medicación a la que el paciente es alérgico, y comprobar que la advertencia es emitida por el sistema.
- Establezca un registro de pacientes en el que se registren las alergias a dos o más medicamentos. Prescriba ambos medicamentos por separado y compruebe que se emite la advertencia correcta para cada uno de ellos.
- Prescribe dos medicamentos a los que el paciente es alérgico. Compruebe que las dos advertencias se emiten correctamente.
- Prescribir una droga que emita una advertencia y anular esa advertencia. Comprobar que el sistema requiere que el usuario proporcione información que explique por qué se anuló la advertencia.

Características probadas por el escenario



- Autenticación mediante el ingreso al sistema.
- Descarga y carga de registros de pacientes específicos en un ordenador portátil.
- Programación de visitas a domicilio.
- Cifrado y descifrado de las historias clínicas de los pacientes en un dispositivo móvil.
- Recuperación y modificación de registros.
- Enlaces con la base de datos de drogas que mantiene información sobre los efectos secundarios.
- El sistema de aviso de llamada.

Un escenario de uso para el MHC-PMS

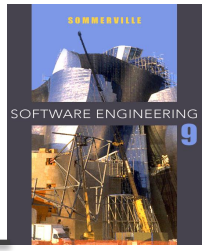


Kate es una enfermera que se especializa en el cuidado de la salud mental. Una de sus responsabilidades es visitar a los pacientes en sus casas para comprobar que su tratamiento es efectivo y que no sufren los efectos secundarios de la medicación.
En un día de visitas a domicilio, Kate entra en el MHC-PMS y lo utiliza para imprimir su programa de visitas a domicilio para ese día, junto con información resumida sobre los pacientes a visitar. Ella solicita que los registros de estos pacientes sean descargados a su computadora portátil. Se le pide su frase clave para encriptar los registros en la computadora portátil.

Uno de los pacientes a los que visita es Jim, que está siendo tratado con medicación para la depresión. Jim siente que la medicación le está ayudando pero cree que tiene el efecto secundario de mantenerlo despierto por la noche. Sara busca el historial de Juan y se le pide su frase clave para descifrarlo. Revisa el medicamento recetado y pregunta por sus efectos secundarios. El insomnio es un efecto secundario conocido, así que anota el problema en el registro de Juan y sugiere que visite la clínica para que le cambien la medicación. Él está de acuerdo, así que Sara ingresa un aviso para llamarlo cuando vuelva a la clínica para hacer una cita con un médico. Termina la consulta y el sistema vuelve a codificar el expediente de Juan.

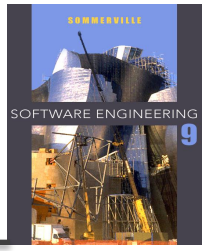
Después de terminar sus consultas, Kate regresa a la clínica y sube los registros de los pacientes visitados a la base de datos. El sistema genera una lista de llamadas para Kate de aquellos pacientes con los que tiene que contactar para

Pruebas de rendimiento



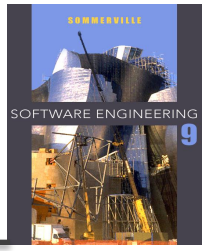
- Parte de las pruebas de lanzamiento pueden implicar la comprobación de las propiedades emergentes de un sistema, como el rendimiento y la fiabilidad.
- Las pruebas deben reflejar el perfil de uso del sistema.
- Las pruebas de rendimiento suelen implicar la planificación de una serie de pruebas en las que la carga se aumenta constantemente hasta que el rendimiento del sistema se vuelve inaceptable.
- Las pruebas de estrés son una forma de prueba de rendimiento en la que el sistema se sobrecarga deliberadamente para comprobar su comportamiento de fallo.

Prueba de usuario



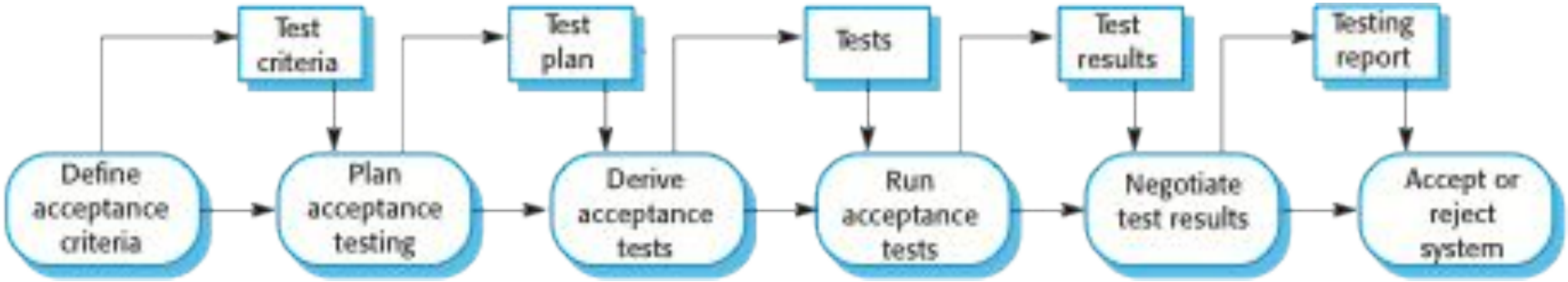
- La prueba de usuario o cliente es una etapa del proceso de prueba en la que los usuarios o clientes proporcionan información y asesoramiento sobre la prueba del sistema.
- Las pruebas de usuario son esenciales, incluso cuando se han llevado a cabo pruebas exhaustivas del sistema y de la liberación.
  - La razón de ello es que las influencias del entorno de trabajo del usuario tienen un efecto importante en la fiabilidad, el rendimiento, la facilidad de uso y la robustez de un sistema. Estos no pueden ser replicados en un entorno de pruebas.

Tipos de pruebas de usuario

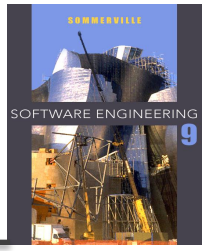


- La prueba del alfa
  - Los usuarios del software trabajan con el equipo de desarrollo para probar el software en el sitio del desarrollador.
- Pruebas Beta
  - Se pone a disposición de los usuarios una versión del software para que puedan experimentar y plantear los problemas que descubran con los desarrolladores del sistema.
- Pruebas de aceptación
  - Los clientes prueban un sistema para decidir si está listo o no para ser aceptado por los desarrolladores del sistema y desplegado en el entorno del cliente. Principalmente para sistemas personalizados.

El proceso de prueba de aceptación

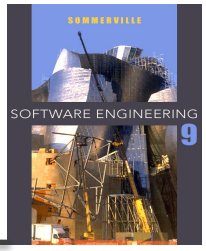


Etapas del proceso de pruebas de aceptación



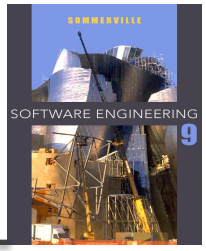
- Definir los criterios de aceptación
- Prueba de aceptación del plan
- Derivar las pruebas de aceptación
- Ejecutar pruebas de aceptación
- Negociar los resultados de las pruebas
- Sistema de rechazo/aceptación

Métodos ágiles y pruebas de aceptación



- En los métodos ágiles, el usuario/cliente forma parte del equipo de desarrollo y es responsable de tomar decisiones sobre la aceptabilidad del sistema.
- Las pruebas son definidas por el usuario/cliente y se integran con otras pruebas en el sentido de que se ejecutan automáticamente cuando se realizan cambios.
- No hay un proceso de pruebas de aceptación separado.
- El principal problema en este caso es si el usuario incorporado es "típico" y puede representar los intereses de todos los interesados en el sistema.

Puntos clave



- Cuando se prueba un software, se debe tratar de "romper" el software utilizando la experiencia y las directrices para elegir los tipos de casos de prueba que han sido eficaces para descubrir defectos en otros sistemas.
- Siempre que sea posible, deberías escribir pruebas automatizadas. Las pruebas están incrustadas en un programa que puede ser ejecutado cada vez que se hace un cambio en un sistema.
- El desarrollo de pruebas es un enfoque del desarrollo en el que las pruebas se escriben antes del código que se va a probar.
- La prueba de escenarios implica inventar un escenario de uso típico y utilizarlo para derivar casos de prueba.
- La prueba de aceptación es un proceso de prueba de usuario en el que el objetivo es decidir si el software es lo suficientemente bueno para ser desplegado y utilizado en su entorno operativo.