Tipos de probas

Existen distintas maneiras de clasificar as probas non excluíntes entre si:

- Funcionais ou non funcionais. As primeiras están destinadas a comprobar algún requisito funcional do software e as segundas non. Exemplos de probas non funcionais: as que permiten indicar o esforzo requirido para aprender a manexar ese software, para analizar o código e localizar un fallo, para soportar cambios e facilitar as probas asociadas a eses cambios, ou para migrar o software a outro entorno.
- Manuais e probas automáticas. As primeiras realizaranse seguindo un plan detallado e estruturado pero sen dispoñer dun proceso automático para executalas. As segundas dispoñen dun proceso automático que pode repetirse cantas veces se queira de forma cómoda.
- Dinámicas e probas estáticas. As primeiras realízanse mentres o software se está executando e as segundas non.

Existen operacións que realizan certas ferramentas sobre o código para detectar defectos e que non son estritamente probas, como por exemplo:

- Validación de código: Permite garantir que o código cumpre algún estándar da linguaxe.
- Depuración (*Debugging*): Permite detectar o código que dá resultados erróneos na execución. Os contornos de desenvolvemento dispoñen de ferramentas automáticas de depuración que permiten executar o código de maneira controlada polo programador e permitindo que o programador realice a execución liña a liña ou defina puntos de interrupción para que a execución se pare nese punto; cando se para a execución, o programador pode inspeccionar variables, expresións ou a pila e realizar modificacións sobre elas para ver como se comporta a execución.
- Análise de liñas de código: Permite detectar por exemplo: código repetido en varios sitios, revisar o código que está documentado e que facilitará a tarefa de xeración automática de documentación (Javadoc en Java), encontrar liñas de código comentado que ocupan espazo aínda que no se executen e que poden eliminarse utilizando un sistema de control de versións.

Probas unitarias

Son probas funcionais realizadas por persoal técnico que permiten detectar problemas nun módulo individual e elemental como por exemplo, un método dunha clase ou unha clase. Centran a súa actividade en exercitar a lóxica do módulo seguindo a estrutura do código (técnica de caixa branca) e as funcións que debe realizar o módulo atendendo ás entradas e saídas (técnica de caixa negra).

A porcentaxe de código probado mediante probas unitarias denomínase cobertura do software.

Os contornos de desenvolvemento dispoñen de ferramentas para deseñar e executar probas unitarias. Dentro das ferramentas dispoñibles no mercado destaca a familia XUnit: JUnit para Java, CppUnit para C++, PhpUnit para PHP e NUnit para .Net e Mono.

As probas unitarias son o tema central da actividade 3 desta unidade didáctica.

Probas de integración

Son probas funcionais realizadas por persoal técnico que permiten detectar problemas entre módulos relacionados e probados anteriormente de forma individual mediante probas unitarias. Deben ter en conta os mecanismos de ensamblaxe de módulos fixados na estrutura do programa, é dicir, as interfaces entre compoñentes da arquitectura do software. As probas de unidade e de integración solápanse e mestúranse no tempo normalmente.

Existen distintos tipos de probas de integración en función da orde de integración. A orde de integración elixida afecta a diversos factores, como: a forma de preparar casos, as ferramentas necesarias, a orde de codificar e probar os módulos, o custo da depuración, e o custo de preparación de casos. Os tipos fundamentais de integración son os seguintes:

- Integración incremental: Combínase o seguinte módulo que se debe probar co conxunto de módulos que xa foron probados. Existen dous tipos fundamentais:
  - Ascendente: Comézase polos módulos folla.
  - Descendente: Comézase polo módulo raíz.
- Integración non incremental: Próbase cada módulo por separado e logo intégranse todos dunha vez e próbase o programa completo. Denomínase tamén *Big-Bang* porque o número de módulos crece instantaneamente.

Integración incremental ascendente

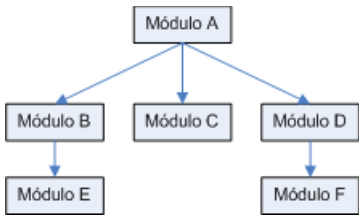
Na integración incremental ascendente empézase combinando os módulos de máis baixo nivel. As características da integración ascendente son:

- Pódese traballar con módulos de forma individual ou combinar os módulos de baixo nivel en grupos que realizan algunha función específica co obxectivo de reducir o número de pasos de integración.
- Escríbese para cada grupo un módulo impulsor ou condutor que é un módulo escrito para permitir simular a chamada aos módulos, introducir os datos de proba a través dos parámetros de entrada e recoller os resultados a través dos parámetros de saída.
- Próbase cada grupo empregando o seu impulsor.
- Elimínanse os módulos impulsores de cada grupo e substitúense polos módulos do nivel superior da xerarquía.

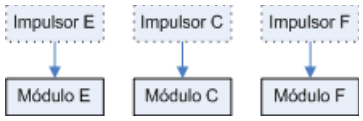
A construción de módulos impulsores non adoita ser moi complexa. Esta facilidade levou á creación de ferramentas automáticas capaces de realizar os labores dun impulsor. Normalmente están formados por:

- Instrucións para obter os datos necesarios para pasarlle ao módulo a probar.
- A chamada ao módulo a probar.
- Instrucións necesarias para mostrar os resultados devoltos polo módulo a probar.

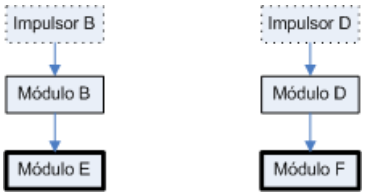
Por exemplo, as etapas de integración serían 6 no caso de contar cos módulos seguintes.



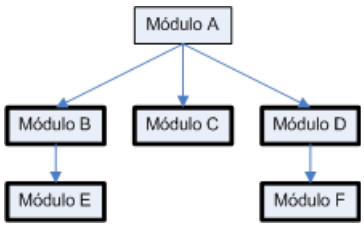
Etapas 1,2 e 3: Realízanse as probas unitarias de E, C e F. Os impulsores represéntanse cun borde punteado na seguinte gráfica.



Etapas 4 e 5: Realízanse por unha parte e de forma simultánea, a proba de unidade de B e a de integración (ou de interfaz) B-E, e por outra e de forma simultánea, a proba da unidade D e a interface D-F. Os módulos xa probados en etapas anteriores represéntanse cun borde groso na seguinte gráfica.



Etapa 6: Incorpórase o módulo A e próbase o programa completo, que non require impulsores, xa que todo o código de xestión da entrada e saída do programa está presente.



Integración incremental descendente

A integración incremental descendente comeza co módulo principal (o de maior nivel ou módulo raíz) e vai incorporando módulos subordinados progresivamente. Non existe unha regra xeral para determinar os módulos subordinados que convén incorporar primeiro. Como recomendación débense incorporar canto antes as partes críticas e os módulos de Entrada/Saída. Existen dúas ordes fundamentais de integración descendente:

- Primeiro en profundidade: vanse completando ramas da árbore modular. No exemplo anterior, a secuencia de módulos sería A-B-E-C-D-F.

- Primeiro en anchura: vanse completando niveis horizontais de xerarquía modular. No exemplo anterior, a secuencia de módulos sería A-B-C-D-E-F.

As características da integración descendente son:

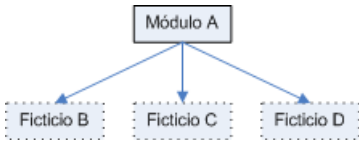
- O módulo raíz é o primeiro en probarse e escríbense módulos ficticios (*stubs*), para simular a presenza dos subordinados ausentes, que serán chamados polo módulo raíz.
- Unha vez probado o módulo raíz substitúese un dos subordinados ficticios polo módulo correspondente segundo a orde elixida, e incorpóranse ficticios para recoller as chamadas do último incorporado.
- Repítese o proceso detallado para o módulo raíz, é dicir, execútanse as correspondentes probas cada vez que se incorpora un módulo novo e ao terminar cada proba, substitúese un ficticio polo seu correspondente real. Convén repetir algúns casos de proba de execucións anteriores para asegurarse de que non se introduciu ningún defecto novo.

A codificación de módulos ficticios subordinados é máis complicada que a creación de impulsores. Os ficticios deben simular que recollen o control da chamada e que fan algo cos parámetros que se lles pasan. Existen diversos niveis de sofisticación dos ficticios:

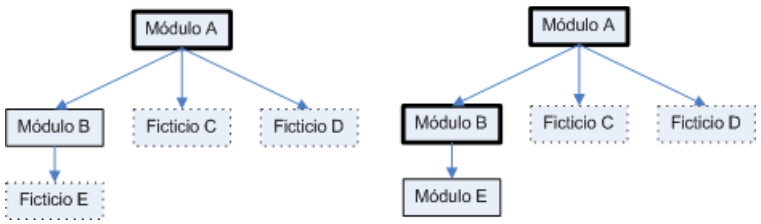
- Módulos que só mostran unha mensaxe de traza. Por exemplo, printf("Executouse o módulo 1");
- Módulos que mostran os parámetros que se lles pasan. Por exemplo, recibe o parámetro x e o mostra con printf("%d", x);
- Módulos que devolven un valor que non depende dos parámetros que se pasen como entrada (sempre devolve o mesmo valor, ou un valor aleatorio, etc.). Por exemplo, return(5); independentemente dos parámetros que reciba.
- Módulos que, en función dos parámetros pasados, devolven un valor de saída que máis ou menos corresponda a dita entrada. Por exemplo, devolver un valor utilizando as entradas para buscar nun array bidimensional, return(Táboa[x][y]); cando recibe x e y.

Por exemplo, utilizando o mesmo deseño de módulos que para a integración ascendente, e utilizando a orde primeiro en anchura, serían necesarias 6 etapas.

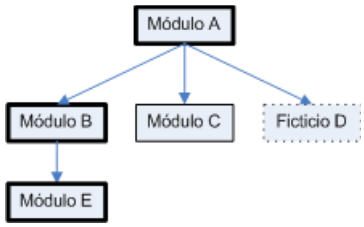
Etapa 1: Realízase a proba unitaria de A. Os ficticios represéntanse cun borde punteado na seguinte gráfica.



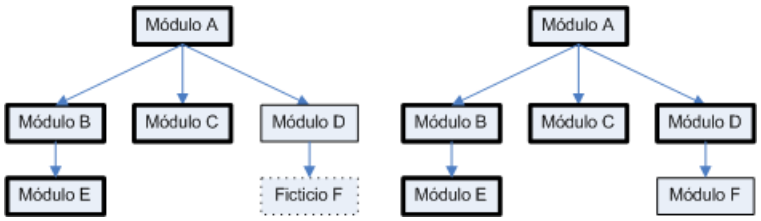
Etapas 2 e 3: Realízanse simultaneamente a proba de unidade de B e a de integración (ou de interface) A-B, e a continuación realízase simultaneamente a proba da unidade E e a interface B-E. Os módulos probados en etapas anteriores represéntanse cun borde groso na seguinte gráfica.



Etapa 4: Realízase simultaneamente a proba do módulo C e a interface A-C.



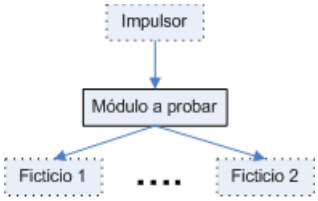
Etapas 5 e 6: Realízase simultaneamente a proba do módulo D e a interface A-D e a continuación a proba do módulo F e a interface D-F, quedando toda a integración probada.



Integración non incremental

A integración non incremental é o único caso no que as probas de unidade e integración están totalmente separadas. As características da integración non incremental son:

- Cada módulo require para ser probado:
  - Dun módulo impulsor que transmite os datos de proba ao módulo e mostra os resultados dos devanditos casos de proba.
  - Os módulos ficticios necesarios para simular a función de cada módulo subordinado ao módulo que imos probar.



- Despois de probar cada módulo por separado, se ensamblan todos eles dunha soa vez para formar o programa completo.

Comparación entre os distintos tipos de integración

A comparación entre as vantaxes da integración non incremental e a incremental queda reflectida na seguinte táboa:

Vantaxes da integración non incremental	Vantaxes da integración incremental
▪ Require menos tempo de máquina para as probas, xa que se proba dunha soa vez a combinación dos módulos. ▪ Existen máis oportunidades de probar módulos en paralelo.	▪ Require menos traballo, xa que se escriben menos módulos impulsores e ficticios. ▪ Os defectos e erros nas interfaces detéctanse antes, xa que se empeza antes a probar as unións entre os módulos. ▪ A depuración é moito máis fácil, xa que de detectar os síntomas dun defecto nun paso da integración, hai que atribuílo moi probablemente ao último módulo incorporado. ▪ Examínase con maior detalle o programa, ao ir comprobando cada interfaz aos poucos.

A comparación entre as características da integración incremental ascendente e descendente queda reflectida na seguinte táboa:

Integración incremental ascendente	Integración incremental descendente
▪ É vantaxosa cando hai defectos en niveis inferiores do programa. ▪ As entradas son máis fáciles de crear. ▪ O programa, como entidade, non existe até incorporar o último módulo. ▪ Requírense módulos impulsores. Os módulos impulsores son fáciles de crear. ▪ É máis fácil observar os resultados das probas.	▪ É vantaxosa cando hai fallos nos niveis superiores do programa. ▪ Antes de incorporar E/S, é difícil representar casos e as entradas poden ser difíciles de crear. Tras incorporar funcións de E/S, é fácil a representación de casos. ▪ Antes de incorporar o último módulo, vese a estrutura previa do programa. ▪ Require ficticios subordinados. Os ficticios subordinados non son fáciles de crear. ▪ Difícil observar resultados. ▪ Induce a atrasar a terminación da proba dalgúns módulos.

A selección dunha estratexia de integración depende das características do software e, ás veces, da planificación do proxecto. Algunhas recomendacións poderían ser:

- Identificar e probar canto antes aqueles módulos considerados críticos ou problemáticos.
- En xeral, o mellor compromiso pode ser un enfoque combinado (ás veces denominado *sandwich*) que consiste en:
  - Usar a integración descendente para os niveis superiores da estrutura do programa.
  - Utilízase simultaneamente a ascendente para os niveis subordinados.
  - Continúase ata que ambas as aproximacións atópense nalgún nivel intermedio.

Probas de sistema ou implantación

Son probas non funcionais realizadas por persoal técnico que permiten comprobar que o sistema completo, compre os requisitos funcionais e técnicos especificados e se relaciona correctamente con outros sistemas. Ademais realízanse outras probas como:

- Probas de sobrecarga ou estrés: Permiten avaliar o comportamento do sistema ao someter a unha situación límite como por exemplo demanda excesiva de peticións, utilización da máxima cantidade de memoria, traballar con pouca memoria, ter moitos usuarios realizando ao mesmo tempo a mesma operación, utilizar o máximo volume de datos.

- Probas de compatibilidade: Permiten probar o sistema en diferentes contornos, medios ou sistemas operativos e ver se hai fallos no aspecto ou no funcionamento.
- Probas de seguridade e acceso a datos: Permiten probar como responde o sistema ante ataques externos como por exemplo irrupcións non autorizadas ou camuflaxe de código malicioso nunha entrada de datos.
- Probas de recuperación e tolerancia a fallos: Permiten provocar anomalías externas como fallos eléctricos, de dispositivos, de software externo ou de comunicacións, e ver que o sistema se recupera sen perda de datos e sen fallos de integridade e o tempo que lle leva facelo.

Probas de validación ou aceptación

Son probas non funcionais realizadas por persoal non técnico e serven para comprobar se o produto final se axusta aos requisitos iniciais do software e está baseada nas accións visibles para o usuario e nas saídas do software que este pode recoñecer.

Poden existir probas alfa e/ou probas beta:

- As probas alfa asócianse normalmente ao software contratado. Realízaas o cliente nun contorno controlado baixo a supervisión da empresa que desenvolve o software. O desenvolvedor de software tomará nota dos posibles erros e problemas de uso.
- As probas beta asócianse normalmente ao software de interese xeral como por exemplo os paquetes ofimáticos pero tamén poden utilizarse para software contratado. Realízanas usuarios de confianza no seu entorno real de traballo sen control directo da empresa de software. O usuario informará dos resultados das probas.

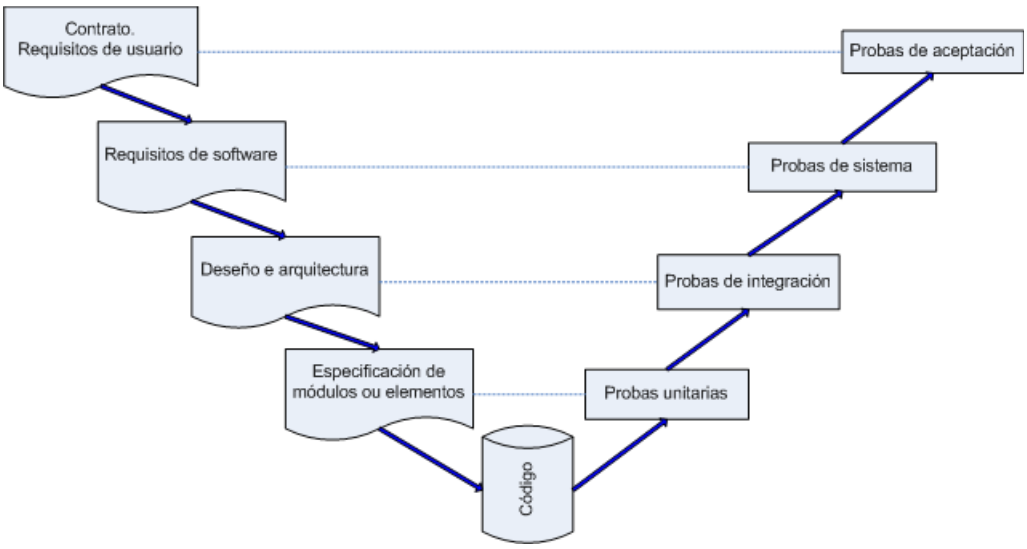
Probas de regresión

Son probas realizadas por persoal técnico para evitar efectos colaterais. Aplícanse cada vez que se fai un cambio no software para verificar que non aparecen comportamentos non desexados ou erros noutros módulos ou partes do software.

Un caso particular de cambio no software dáse cando se agrega un novo módulo nas probas de integración incremental. Pode ocorrer que un módulo que funcionaba ben deixe de facelo pola incorporación doutro. Neste caso as probas de regresión consistirán en volver a aplicar un subconxunto significativo das probas realizadas aos módulos xa integrados.

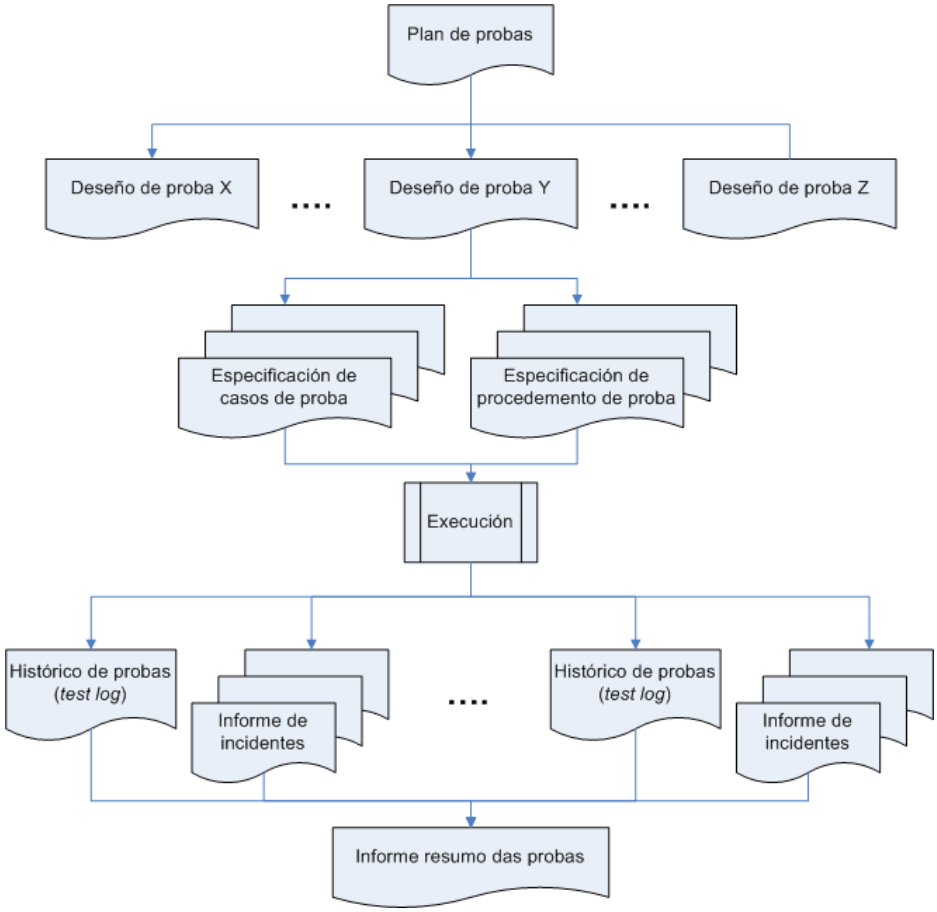
Estratexia de aplicación das probas no ciclo de vida clásico

A estratexia de aplicación e a planificación das probas pretende integrar o deseño dos casos de proba nunha serie de pasos ben coordinados, a través da creación de distintos niveis de proba, con diferentes obxectivos. En xeral a estratexia de probas adoita seguir durante o ciclo de vida do software as etapas que se mostran na seguinte imaxe.



Documentación do deseño das probas

A documentación das probas é necesaria para unha boa organización das mesmas, así como para asegurar a súa reutilización. Segundo o estándar IEEE 829, os documentos relacionados coas probas asócianse ás distintas fases das probas segundo se indica na gráfica seguinte.



O primeiro paso sitúase na planificación xeral do esforzo de proba (plan de probas) que inclúe o alcance do plan, os recursos a utilizar, a planificación das probas e as actividades a realizar. O segundo paso consiste no deseño das probas que xorde da ampliación e detalle do plan de probas e no que se especifican as características das diferentes probas. A partir deste deseño, especifícanse cada un dos casos de proba mencionados escuetamente no deseño da proba e especifícase como proceder en detalle e desenvolver a execución dos devanditos casos (procedementos de proba).

Tanto as especificacións de casos de proba como as especificacións dos procedementos deben ser os documentos básicos para a execución das probas. O último paso é o informe co resumo das probas no que se reflicten os resultados das actividades de proba e se achega unha avaliación do software baseada nos devanditos resultados.

A documentación da execución das probas é fundamental para a eficacia na detección e corrección de defectos, así como para deixar constancia dos resultados da execución das probas. Os documentos xerados para cada execución son:

- Histórico de probas: Documenta os feitos relevantes ocorridos durante a execución das probas.
- Informe de incidentes: Documenta cada incidente ocorrido na proba e que requira unha posterior investigación.