
MONITORIZACIÓN DE PROBAS

Título proxecto: Práctica de validación automática. Equipo 6

Ref. proxecto: <https://github.com/alejandrofortes/Practica-VVS>

Validación e Verificación de Software

Data de aprobación	Control de versións	Observacións
16/12/2015	1.0	

1. Contexto

O informe que estamos a ler, corresponde ao documento plan de probas da práctica de validación e verificación de probas do curso 2015-2016.

2. Estado actual

As funcionalidades e as súas especificacións actuais poden verse na figura 1.

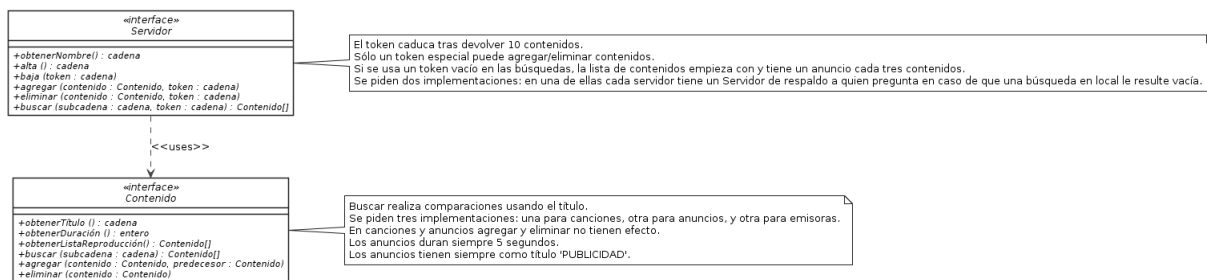


Figura 1: *Diagrama de clases do sistema*

O persoal encargado do proceso de proba está conformado por dúas persoas:

- Alejandro Fortes Lopes
- Ángel García Santos

Por último, para cada unidade, temos os seguintes datos:

- Nº de probas obxectivo: poden verse na sección “Automatización da execución de probas dinámicas”
- Nº de probas preparadas: poden verse na sección “Automatización da execución de probas dinámicas”
- Porcentaxe executada: 100 %.
- Porcentaxe executada: 100 %.

3. Rexisto de probas

3.1. Automatización da execución de probas dinámicas

Para a automatización da execución de probas dinámicas empregamos *JUnit*. Automatizamos as probas para as clases **Contenido** e **Servidor**. Empregamos a técnica de probas positivas (exercer o software en condicións normais de funcionamento).

No caso da clase **Contenido** (sendo as súas implementacións **Anuncio**, **Cancion** e **Emisora**) comprobamos o correcto funcionamento dos seguintes escenarios:

- Obter título
 - Escenario: obter o título do contido.
 - Entrada: un contido.
 - Saída: o título do contido.
 - Éxito: o título do contido é o agardado.
- Obter duración
 - Escenario: obter a duración do contido.
 - Entrada: un contido.
 - Saída: a duración do contido.
 - Éxito: a duración do contido é a agardada.
- Obter lista de reprodución
 - Escenario: obter a lista de reprodución do contido.
 - Entrada: un contido.
 - Saída: a lista de reprodución do contido.
 - Éxito: a lista de reprodución do contido é a agardada.
- Buscar
 - Escenario: procurar un contido que ten como título unha subcadea especificada.
 - Entrada: a subcadea especificada e o contido.
 - Saída: unha lista de contidos nos que a subcadea está contida no título do contido.
 - Éxito: que a búsqueda devolve os contidos agardados.

- Agregar¹
 - Escenario: engadir un contido a unha lista de reprodución.
 - Entrada: o contido a engadir e o predecesor do contido na lista de reprodución.
 - Saída: sen saída.
 - Éxito: o contido foi agregado na lista de reprodución.
- Eliminar²
 - Escenario: eliminar un contido de unha lista de reprodución.
 - Entrada: o contido a eliminar.
 - Saída: sen saída.
 - Éxito: o contido foi eliminado da lista de reprodución.

No caso da clase **Servidor**³ (sendo as súas implementacións **ServidorImpl** e **ServidorConRespaldoImpl** comprobamos o correcto funcionamento dos seguintes escenarios:

- Obter nome
 - Escenario: obter o nome do servidor.
 - Entrada: un servidor.
 - Saída: o nome do servidor.
 - Éxito: o nome do servidor é o agardado.
- Alta
 - Escenario: xerar e engadir un *token* ó servidor.
 - Entrada: un servidor.
 - Saída: o *token* engadido.
 - Éxito: o *token* foi engadido correctamente.
- Baixa
 - Escenario: eliminar un *token* do servidor.

¹Esta funcionalidade so está dispoñible para a implementación **Emisora**

²Esta funcionalidade so está dispoñible para a implementación **Emisora**

³Como a clase **ServidorConRespaldoImpl** estende á clase **ServidorImpl**, para a clase **ServidorConRespaldoImpl** so se proba o funcionamento do método **Buscar** porque é o único que sobrescribe

- Entrada: un servidor e o *token* a dar de baixa.
 - Saída: sen saída.
 - Éxito: o *token* foi dado de baixa do servidor de forma axeitada.
- Agregar
 - Escenario: engadir un contido a un servidor.
 - Entrada: o contido a engadir, un *token* mestre e o servidor no que se engade.
 - Saída: sen saída.
 - Éxito: o contido foi correctamente engadido no servidor en caso de que o *token* sexa de tipo mestre.
 - Eliminar
 - Escenario: eliminar contido dun servidor.
 - Entrada: o contido a eliminar, un *token* mestre e o servidor no que se elimina.
 - Saída: sen saída.
 - Éxito: o contido foi correctamente eliminado do servidor en caso de que o *token* sexa de tipo mestre.
 - Buscar
 - Escenario: procurar un contido no servidor no que o título conteña unha subcadea especificada.
 - Entrada: a subcadea especificada, un *token* e o servidor no que se procura.
 - Saída: unha lista de contidos nos que a subcadea está contida no título do contido.
 - Éxito: a búsqueda devolve os contidos agardados.

IMPORTANTE: Esta probas foron diseñadas e implementadas nun contexto no que as clases da asignatura non habían avanzado o suficiente como para poder aplicar todos os procedementos de proba explicados nas clases.

3.2. Automatización da xeración de datos en execución de probas dinámicas

Para a automatización da xeración de datos en execución de probas dinámicas escollemos a ferramenta *QuickCheck* (implementación en *Java* dos test basados en propiedades).

Empregamos a xeración de datos para xerar contidos e servidores de forma aleatoria e así poder probar o correcto funcionamento da implementación usando valores que normalmente non se nos ocorrerían.

No seguinte código podemos ver parte de unha das feitas con *QuickCheck*, comprobando os datos de dúas cancións xeradas de xeito automático, e introducindoas nunha emisora, comprobando así tamen os datos da emisora:

```
@Test
public void AgregarObtenerTituloYDuracionListaReproQC() {
    String titulo = generarString();
    Double duracion = generarDouble();
    Contenido cancion = new Cancion(titulo, duracion);

    String titulo2 = generarString();
    Double duracion2 = generarDouble();
    Contenido cancion2 = new Cancion(titulo2, duracion2);

    String tituloEmi = generarString();
    Contenido emisora = new Emisora(tituloEmi, 0);

    emisora.agregar(cancion, null);
    emisora.agregar(cancion2, cancion);

    assertTrue(emisora.obtenerTitulo().equals(tituloEmi));
    assertTrue(emisora.obtenerDuracion() == (duracion + duracion2));
}
```

3.3. Validación da calidade das probas

Para esta tarefa escollemos a ferramenta *PIT*, a cal, permitiunos obter tanto a cobertura de liñas código das nosas probas como a cobertura de código despois de aplicar mutacions no mesmo. Ademais, empregamos por separado a ferramenta de cobertura de código *Emma*.

O uso das dúas ferramentas mencionadas anteriormente mostra que a cobertura de liñas de código é superior o 97 %. Non acadamos o 100 % de cobertura

de código porque as clases `Anuncio` e `Cancion` non teñen implementación dos `Agregar` e `Eliminar`, polo que non poden ser probados.

Os resultados da execución da ferramenta poden visualizarse na figura 2.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage
5	97%  122/126	83%  44/53

Breakdown by Package

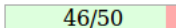
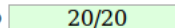
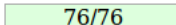
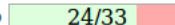
Name	Number of Classes	Line Coverage	Mutation Coverage
es.udc.fic.vvs.contenido	3	92%  46/50	100%  20/20
es.udc.fic.vvs.servidor	2	100%  76/76	73%  24/33

Figura 2: Cobertura de liña e de mutación

3.4. Execución de probas dinámicas de unidade

Debido a necesidade de probar as unidades de forma illada, recurrimos o uso de *mocks* (obxectos que imitan o comportamento de obxectos reais de forma contrlada), en concreto a ferramenta *Mockito*.

Simulouse o comportamento dos diferentes métodos das clases *Contenido* e *Servidor*. Unha vez simulado o comportamento desas clases, procedeuse a comprobar o correcto funcionamento dos mesmos mediante a execución automática de probas.

Un exemplo de simulación de comportamento de código sería facer que sempre que se pida o título dun anuncio, sempre se devolva a cadea “PUBLICIDAD”:

```
when(anuncio.obtenerTitulo()).thenReturn("PUBLICIDAD");
```

E o test que probe esta funcionalidade sería:

```
@Test
public void testObtenerTitulo() {
    assertEquals(anuncio.obtenerTitulo(), "PUBLICIDAD");
}
```

3.5. Probas non funcionas

Para as probas non funcionais empregamos a ferramenta *JETM*. Esta é una librería que permite localizar problemas de rendemento nun sistema.

Unha vez implementados os métodos que permiten medir o rendemento, os executamos e acadamos os resultados que se reflexan na figura 3.

As probas foron feitas executando mil chamadas a cada método. Como se pode apreciar na figura, non existen irregularidades nos tempos medidos, polo que entendemos que non existen problemas de rendemento graves.

3.6. Probas estruturais/estáticas

Nos decantamos por utilizar o analizador de código *PDM* para atopar de forma estática posibles erros.

Tras analizar os datos xerados pola ferramenta, correximos os erros máis críticos ou relevantes, pero a maioría os deixamos sen resolver porque consideramos que non eran o suficientemente críticos.

4. Rexistro de erros

- **QuickCheck:** atopouse e resolviuse un erro na clase *Emisora*. Non funcionaba correctamente ó intentar engadir un contido cando aínda non había predecesores.
- **Jetm:** non se atoparon erros de rendemento.
- **PDM:** atopáronse gran cantidade de erros e solucionáronse aqueles que consideramos que eran críticos ou importantes.

5. Estatísticas

- **QuickCheck:**
 - Número de erros encontrados diariamente e semanalmente: 0
 - Nivel de progreso na execución das probas: Medio
 - Informe de erros abertos e pechados por nivel de criticidade: erro na clase *Emisora*, que non permitía engadir contido cando aínda non existían predecesores. O erro está pechado actualmente.
 - Avaliación global do estado de calidade e estabilidade actuais: grazas a esta ferramenta incrementouse a calidade e estabilidade do sistema.

■ **Jetm:**

- Número de erros encontrados diariamente e semanalmente: 0
- Nivel de progreso na execución das probas: alto
- Informe de erros abertos e pechados por nivel de criticidade: os erros eran tan sinxelos que non se mantivo un rexistro de erros.
- Avaliación global do estado de calidade e estabilidade actuais: grazas a esta ferramenta incrementouse a calidade e estabilidade do sistema.

■ **PDM:**

- Número de erros encontrados diariamente e semanalmente: 593
- Nivel de progreso na execución das probas: medio
- Informe de erros abertos e pechados por nivel de criticidade: os erros eran tan sinxelos que non se mantivo un rexistro de erros.
- Avaliación global do estado de calidade e estabilidade actuais: grazas a esta ferramenta incrementouse a calidade e estabilidade do sistema.

6. Outros aspectos de interese

6.1. Revisión de código

Para facer una depuración de erros preliminar fixemos uso da técnica de probas chamada “revisión por un experto”, actuando no papel de “experto.º outro compañeiro de práctica.

6.2. *Travis*

Unha das prioridades que nos marcamos foi a de garantir que a estabilidade do proxecto fose óptima. Para iso, empregamos a ferramenta de integración contínua *Travis*. A utilización de dita ferramenta permitiunos asegurar que o sistema era estable en todo momento.

Poden verse as distintas execucións de *Travis* na figura 4.

Para máis información pode visitarse o *Travis* do proxecto en:

<https://travis-ci.org/alejandrofortes/Practica-VVS/>

[INFO] [EtmMonitor] JETM 1.2.3 started.					
Measurement Point	#	Average	Min	Max	Total
Anuncio:buscar	1	3,443	3,443	3,443	3,443
Anuncio:obtenerDuracion	1	0,399	0,399	0,399	0,399
Anuncio:obtenerListaReproduccion	1	0,151	0,151	0,151	0,151
Anuncio:obtenerTitulo	1	0,742	0,742	0,742	0,742
Cancion:buscar	1	1,078	1,078	1,078	1,078
Cancion:obtenerDuracion	1	0,199	0,199	0,199	0,199
Cancion:obtenerListaReproduccion	1	0,147	0,147	0,147	0,147
Cancion:obtenerTitulo	1	0,232	0,232	0,232	0,232
Emisora:agregar	1	1,477	1,477	1,477	1,477
Emisora:buscar	1	1,880	1,880	1,880	1,880
Emisora:eliminar	1	0,578	0,578	0,578	0,578
Emisora:obtenerDuracion	1	0,195	0,195	0,195	0,195
Emisora:obtenerListaReproduccion	1	0,182	0,182	0,182	0,182
Emisora:obtenerTitulo	1	0,215	0,215	0,215	0,215
Servidor:agregar	1	0,383	0,383	0,383	0,383
Servidor:alta	1	12,172	12,172	12,172	12,172
Servidor:baja	1	0,997	0,997	0,997	0,997
Servidor:buscar	1	0,856	0,856	0,856	0,856
Servidor:eliminar	1	0,252	0,252	0,252	0,252
Servidor:obtenerNombre	1	0,213	0,213	0,213	0,213
ServidorConRespaldo:buscar	1	1,041	1,041	1,041	1,041
[INFO] [EtmMonitor] Shutting down JETM.					

Figura 3: *Análise do rendimento do sistema*

alejandrofortes / Practica-VVS

build: passing

Current

Branches

Build History

Pull Requests

✓ master	Añadidas pruebas de rendimiento con jetm	⇨ #36 passed	⌚ 27 sec
🕒 alejandrofortes		🔑 dcb4772	📅 about 2 hours ago
✓ master	-Añadida dependencia al POM para QuickCheck.	⇨ #35 passed	⌚ 45 sec
🕒 Angel Garcia Santos		🔑 da34fe6	📅 3 days ago
✓ master	Corregido bug en buscar con respaldo. Devolvía una lista vacía en lugar	⇨ #34 passed	⌚ 38 sec
🕒 alejandrofortes		🔑 378a6c0	📅 4 days ago
✓ master	Cambio de TestCase a Junit	⇨ #33 passed	⌚ 35 sec
🕒 alejandrofortes		🔑 31bd99f	📅 14 days ago
✓ master	Merge branch 'master' of https://github.com/alejandrofortes/Practica-VVS.git	⇨ #32 passed	⌚ 21 sec
🕒 alejandrofortes		🔑 b58b498	📅 27 days ago
✓ master	Movidos los tests de servidor a un paquete propio para ellos.	⇨ #31 passed	⌚ 19 sec
🕒 Angel Garcia Santos		🔑 d9263ed	📅 30 days ago
✓ master	Hechos los tests para ServidorImpl. Corregidos algunos bugs en	⇨ #30 passed	⌚ 24 sec
🕒 Angel Garcia Santos		🔑 d2b47b7	📅 30 days ago
✓ master	Corregido metodo buscar de ServidorImpl, habia una comparacion entre u	⇨ #29 passed	⌚ 21 sec
🕒 Angel Garcia Santos		🔑 71e82af	📅 about a month ago

Figura 4: *Execuciones de Travis*