Pruebas software

PPROG

Content

- Definiciones de pruebas software
- Tipos
  - Pruebas unitarias
    - Caja blanca vs. caja negra
  - Pruebas de integración
  - Pruebas de regresión
- Desarrollo de pruebas unitarias
  - Documentación
  - Descripción de casos de prueba
  - Automatización de pruebas
    - Uso de aserciones
  - Informe de resultados

Algunas definiciones

- (Galin, 2004)
  - *Software testing is a formal process carried out by a specialized testing team in which a software unit, several integrated software units or an entire software package are examined by running the programs on a computer. All the associated tests are performed according to approved test procedures on approved test cases.*
- IEEE Standard Glossary of Software Engineering Terminology, (1990)
  - *The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system component.*
- (Craig & Jaskiel, 2002)
  - *Testing is a concurrent lifecycle process of engineering using and maintaining testware in order to measure and improve the quality of the software being tested*

Pruebas unitarias

- Prueban una unidad de código concreta → un módulo

- ¿Cómo?
  - Describir casos de prueba para cada función no trivial del módulo a probar
    - Cada caso es independiente del resto
    - Necesario describir pruebas para casos habituales, extremos y excepciones

- Pueden ser:
  - de *caja negra (blackbox-testing)*: comprobar que para una entrada se obtiene la salida esperada sin examinar la lógica del módulo
  - de *caja blanca* (*whitebox-testing*): se estudia qué pasos da el algoritmo y si los valores que han ido tomando los datos en cada caso han sido los correctos

Caja negra vs. Caja blanca

- Pruebas de caja negra (*Black-box testing*):
  - *Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.*
  - *Testing conducted to evaluate the compliance of a system or component with specified functional requirements.*

- Pruebas de caja blanca (*White-box testing*):
  - *Testing that takes into account the internal mechanism of a system or component.*

IEEE Standard Glossary of Software Engineering Terminology, (1990)

Pruebas de integración

- Prueban varias (o todas) unidades de código de forma conjunta
  - → Integración

- ¿Cómo?
  - Una vez que todas las pruebas unitarias han pasado
    - Los módulos funcionan correctamente de forma individual
  - Hay que garantizar que conjuntamente no produzcan errores

- Aconsejable:
  - Se deben realizar pruebas de integración incrementales a medida que se van probando que los módulos funcionan de forma individual
    - Garantiza que no se introducen nuevos errores cuando se integran los módulos

Pruebas de regresión

- Al modificar un sistema software se pueden introducir nuevos errores

- Es preciso no probar sólo el código nuevo/modificado sino garantizar el buen funcionamiento del resto del sistema
  - Una pequeña modificación de una función puede tener efectos negativos en otras funciones y módulos del sistema.

- Pruebas automatizadas que deben realizarse para comprobar que todo el sistema software sigue funcionando como se espera después de realizar modificaciones

Desarrollo de pruebas unitarias

- **Documento para la descripción de baterías de pruebas**
  - Descripción de casos de prueba para cada función de un módulo
    - Establecer entradas para cada caso → pre-condiciones
    - Establecer salidas para cada caso → post-condiciones

- **Automatización de pruebas**
  - Implementación de cada uno de los casos de prueba con la herramienta(s) o mecanismo(s) apropiado(s)

- **Informe de resultados de las baterías de pruebas**
  - Descripción detallada de las pruebas superadas y no superadas
  - Si es posible, indicar causa de error

Documento para la descripción de baterías de pruebas

No.	Módulo/Función	Objetivo	Pre-Condiciones	Pos-condiciones	Implementación
1	space/space_create	Prueba a crear un espacio	No hay	Referencia al espacio creado	test1_space_create()
2	space/space_create	Prueba a crear un espacio con un id concreto	Id del espacio	Id del espacio es el establecido	test2_space_create()
3	space/set_name	Prueba a establecer el nombre de un espacio	Cadena con el nombre	Salida Ok	test1_space_set_name()
4	space/set_name	Prueba a establecer un nombre en un espacio nulo	-Puntero nulo a espacio -Cadena con nombre	Salida error	test2_space_set_name()
5	space/set_name	Prueba a establecer un nombre nulo en un espacio	-Puntero al espacio -Cadena nula	Salida error	test3_space_set_name()
6	space/set_north	Prueba a establecer el id del espacio al norte	-Puntero al espacio -Id del espacio al norte	Salida Ok	test1_space_set_north()
…	…	…	…	…	…

Informe de resultados de las baterías de pruebas

No.	Módulo/Función	Responsable (Fecha realización)	Supera prueba	Supera prueba con valgrind	Breve descripción del error, de sus posibles causas y localización (sólo en caso de fallo)
1	Módulo/Función	Fulano Pérez Pérez (DD/MM/AAAA)	SÍ/NO	SÍ/NO	
2					

¿Puede generarse de forma parcial o total de modo automático?

Un ejemplo de automatización de pruebas unitarias

```
int main(int argc, char** argv) {
    int test = 0;
    int todas = 1;

    if (argc < 2) {
        printf("Pasando todas las pruebas al modulo Space:\t");
    } else {
        test = atoi(argv[1]);
        todas = 0;
        printf("Pasando prueba %d:\t", test);
        if (test < 1 && test > MAX_TEST) {
           printf("Error prueba no reconocida\t");
           exit(EXIT_SUCCESS);
        }
    }
…
```

Implementación de un programa de prueba

Un ejemplo de automatización de pruebas unitarias

```
    …
    if (todas || test == 1) test1_space_create();
    if (todas || test == 2) test2_space_create();
    if (todas || test == 3) test1_space_set_name();
    if (todas || test == 4) test2_space_set_name();
    if (todas || test == 5) test3_space_set_name();
    if (todas || test == 6) test1_space_set_north();
    if (todas || test == 7) test2_space_set_north();
    if (todas || test == 8) test1_space_set_south();
    if (todas || test == 9) test2_space_set_south();
    if (todas || test == 10) test1_space_set_east();
    …
    exit(EXIT_SUCCESS);
}
```

Lanzando cada prueba unitaria

Un ejemplo de automatización de pruebas unitarias

```
void test1_space_create() {
    if (space_create(5)!=NULL)
        printf("test1_space_create superado\n");
    else
        printf("test1_space_create no superado\n");
}

void test2_space_create() {
    Space *s;
    s = space_create(4);
    if (space_get_id(s) == 4)
        printf("test2_space_create superado\n");
    else
        printf("test2_space_create no superado\n");
}
```

Implementación de cada prueba unitaria

Más legible con algunas macros

```
void test1_space_create() {
  int result = space_create(5)!=NULL ;
  PRINT_TEST_RESULT(result);
}

void test2_space_create() {
    Space *s;
    s = space_create(4);
    PRINT_TEST_RESULT(space_get_id(s) == 4);
}

void test1_space_set_name() {
    Space *s;
    s = space_create(5);
    PRINT_TEST_RESULT(space_set_name(s, "hola") == OK);
}
```

Implementación de cada prueba unitaria

Más legible con algunas macros

```
void test3_space_set_name() {
    Space *s;
    s = space_create(5);
    PRINT_TEST_RESULT(space_set_name(s, NULL) == ERROR);
}

void test1_space_set_north() {
    Space *s;
    s = space_create(5);
    PRINT_TEST_RESULT(space_set_north(s, 4) == OK);
}

void test2_space_set_north() {
    Space *s = NULL;
    PRINT_TEST_RESULT(space_set_north(s, 4) == ERROR);
}
```

Implementación de cada prueba unitaria