Základní druhy softwarových chyb

Aleš Smrčka

Fakulta informačních technologií, Vysoké učení technické v Brně

Příklady pro přednášku kurzu IZP

6. prosince 2016

Základní pojmy:

Specifikace je popis korektního chování, činností, reakcí systému, včetně popisu vstupních a výstupních dat = popis vlastností produktu, např.:

- `assert(n_req <= n_ack + 1);` ≡ příchozích požadavků může být maximálně o jedna víc než obsloužených.
- $G(req \rightarrow F\ ack)$ ≡ kdykoliv přijde požadavek, někdy v budoucnu bude zpracován.

Fault (vada) je statický defekt v software.

Error (chyba) je špatný/nekorektní vnitřní stav systému, který je projevem nějaké vady.

Failure (selhání) je externí projev nesprávného chování (s ohledem na požadavky na systém).

Bug je obecný termín vyjadřující vadu, chybu nebo neočekávané (pravděpodobně špatné) chování systému.

Příklad vady, chyby

Specifikace: `numZero` vrácí počet prvků pole `x` s hodnotou 0.

```
int numZero(int x[], int length)
{
    int count = 0;
    for (int i = 1; i < length; i++)
        if (x[i] == 0)
            count++;
    return count;
}
```

Příklad vady, chyby

Specifikace: `numZero` vrácí počet prvků pole `x` s hodnotou 0.

```
int numZero(int x[], int length)
{
    int count = 0;
    for (int i = 1; i < length; i++)
        if (x[i] == 0)
            count++;
    return count;
}
```

Vada (fault) je v inicializaci cyklu `int i = 1`

Chyba (error) nastává ve všech případech polí s nenulovou délkou, tj. $[2, 7, 0]$ nebo $[0, 7, 2]$.

Selhání (failure) nastává v případě všech polí, jejichž první prvek je 0.

Základní pojmy (pokr.):

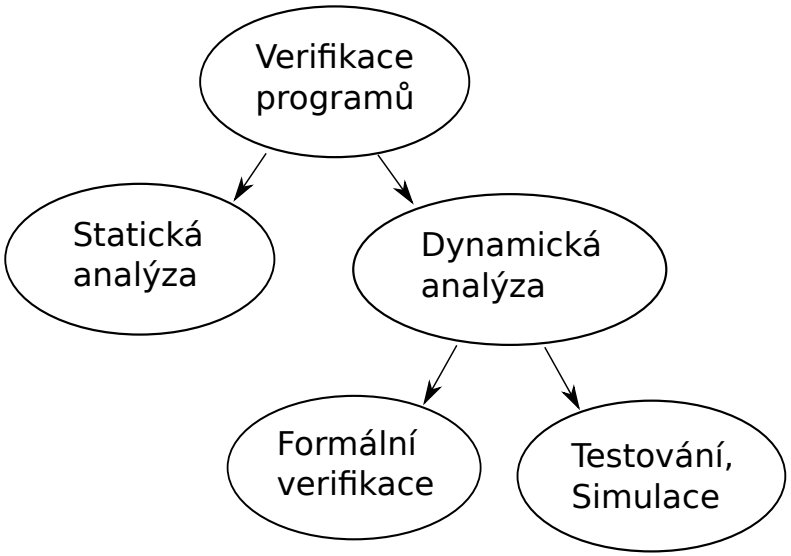
Statická analýza zkoumá vlastnosti SW bez jeho provádění.

Dynamická analýza ověřuje vlastnosti SW na základě provádění kódu.

Formální metody s pomocí matematických důkazů ověřují, že systém M odpovídá specifikaci φ: $M \models \varphi$

Testování zkoumá SW jeho systematickým spouštěním za účelem zvýšení jeho kvality.

Ladění (debugging) je proces hledání vady při znalosti selhání.



Základní druhy chyb

Základní druhy chyb:

1. **Překlep** — většina chyb; nezamýšlené omyly (nejen typografické), které lze po selhání systematickým laděním najít.
2. **Vynechání** — opomenutí nějakého případu (běhu, navrácených dat, ...)
3. **Chyba z neznalosti** — nedostatečná znalost jazyka, knihovních funkcí, API nebo architektury cílového stroje.
4. **Chyba specifikace** — chybě položená otázka při verifikaci programu.
5. **Chyba překladače** — většinou z jeho neznalosti, při zapnuté optimalizaci.
6. **Chyba v požadavcích** — opomenutí nějakého požadavku, nedostatečně popsaný požadavek.
7. ...

1. **Překlep** — většina chyb; nezamýšlené omyly (nejen typografické), které lze po selhání systematickým laděním najít.

Příklad chyby překlepem

```
int numZero(int x[], int length)
{
    int count = 0;
    for (int i = 0; i++; i < length)
        if (x[i] == 0)
            count++;
    return count;
}
```

2. **Vynechání** — opomenutí nějakého případu (běhu, navrácených dat, ...)

Příklad chyby vynecháním

```
int readRecords(FILE *input, List *list)
{
    int count = 0;
    ListItem *item, *pred;
    while (!feof(input))
    {
        item = newListItem();
        readData(input, item);
        item->next = NULL;
        if (count == 0)
            list->first = item;
        else
            pred->next = item;
        pred = item;
        count++;
    }
    return count;
}
```

2. **Vynechání** — opomenutí nějakého případu (běhu, navrácených dat, ...)

Příklad chyby vynecháním

```
int readRecords(FILE *input, List *list)
{
    int count = 0;
    ListItem *item, *pred;
    while (!feof(input))
    {
        item = newListItem();
        readData(input, item);
        item->next = NULL;
        if (count == 0)
            list->first = item;
        else
            pred->next = item;
        pred = item;
        count++;
    }
    return count;
}
```

Pokud je prázdný vstupní soubor, `list->first` je nedefinován.

3. **Chyba z neznalosti** — nedostatečná znalost jazyka, knihovních funkcí, API nebo architektury cílového stroje.

Příklad chyby z neznalosti

Záměr autora: Napsat kód na jeden řádek.

```
printf("Pocet zaznamu: %d, maximum: %d\n",
    readRecords(ifile, list), getMaxRecord(list));
```

3. **Chyba z neznalosti** — nedostatečná znalost jazyka, knihovních funkcí, API nebo architektury cílového stroje.

Příklad chyby z neznalosti

Záměr autora: Napsat kód na jeden řádek.

```
printf("Pocet zaznamu: %d, maximum: %d\n",
    readRecords(ifile, list), getMaxRecord(list));
```

Pořadí vyhodnocování parametrů není normou jazyka C definováno.

4. **Chyba specifikace** — chybě položená otázka při verifikaci programu.

Příklad chyby specifikace

Záměr autora: Počet příchozích pořadavků nReq odpovídá počtu odpovězeným nAck.

```
assert(nReq >= nAck);
```

4. **Chyba specifikace** — chybě položená otázka při verifikaci programu.

Příklad chyby specifikace

Záměr autora: Počet příchozích pořadavků nReq odpovídá počtu odpovězeným nAck.

```
assert(nReq >= nAck);
```

Tvrzení je platné téměř vždy. Lépe:

```
assert(nReq == nAck || nReq == nAck+1);
```

5. **Chyba překladače** — většinou z jeho neznalosti, při zapnuté optimalizaci.

Příklad chyby překladače (Crossware, 68k)

```
case '2':
    /* edit_recipe(choose_recipe()); */
    which = choose_recipe();
    edit_recipe(which);
    break;
```

Příklad chyby překladače (GCC 4.3.0, -O, IA32)

```
volatile int w;
int bar(void) {
    if (foo())
        return 0;
    else
        return w;
}
```

```
bar:
    subl $12, %esp
    call foo
    cmpl $1, %eax
    sbbl %eax, %eax
    andl w, %eax
    addl $12, %esp
    ret
```

Pokračování v dalších kurzech (ITS, FAV)