Formální metody - Z

Radek Mařík

ČVUT FEL, K13133

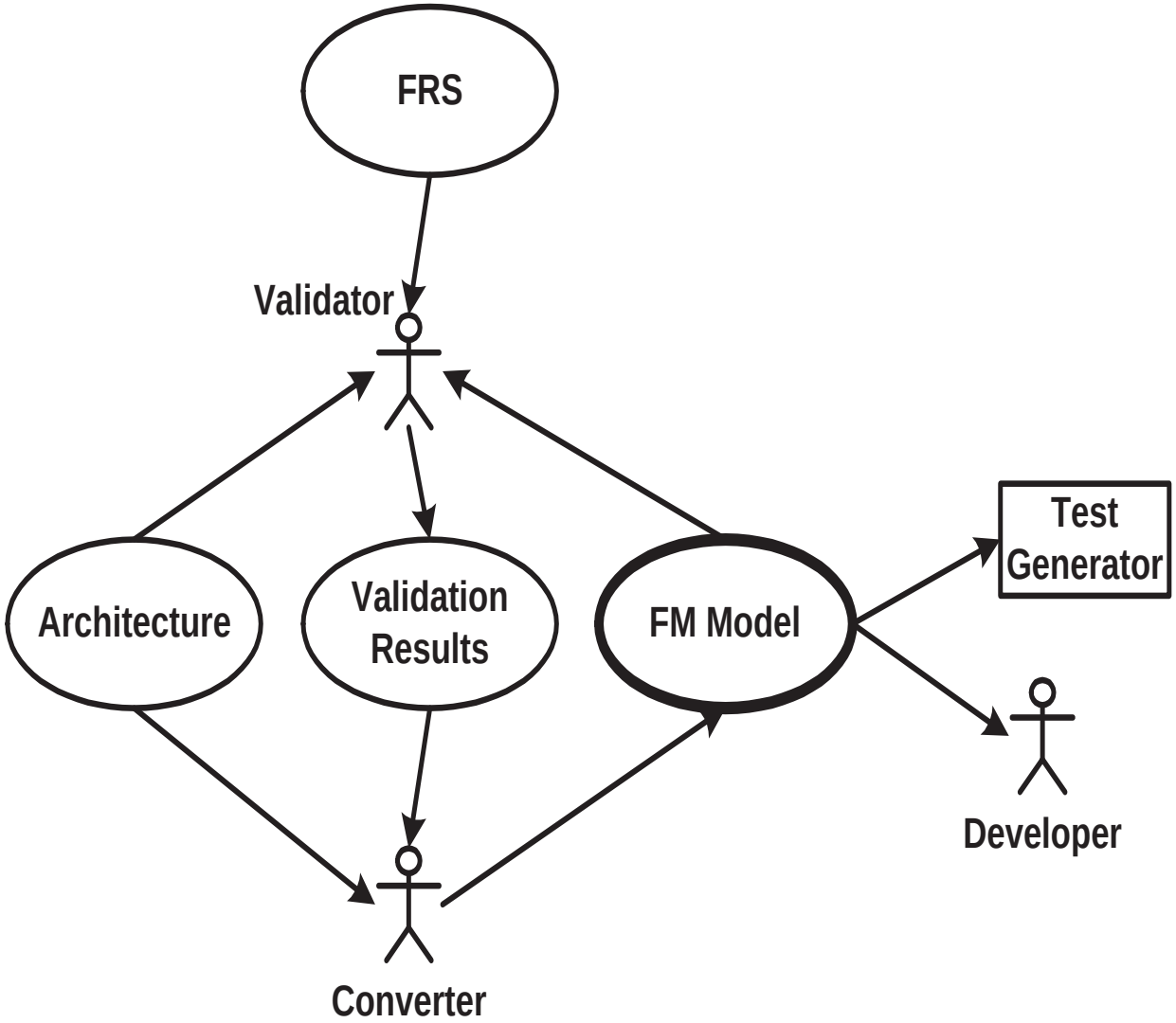
September 6, 2011

Obsah

Specifikace softwaru

- vstupní data pro přípravu testwaru,
- často prezentovány jako vágní představy,
  - chybějící data,
  - přeurčené položky - přeurčená data,
- volné věty nelze interpretovat počítačem,
- snaha o přesnější, stručné a strojem intepretované zapisy:
  - založeno na logice a jednoduché matematice,
  - vyžaduje další zápis popisu softwaru,
  - používá se pro
    - specifikaci,
    - návrh,
    - modelování,
    - verifikaci,
  - snaha o využití v nástrojích pro testování.

Použití formálních metod při testování



Příklad softwarového projektu [Jac97]

Systém řízení dokumentů

Výtah z neformálního popisu:

- Jestliže uživatel chce získat dokument ze účelem změny, má povolení ke změně, nikdo jiný v daný okamžik dokument nemodifikuje, potom uživatel může získat dokument pod svou správu.
- Jakmile jeden uživatel zamkne daný dokument za účelem editování, potom jej nikdo jiný nemůže rovněž zamknout (samozřejmě ostatní jej mohou číst pokud k tomu mají právo).
- Když uživatel ukončí editování dokumentu, měl by jej vrátit a odemknout a takto povolit jiným uživatelům provádět změny.

Formální metody [Jac97, WD96]

- aplikují logiku a jednoduchou matematiku na programování.
- znamenají (další) zápis formálního popisu softwaru.

Formální metody - praxe [Jac97, WD96]

Modelování Modely umožňují popis a predikci chování programu. Model je zjednodušená reprezentace. Formální metodu lze použít jako **oraculus**, nezávislý standard, který nás informuje o výsledcích testovacích běhů programů.

Návrh znamená organizaci interní struktury programu. **Členění** znamená rozdělení celého systémy na menší části nebo **moduly**, které mohou být vyvinuty nezávisle. **Opětné použití** jako základní technika složení systému z připravených stavebních bloků nebo **vícenásobně použitelných softwarových komponent**

Verifikace snaha prokázat, že implementace bude dělat to, co bylo zamýšleno. **Důkaz** jako demonstrační prostředek je založen pouze na specifikaci a textu programu, ne na běhu programu.

Z notace

- Z je jedna ze standardizovaných notací(ANSI standard).
- Z je právě jenom notace:
  - není exekuční,
  - není to programovací jazyk.
- Z je založena na teorii množin a matematické logice.
- Matematickou logikou je **predikátový kalkulus prvého řádu**.
- Z je založena na modelech. Systém je modelován pomocí
  - **stavu** - tj. sadě **stavových proměnných** a jejich hodnot
  - **operací**, které mohou měnit stav.
- **Schéma**: vzory deklarací a omezení.
- Charakteristickou vlastností *Z* jsou **typy**. Každý objekt má jednoznačný typ reprezentovaný jako maximální množina v rámci dané specifikace.

Z základní položky

Základní typ - deklarace nového typu.

$$[PERSON, DOCUMENT]$$

Proměnná - deklarace nového objektu.

$$joe, petr, sheila : PERSON$$

Zkratky

$$USERS == \{joe, petr, sheila\}$$
$$USER\_GROUP == \mathbb{P}\, PERSON$$
$$PERSON \leftrightarrow DOCUMENT == \mathbb{P}(PERSON \times DOCUMENT)$$

Z axiomatické popisy

$$maxUsers : \mathbb{N}$$

$$maxUsers \leq 100$$

$$permission : DOCUMENT \leftrightarrow PERSON$$
$$doug, aki, phil : PERSON$$
$$spec, design, code : DOCUMENT$$

- Deklarace nad čarou říká, že *maxUsers* je nezáporné číslo.
- Predikát pod čarou omezuje hodnoty deklarace.
- Položky v axiomatickém popisu jsou vždy **konstanty**.

Z k-tice

vázají na sebe v pevném uspořádání několik prvků jakéhokoliv typu.

$$DAY == 1 \,..\, 31$$
$$MONTH == 1 \,..\, 12$$
$$YEAR == \mathbb{Z}$$
$$DATE == DAY \times MONTH \times YEAR$$

$$\begin{array}{|l} landing, opening : DATE \\ \hline landing = (20, 7, 1969) \\ opening = (9, 11, 1989) \end{array}$$

Z projekční operátory

extrahují komponenty ze struktur.

$$first(aki, 4117) = aki$$
$$second(aki, 4117) = 4117$$

Příklad - specifikace práv

$$\begin{aligned} permission = \{ & \\ & (spec, doug), \\ & (design, doug), \\ & (design, aki), \\ & (code, aki), \\ & (code, phil) \\ & \} \end{aligned}$$

Z schéma

- pojmenovaný důležitý koncept,
- sada proměnných vázaných nějakými podmínkami,
- obecná formát

Name

declaration

---

predicate

- příklad

RightArrow

$ch? : CHAR$

$ch? = right\_arrow$

- konvence pojmenovávání komponent

vstup : ?,

výstup : !

Z dekorace

operace nad stavy používají se dvě kopie *stavu*:

- stav před operací,
- stav po operaci.

State

$$a : A$$
$$b : B$$

$$P$$

State′

$$a' : A$$
$$b' : B$$

$$P[a'/a, b'/b]$$

Z operace

OperationA

$State$

$State'$

$\ldots$

$\Delta State$

$State$

$State'$

OperationB

$\Delta State$

$\ldots$

Příklad schéma I

Dokument může být zamknut za účelem změny v daném časovém okamžiku pouze jednou osobou.
relace je parciální funkcí.

Documents

$$checked\_out : DOCUMENT \pfun PERSON$$

$$checked\_out \subseteq permission$$

Možný stav systému:

$$checked\_out = \{ (design, doug) (spec, doug) (code, phil) \}$$

Příklad schéma II

První operace měnicí stav:

CheckOut

$\Delta Documents$
$p? : PERSON$
$d? : DOCUMENT$

$d? \notin \mathrm{dom}\, checked\_out$
$(d?, p?) \in permission$
$checked\_out' = checked\_out \cup \{(d?, p?)\}$

Vstupní podmínky - predikáty, které neobsahují čárkované proměnné.

Příklad schéma III

Případy, kdy vstupní podmínka není splněna:

- dokument je již zamknut:

CheckedOut

$$\Xi Documents$$
$$d? : DOCUMENT$$
$$d? \in \mathrm{dom}\, checked\_out$$

- osoba nemá povolení:

Unauthorized

$$\Xi Documents$$
$$p? : PERSON$$
$$d? : DOCUMENT$$
$$(d?, p?) \notin permission$$

Příklad schéma IV

Celá operace pokrývá všechny tři možnosti:

$$T\_CheckOut \mathrel{\widehat=} CheckOut \lor CheckedOut \lor Unathorized$$

Dokazování ve výrokové logice - princip

- Výrok $p \wedge q$.
- K dokázání $p \wedge q$, jak p tak i q se musí dokázat.
- Za předpokladu platnosti $p \wedge q$ víme, že p musí být pravdivé, rovněž q musí být pravdivé.
- Shrnutí:

$$\frac{p \quad q}{p \wedge q} \quad [\wedge - \text{intro}]$$

$$\frac{p \wedge q}{p} \quad [\wedge - \text{elim1}]$$

$$\frac{p \wedge q}{q} \quad [\wedge - \text{elim2}]$$

Dokazování ve výrokové logice - inference

Inferenční pravidlo obecně

$$\frac{premiss_1 \cdots premiss_n}{conclusion} \quad [\ name\ ]$$

Literatura I

Jonathan Jacky.

The Way of Z: Practical Programming with Formal Methods.
Cambridge University, 1997.

Jim Woodcock and Jim Davies.

Using Z: Specification, Refinement, and Proof.
Prentice Hall, 1996.

JAPE demonstration

cd Radek/Papers/TestSemTAB

1. run jape
2. /File/Select top theory
3. JAPEHOME/EXAMPLES/SCS.jt
4. Prove
5. select the main window
6. by pointing apply the translation rules
7. quit JAPE with no saving