Functional Mockup Interface , Prostředek pro snadnou integraci simulačních nástrojů

Petr Huňka

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics
BioDat Research Group
http://bio.felk.cvut.cz

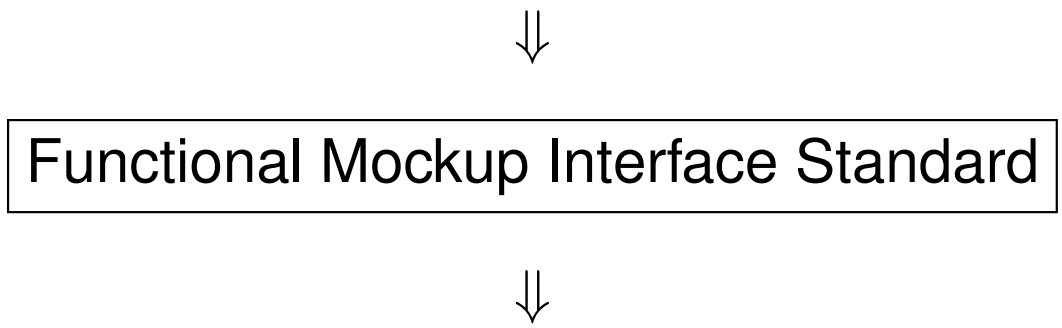
22.4.2013

Motivace

- Potřeba řešit velké a složité integrační simulační úlohy,
- existují stovky simulačních nástrojů - každý používá jiný formát,
- nutnost spojit několik simulačních nástrojů a modelů do jedné simulace.

Řešení

- Dva hlavní přístupy:
  1. Export modelů ze simulačního nástrojů a následný import do jiného
  2. Co-simulace modelů v různých simulačních nástrojích



- včasné zachycení chyb návrhu
- zvýšení kvality
- ušetřené peníze

Úvod 1

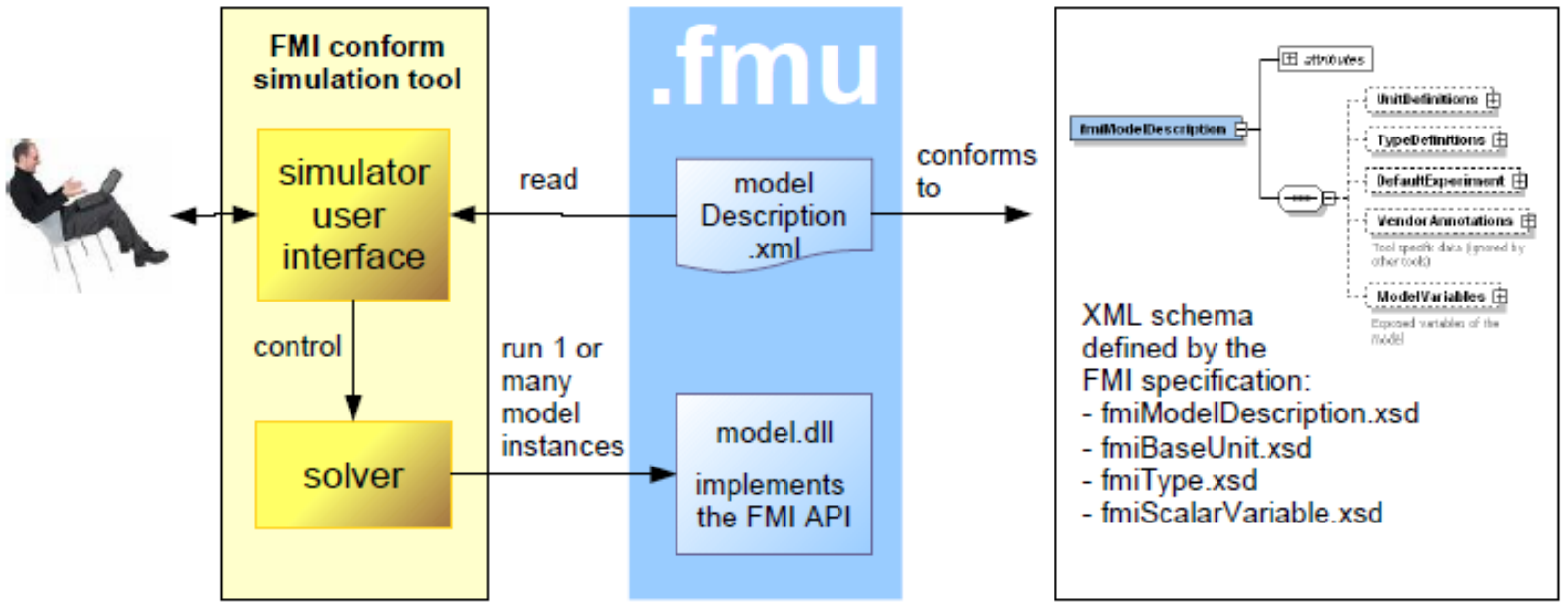
- Výsledek projektu ITEA2/Modelisar[1]
  - 3,5 roku
  - 30 mil EUR
  - 175 lidí
  - Německo, Francie, Švédsko, Belgie, Rakousko
- vznik inicioval Daimler AG
- zlepšení SW/Model/HW-in-the-loop simulace
- snadné znovupoužití vytvořených modelů
- snažší komunikace mezi výrobci a subdodavateli
- ochrana firemního know-how
- **otevřený standard**

[1] Information Technology for European Advancement

Úvod 2

- K dnešnímu dni jej podporuje 50 simulačních nástrojů a prostředí,
- v současné době je další vývoj zastřešen organizací Modelica Association
- dokončují se práce na FMI v.2.0

Simulace importovaného modelu



Functional Mockup Interface Prostředek pro snadnou integraci simulačních nástrojů

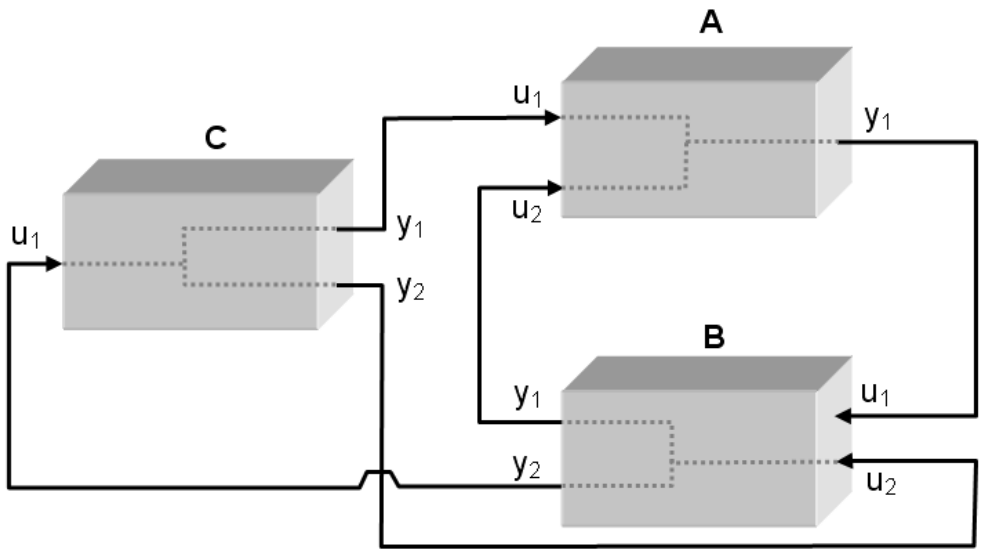
Petr Huňka

FMU - Functional Mockup Unit

- Komponenta, která implementuje FMI rozhraní se nazývá FMU,
- FMU je distribuován jako zip soubor obsahující:
  1. XML soubor popisující proměnné modelu a další popis
  2. C soubory, obsahující rovnice modelu převedené na jednotlivé funkce
  3. další data jako dokumentace, ikony, knihovny aj.

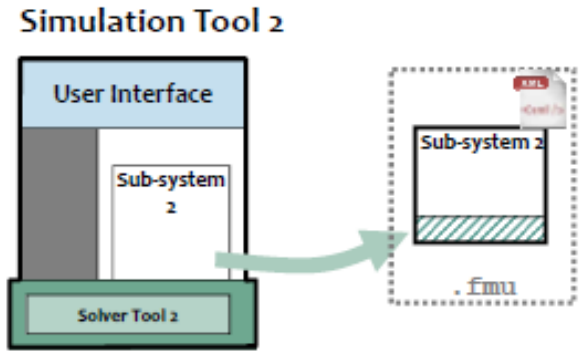
FMI for Model Exchange 1 - FMU

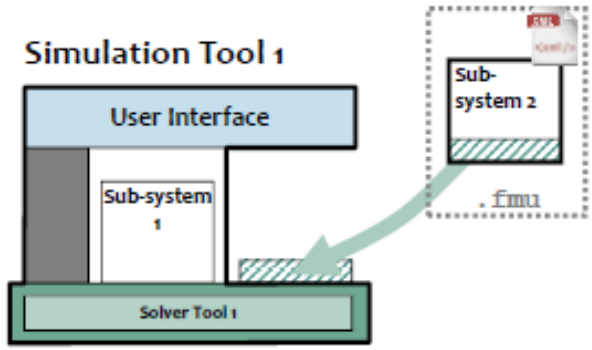
- Import a export in/out simulačních bloků (FMU)
- blok popsán jako:
  1. diferenciální-, algebraické, diskrétní rovnice,
  2. spolu s časovými, stavovými a skokovými událostmi
- FMU může:
  - být rozsáhlé 10^6 proměnných a 10^4 stavů,
  - použito v embedded systémech,
  - mohou se propojovat dohromady.



FMI for Model Exchange 2 - Použití

1. **Export** modelu subsystému ze simulačního nástroje
   - vytvoření fmu archivu obsahující
     - XML descriptor
     - spouštěcí dll (převedené rovnice)
     - případně C funkce (otevřený model)
2. **Import** modelu subsystému do simulačního systému, kde chceme simulaci provést
   - načtení FMU archivu
     - informace o modelu jsou obsaženy v XML descriptoru
     - propojení proměnných
     - spuštění simulace



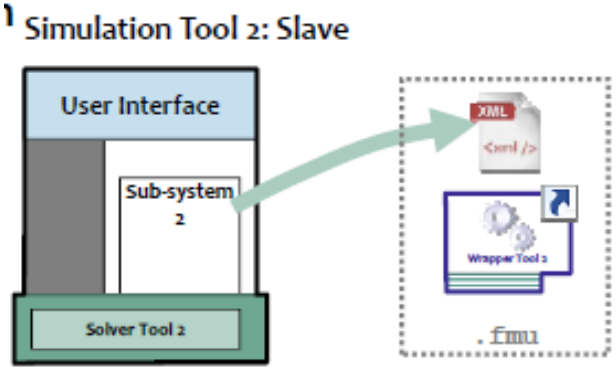


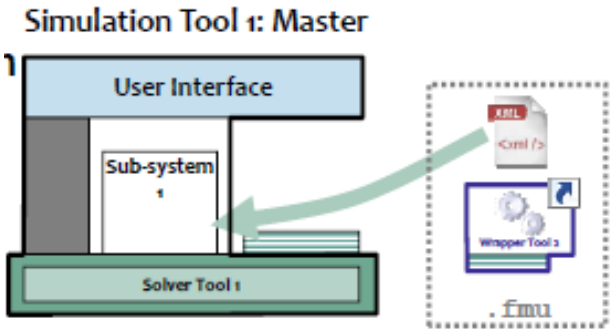
Functional Mockup Interface Prostředek pro snadnou integraci simulačních nástrojů

Petr Huňka

FMI for Co-Simulation 1 - Použití

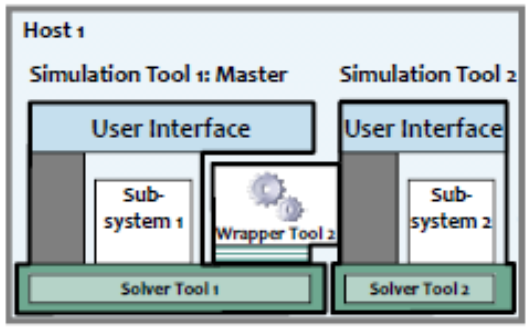
1. **Export** popisu subsystému ze simulačního nástroje
   - vytvoření fmu archivu obsahující
     - XML descriptor (**popisující možnosti solveru simulačního toolu**)
     - spouštěcí dll sloužící jako wrapper, obsahující proprietární implementaci sim. nástroje a co-simulačního slave rozhraní
2. **Import** popisu subsystému do simulačního systému, kde chceme simulaci provést
   - načtení FMU archivu
     - informace o modelu jsou obsaženy v XML descriptoru
     - propojení proměnných

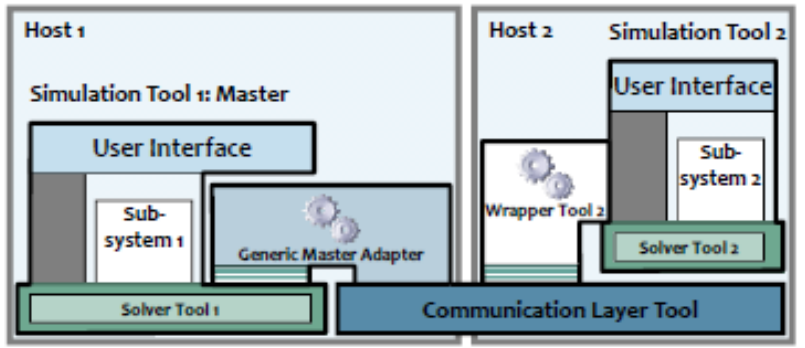




FMI for Co-Simulation 2 - Použití

1. **Spustění** simulace ve stejném prostředí (na jednom počítači)
   - Master subsystém je propojen se slave subsystémem (wrapper) via co-simulační rozhraní
   - Slave subsystem je volán via wrapper simulačního nástroje jako by byl importován přímo do master simulačního nástroje
2. **Spuštění** distribuované simulace
   - Master subsystém je připojen via generický adaptér a komunikační protokol (např. REST)
     - Adaptér umožňuje co-simulační slave rozhraní
   - komunikační protokol využívá wrapper slave subsystému





Distribuce fmi - bližší pohled

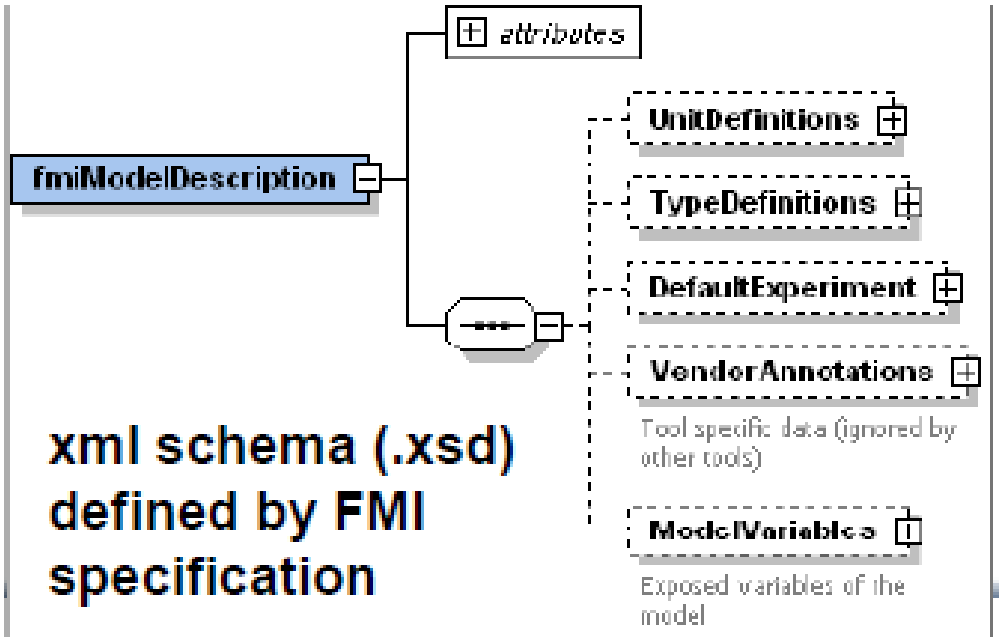
- Model je distribuován v jednom zip souboru (***.fmu**) obsahující:
  1. **XML model descriptor**
     Všechny informace o modelu potřebné pro jeho integraci.
     - při samotné simulaci se xml již nepoužívá (šetří systémové zdroje)
     - simulační nástroj může číst tyto informace preferovaným jazykem
  2. **rovnice modelu** definované malým počtem C-funkcí
     - zdrojové kódy v C (nepovinné - uzavření modelu)
     - binární kód (DLL) pro cílovou platformu
  3. **Další zdroje**
     - dokumentace (html)
     - ikona modeli (bitmapa)
     - mapy, tabulky, knihovny potřebné pro inicializaci modelu

Struktura FMU zip archivu

```
1. modelDescription.xml     // Description of model (required file)
2. model.png                // Optional image file of model icon
3. documentation            // Optional directory containing the model
                            // documentation
   _main.html               // Entry point of the documentation
   <other documentation files>
4. sources                  // Optional directory containing all C-sources
   // all needed C-sources and C-header files to compile and link the model
   // with exception of: fmiModelTypes.h and fmiModelFunctions.h
5. binaries                 // Optional directory containing the binaries
   win32 // Optional binaries for 32-bit Windows
      <modelIdentifier>.dll // DLL of the model interface implementation
      VisualStudio8         // Microsoft Visual Studio 8 (2005)
        <modelIdentifier>.lib  // Binary libraries
      gcc3.1              // Binaries for gcc 3.1.
   win64    // Optional binaries for 64-bit Windows
      ...
   linux32  // Optional binaries for 32-bit Linux
      ...
6. resources  // Optional resources needed by the model
   < data in model specific files which will be read during initialization >
```

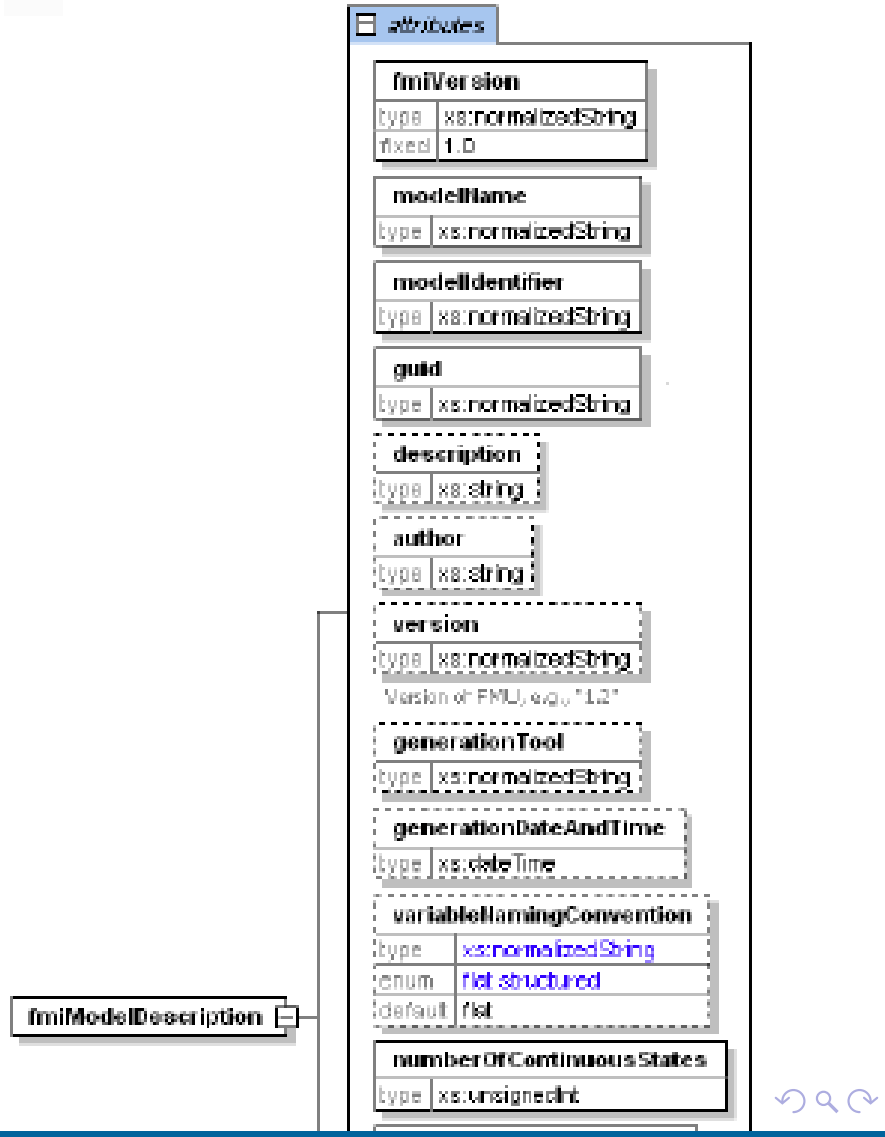
XML descriptor 1

- definice datových typů
- jména proměnných a jejich atributy
- specifická data simulačního nástroje



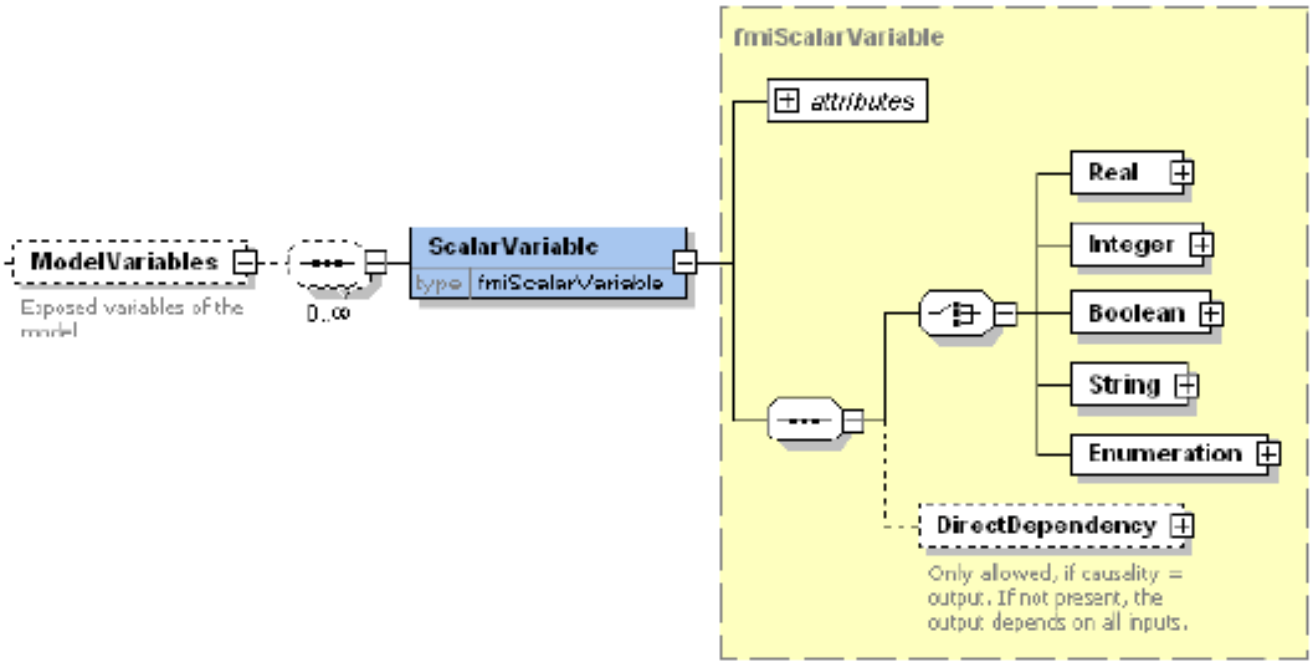
XML descriptor 2 - atributy modelu

- **modelIdentifier** - C proměnná, která je použita jako prefix pro C-funkce
- **guid** globální unikátní identifikátor (konzistence mezi xml a C-funkcemi)



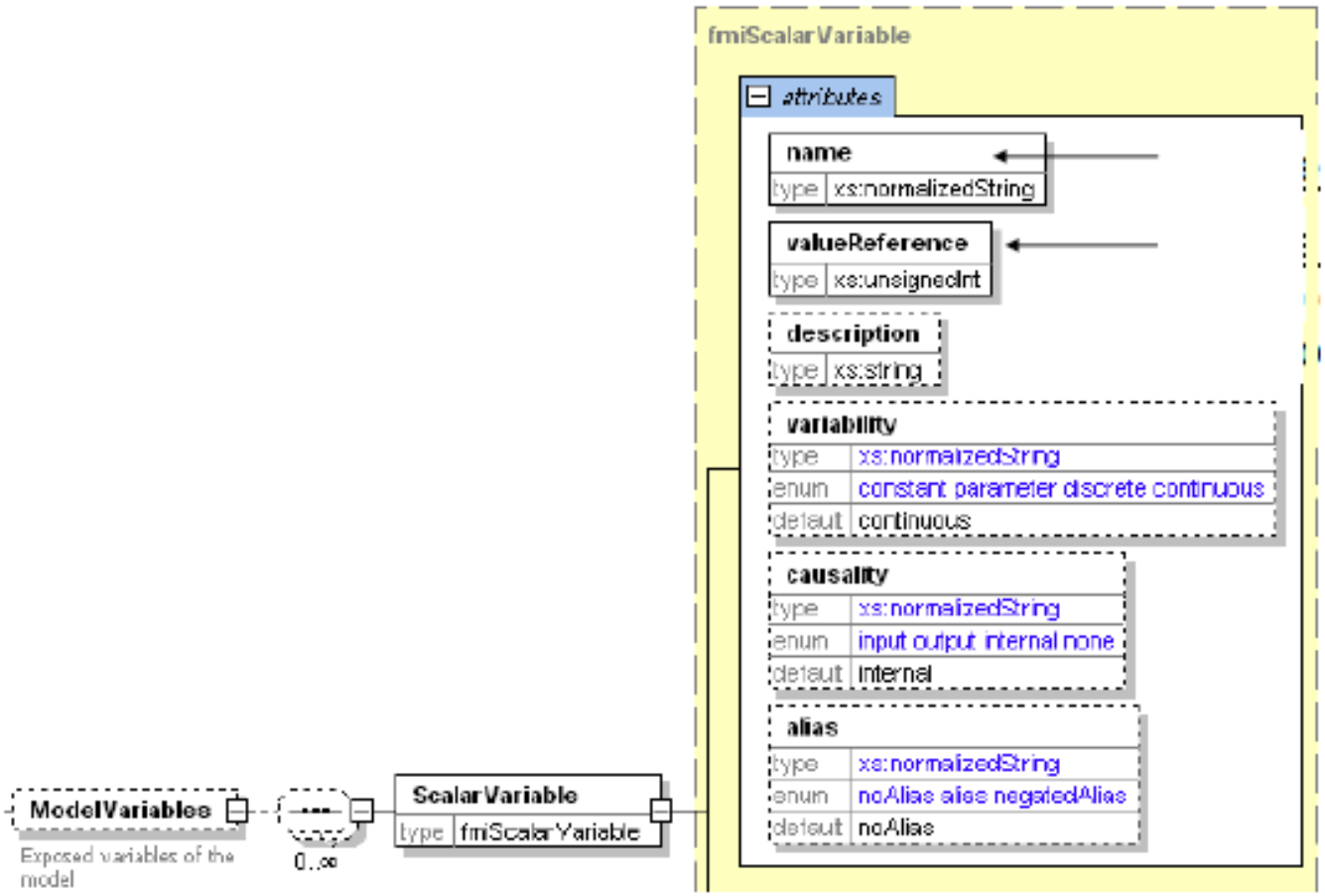
XML descriptor 3 - proměnné modelu

- setříděný seznam skalárních veličin
- pole, záznamy apod. jsou namapovány jako skaláry
- definují datové typy proměnných



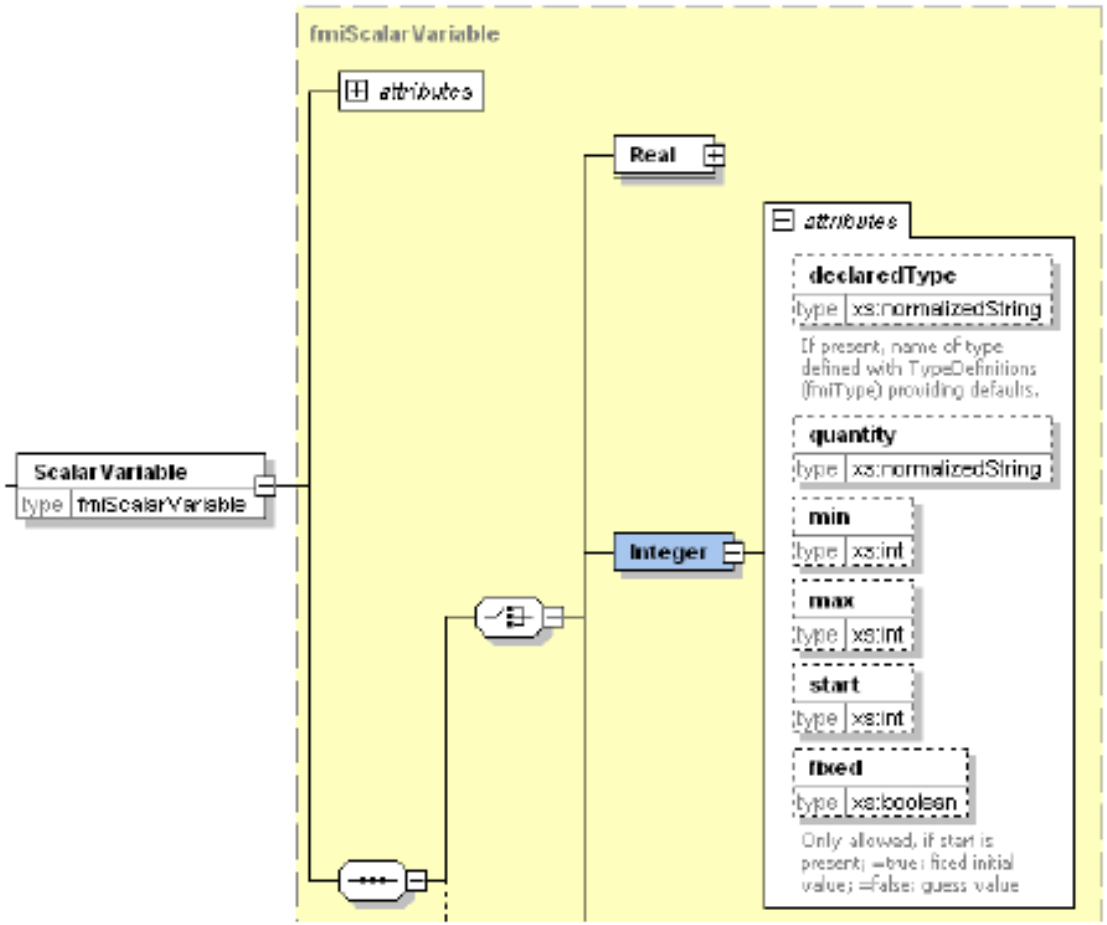
XML descriptor 4 - atributy proměnných modelu

- **name** unikátní jméno
- **valueReference** handle sloužící k propojení xml s C proměnnou



XML descriptor 5 - datové typy

- datové typy umožnují uložit všechny relevantní atributy (jednotky apod)



XML descriptor 5 - příklad modelu

```
model basic_eq
Real s( start = 20, unit = "m") "poloha";
constant Real g( unit="m.s-2")= 9.8;
Real v "rychlost";

initial equation
v = 0;

equation

v = der(-s);
der(v) = g;

end basic_eq;
```

XML descriptor 6 - příklad descriptoru

```
<?xml version="1.0" encoding="UTF-8"?>
<fmiModelDescription numberOfEventIndicators="0" numberOfContinuousStates="2"
variableNamingConvention="structured" generationDateAndTime="2013-04-12T01:27:07Z"
generationTool="Dymola Version 2013 (32-bit), 2012-03-28" guid="{eff22032-e0b4-426e-
85fd-82ec233bff15}" modelIdentifier="skoleniI_basic_0eq" modelName="skoleniI.basic_eq
fmiVersion="1.0">
    <DefaultExperiment tolerance="0.0001" stopTime="1.0" startTime="0.0"/>
  - <ModelVariables>
    - <ScalarVariable description="poloha" valueReference="33554432" name="s">
        <Real fixed="true" start="20" unit="m"/>
      </ScalarVariable>
    - <ScalarVariable description="der(poloha)" valueReference="587202560" name="der
      (s)">
        <Real unit="m/s"/>
      </ScalarVariable>
    - <ScalarVariable valueReference="100663296" name="g" variability="constant">
        <Real unit="m.s-2"/>
      </ScalarVariable>
    - <ScalarVariable description="rychlost" valueReference="33554433" name="v">
        <Real unit="m/s"/>
      </ScalarVariable>
    - <ScalarVariable description="der(rychlost)" valueReference="587202561" name="der
      (v)">
        <Real unit="m/s2"/>
      </ScalarVariable>
    </ModelVariables>
</fmiModelDescription>
```

C rozhraní - Dvě C hlavičky

- Platformně závislá definice základních typů

```c
/* Platform (combination of machine, compiler, operating system) */
#define fmiModelTypesPlatform "standard32"

/* Type definitions of variables passed as arguments */
   typedef void*         fmiComponent;
   typedef unsigned int  fmiValueReference;
   typedef double        fmiReal   ;
   typedef int           fmiInteger;
   typedef char          fmiBoolean;
   typedef const char*   fmiString ;

/* Values for fmiBoolean  */
#define fmiTrue  1
#define fmiFalse 0

/* Undefined value for fmiValueReference (largest unsigned int value) */
#define fmiUndefinedValueReference (fmiValueReference)(-1)
```

- Rozhraní C-funkcí
  - 18 core funkcí
  - 6 pomocných funkcí
  - žádná makra
  - jméno C fce odvozeno *ModelIdentifier*_*name*)

C rozhraní - hlavní C-funkce

description	range of t	equation	function names
initialization	$t = t_0$	$(\mathbf{m}, \mathbf{x}, \mathbf{p}, T_{next}) = \mathbf{f}_0(\mathbf{u}, t_0,$ subset of $\{\mathbf{p}, \dot{\mathbf{x}}_0, \mathbf{x}_0, \mathbf{y}_0, \mathbf{v}_0, \mathbf{m}_0\})$	fmiInitialize fmiGetReal/Integer/Boolean/String fmiGetContinuousStates fmiGetNominalContinuousStates
derivatives $\dot{\mathbf{x}}(t)$	$t_i \le t < t_{i+1}$	$\dot{\mathbf{x}} = \mathbf{f}_x(\mathbf{x}, \mathbf{m}, \mathbf{u}, \mathbf{p}, t)$	fmiGetDerivatives
outputs $\mathbf{y}(t)$	$t_i \le t < t_{i+1}$	$\mathbf{y} = \mathbf{f}_y(\mathbf{x}, \mathbf{m}, \mathbf{u}, \mathbf{p}, t)$	fmiGetReal/Integer/Boolean/String
internal variables $\mathbf{v}(t)$	$t_i \le t < t_{i+1}$	$\mathbf{v} = \mathbf{f}_v(\mathbf{x}, \mathbf{m}, \mathbf{u}, \mathbf{p}, t)$	fmiGetReal/Integer/Boolean/String
event indicators $\mathbf{z}(t)$	$t_i \le t < t_{i+1}$	$\mathbf{z} = \mathbf{f}_z(\mathbf{x}, \mathbf{m}, \mathbf{u}, \mathbf{p}, t)$	fmiGetEventIndicators
event update	$t = t_{i+1}$	$(\mathbf{x}, \mathbf{m}, T_{next}) = \mathbf{f}_m(\mathbf{x}^-, \mathbf{m}^-, \mathbf{u}, \mathbf{p}, t_{i+1})$	fmiEventUpdate fmiGetReal/Integer/Boolean/String fmiGetContinuousStates fmiGetNominalStates fmiGetStateValueReferences

```
// Set input arguments
fmiSetTime(m, time);
fmiSetReal(m, id u1, u1, nu1);
fmiSetContinuousStates(m, x, nx);

// Get results
fmiGetContinuousStates(m, derx, nx);
fmiGetEventIndicators (m, z, nz);
```

Srovnání FMI vs S-function

- **S-funkce**:
  - Stovky C-funkcí
  - inicializace modelu - 9000 řádek kódu
  - žádný simulátor nemůže importovat všechny S-funkce (moc komplexní)
  - distribuce několika dll (1 dll na cílovou platformu)
- **FMI**:
  - 20 C-funkcí
  - inicializace modelu - 200 řádek kódu
  - simulátor podporuje úplné rozhraní
  - FMI je distribuováno pouze v jednom dll pro všechny cílové platformy

Závěr

Děkujeme za pozornost