TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Validace konfiguračních souborů Flow123d

Magisterský projekt

M14000168

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Tomáš Křížek**
Vedoucí práce: Jiří Vraný, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Validation of Flow123d configuration files

Project report

M14000168

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technologies

Author: **Bc. Tomáš Křížek**
Supervisor: Jiří Vraný, Ph.D.

Liberec 2015

Tento list nahraďte

originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na můj magisterský projekt se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mého magisterského projektu pro vnitřní potřebu TUL.

Užiji-li magisterský projekt nebo poskytnu-li licenci k jeho využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Magisterský projekt jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mého magisterského projektu a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Tento projekt se zabývá problematikou validace konfiguračních souborů Flow123d, jejichž struktura je dynamicky definována v závislosti na verzi Flow123D. V projektu jsou popsány specifika formátu konfiguračních souborů, jako jsou reference nebo automatické konverze. Dále je navržena datová struktura pro zpracování konfiguračních souborů a dynamicky definovaných validačních pravidel. Nakonec je rozebrána problematika samotné validace.

Abstract

This project addresses the issues surrounding validation of Flow123d configuration files. Data structure of these files is dependent on the version of Flow123d. This paper describes the specifics of the configuration file format. These include references or automatic conversions. Next, a data structure for processing of configuration files and dynamycally defined validation rules is designed. Lastly, the issues of validation are discussed.

Obsah

Seznam zkratek 7

1 Úvod **8**

2 Formát konfiguračních souborů **9**

2.1 Autokonverze 10

2.2 Reference 10

3 Struktura konfiguračních souborů **11**

3.1 Pole 12

3.2 Záznam 12

3.3 Abstraktní záznam 13

Seznam zkratek

JSON JavaScript Object Notation

1 Úvod

V mém magisterském projektu jsem řešil problematiku validace konfiguračních souborů pro Flow123d. Tyto soubory mají dynamicky definovanou strukturu, která se liší podle verze Flow123d. Cílem je vytvořit nástroj, který určí, zda zadaný konfigurační soubor je validní pro danou verzi.

Konfigurační soubory jsou ve formátu CON, který se podobá standardnímu formátu JSON. Od tohoto formátu se mírně odlišuje syntaxí. Konkrétní odlišnosti jsou popsány v kapitole **??**. Kromě odlišné syntaxe také umožňuje vytvářet reference, se kterými je potřeba náležitě pracovat při zpracování dat. Práce s referencemi jsou popsány v kapitole **??**. Formát CON také umožňuje v některých případech použít zkrácený zápis, který se poté čtečkou formátu automaticky zkonvertuje na požadovaný tvar. Problematikou autokonverzí se zabývá kapitola **??**.

Jelikož se struktura konfiguračních souborů se liší podle verze Flow123d, validace nespočívá pouze v ověření dat podle pevně dané struktury. Pro validaci je nutné dynamicky načíst sadu pravidel, která strukturu popisuje. Sada těchto pravidel je dodána ve formátu JSON. V něm se nachází specifikace jednotlivých pravidel, které společně tvoří stromovou strukturu, stejně jako vstupní konfigurační soubor. Problematika použití sady pravidel popisující strukturu je popsána v kapitole **??**.

2 Formát konfiguračních souborů

Konfigurační soubory, které je potřeba zvalidovat, jsou často psány ručně a nejsou strvojově generovány. Formát CON, ve kterém jsou napsány, byl tomuto přizpůsoben. Syntaxe CON je podobná standardnímu formátu JSON, ale pro snažší zápis se mírně liší. Rozdíly v syntaxi oproti formátu JSON jsou následující.

- Klíče neobsahující mezeru mohou být napsány bez uvozovek.
- Klíč může být od hodnoty oddělen znakem „=“ místo „:“.
- V souboru se mohou vyskytovat komentáře. Jsou povoleny víceřádkové komentáře uzavřené mezi sekvencemi `/*` a `*/`, nebo řádkové komentáře, které jsou uvozeny sekvencí `//`.

Formát CON byl navržen pro inicializaci datových typů v C++. Kromě syntaktických rozdílů tedy předpokládá následující sémantická pravidla.

- Záznam má určitý typ, který definuje seznam povolených klíčů.
- Každý klíč má daný typ, jméno a implicitní hodnotu.
- Klíče napsané velkými písmeny mají speciální význam a jsou zpracovány čtečkou CON formátu. Názvy ostatních klíčů jsou malými písmeny.
- Povolené datové typy jsou skalární hodnoty, pole nebo záznamy.
- Pole jsou vždy tvořeny prvky stejného typu.
- U záznamů lze použít polymorfismus, kde klíč `TYPE` udává typ záznamu.
- Lze použít reference na data v rámci CON souboru pomocí speciálního klíče `REF`.

Bližší specifikace formátu CON lze nalézt v dokumentaci Flow123D.

2.1 Autokonverze

2.2 Reference

3 Struktura konfiguračních souborů

Jak již bylo zmíněno v kapitole 2, data v konfiguračním soubor podléhají určitým sémantickým pravidlům. Uvedená pravidla jsou pouze obecná a jejich konkrétní podoba je závislá na verzi Flow123D. Požadavky na strukturu jsou popsány v dokumentaci Flow123D a pro potřeby aplikace je jejich specifikace dodána ve strojově čitelném formátu JSON.

V této kapitole je popsán formát souboru specifikací. Nachází se v něm seznam datových typů, jejich popis, vlastnosti a struktura. Z jednotlivých položek lze vytvořit sada povolených datových typů, které lze použít v konfiguračním souboru. Každý datový typ má specifikované `id`, které ho jednoznačně identifikuje, a `input_type`, který určuje základní datový typ, od kterého je daný typ odvozen. Základními datovými typy jsou skalár (viz tabulka 1), pole (*Array*), záznam (*Record*) a abstraktní záznam (*AbstractRecord*).

Data v konfiguračním souboru tvoří stromovou strukturu. Pro jejich ověření se postupuje od kořene stromu k listům. Popis datového typu kořenu stromu je vždy uveden v seznamu jako první. Aby konfigurační soubor odpovídal dané specifikaci, musí být všechny uzly stromu validní. Pokud validní nejsou, zaznamená se, k jaké chybě došlo a kde.

Tabulka 1: Datové typy skalárních hodnot

Základní datový typ	**Popis**	**Pravidlo pro validaci**
Integer	celé číslo	hodnota musí být v rozmezí `min` a `max`
Double	desetinné číslo	hodnota musí být v rozmezí `min` a `max`
Bool	přepínač	–
String	řetězec	–
Selection	výběr z množiny	musí obsahovat hodnotu z množiny povolených hodnot
FileName	cesta k souboru	–

3.1 Skaláry

U skalárních typů spočívá validace v ověření správného datového typu a ověření validačního pravidla, pokud nějaké existuje. Pro číselné hodnoty se kontroluje rozmezí, které je určeno hodnotami `min` a `max` u specifikace datového typu. Speciální validaci vyžaduje také typ *Selection*. U tohoto datového typu je dána množina povolených hodnot (`values`), kde každá hodnota má jméno (`name`), které ji jednoznačně identifikuje. Zadaná hodnota v *Selection* je validní, pokud se shoduje se jménem některé z povolených hodnot.

3.2 Pole

Typ *Array* má podobně jako číselné typy specifikované hodnoty `min` a `max`. U pole tyto hodnoty určují minimální, resp. maximální počet položek. Pravidlo pro validaci pole tedy spočívá v oveření počtu položek pole. Dále je nutné ověřit, zda jsou validní jednotlivé položky v poli. Pole jsou vždy homogenní, tedy všechny jeho položky jsou stejného typu, který je určen pomocí `subtype`.

3.3 Záznam

Typ *Record* má unikátní jméno (`type_name`) a obsahuje libovolný počet klíčů. Pokud je potomkem nějakých abstraktních záznamů, tak je jejich seznam uveden v položce `implements`. Položka `keys` definuje seznam povolených klíčů. V ní jsou dále uvedeny tyto položky:

- `key`: definuje název klíče,
- `description`: popis klíče,
- `type`: typ klíče,
- `default`: obsahuje dále `type`, který určuje typ defaultní hodnoty a `value` pro implicitní hodnotu.

Typ defaultní hodnoty může nabývat hodnot *obligatory*, *value at declaration*, *value at read time* a *optional*. Pro potřeby validace je důležitá pouze hodnota *obligatory*, která specifikuje, že klíč musí být v záznamu uveden. Všechny ostatní typy

klíčů nemusí být v záznamu uvedeny. Validace záznamů spočívá v oveření, zda jsou přítomny všechny klíče, které jsou povinné (*obligatory*), a dále v oveření typu všech klíčů, které jsou v záznamu uvedené.

3.4 Abstraktní záznam

Speciální typ záznamu, který umožňuje polymorfismus, je *AbstractRecord*. Obsahuje položku `implementations`, ve které jsou uvedeny všechny zázamy, které ho implementují. Další důležitou položkou je `default_descendant`. Pokud je uvedena, tak definuje výchozí typ záznamu.

Při validaci záznamu je nutné nejprve určit jeho typ. K tomu slouží klíč `TYPE`, který je přímo uveden u daného záznamu v konfiguračním souboru. Pokud tento klíč není uveden, použije se typ, který je určen pomocí `default_descendant`. Pokud ani takto nelze určit typ záznamu, jedná se o chybový stav. Po určení typu záznamu se provede validace podle pravidel, které definuje přímo konkrétní datový typ.