Portál pre odborné publikovanie ISSN 1338-0087

Internetom podporované riešenie úloh v programovom balíku OpenModelica

Szolik Ladislav · Informačné technológie

24.09.2010

OpenModelica je program, ktorý slúži na simulovanie zložitých funkcií alebo modelov. Svojou funkčnosťou pripomína Matlab. Na rozdiel od neho je však poskytovaný bezplatne. Má jednoduché užívateľské rozhranie, ale v základnej verzii neobsahuje grafické rozhranie (GUI), ktoré by bolo podobné Simulinku. Dynamické modely systémov môžeme však vytvárať pomocou rôznych funkcií a podmienok. Vytvorený model môžeme simulovať vo vopred určených časových intervaloch s možnosťou vykreslovania grafických závislostí. Program je kompatibilný s rôznymi operačnými systémami, ako napríklad Windows, Linux alebo dokonca aj Macintosh. Oproti Matlabu je výhodou aj to, že nevyžaduje veľkú diskovú kapacitu.

1. Úvod

Program OpenModelica bol pôvodne vyvíjaný pre lokálne inštalácie a použitie. Úlohou tejto práce bolo preskúmať a vytvoriť komunikáciu, s ktorou môžeme jednotlivé funkcie lokálne nainštalovaného programu využiť pomocou internetového pripojenia. Na vybudovanie takejto komunikácie existujú rôzne spôsoby. Základ každého spôsobu je komunikácia medzi serverom a klientom.

Pri analýze problematiky sme najprv overovali komunikáciu pomocou COM objektov, ktoré sa dajú vytvárať pomocou skriptovacieho jazyka PHP. Týmto spôsobom je možné vytvárať webové stránky, ktoré sú schopné komunikovať napríklad s časťami programového balíka Microsoft Office číže napríklad s programami MS Excel alebo MS Word. V prípade programu MS Word, je možné po vytvorení nového dokumentu zmeniť jeho obsah a uložiť ho niekam na pevný disk. Podobné operácie je možné robiť aj s kalkulačkou, ktorá je zabudovaná do každého operačného systému. Všetky tieto operácie je možné vykonávať na strane klienta. Bohužiaľ, napriek tomu, že PHP dokáže pomocou COM objektov spolupracovať s rôznymi programami, vybudovanie komunikácie s OpenModelicou sa nám nepodarilo realizovať, pretože OpenModelica nepodporuje COM objekty.

Po ďalšej analýze možností komunikácie medzi rôznymi programami sme našli možnosť komunikácie pomocou mechanizmu CORBA (Common Object Requesting Broker Architecture). Ide o štandard, ktorý umožňuje vzájomnú spoluprácu rôznym softvérovým komponentom. Štruktúra programu OpenModelica umožňuje tento

program spustiť v móde InteractivCorba. To znamená, že program je schopný komunikovať s iným programom pomocou jedného rozhrania. Je to podobné riešenie ako komunikácia pomocou COM objektov. Na tejto metóde je následne postavená celá vyvinutá web aplikácia.

2. Riešenie problematiky

Ako už bolo povedané, pri vývoji web aplikácii sa je potrebné zaoberať zvlášť serverovskou a zvlášť klientskou aplikáciou, ktoré budú neskôr medzi sebou komunikovať. V tejto časti práce sa budeme postupne zaoberať všetkými týmito krokmi.

2.1. Strana servera

Hlavnou časťou aplikácie je časť servera. Tu prebiehajú rôzne simulácie, výpočty a tu sa generujú dynamické webové stránky. Na vytvorenie tejto časti sú potrebné nasledujúce komponenty:

- OpenModelica
- Apache server
- Python
- Mod Python
- Python Win 32 extension
- OmniORBpy

Program OpenModelica (ver. 1.4.5) tvorí základ webovej aplikácie. Inštaluje sa lokálne na server a budeme ho používať len v jednom režime, o ktorom sme už vyššie písali. Tento režim môžeme spustiť nasledujúcim príkazom z príkazového riadku:

```
omc +d=interactiveCorba
```

Interaktívny režim neobsahuje užívateľské rozhranie, beží len ako process v pozadí. Po spustení si program vytvorí svoje ID číslo, ktoré je uložené v externom súbore. Potom na základe tohto ID čísla bude prebiehať komunikácia. V ďalších fázach bude tento proces vykonávať jednotlivé výpočty a simulácie.

Apache http 2.2 server je program, ktorý tvorí webserver. Na zobrazenie webových stránok vytvorených pomocou Pythonu vo webových prehliadačoch je potrebné naištalovať aj doplnkové programy. Zároveň treba zmeniť nastavenia v konfiguračnom súbore Apache servera (httpd.conf), kde musíme povoliť vykonávanie cgi skriptov.

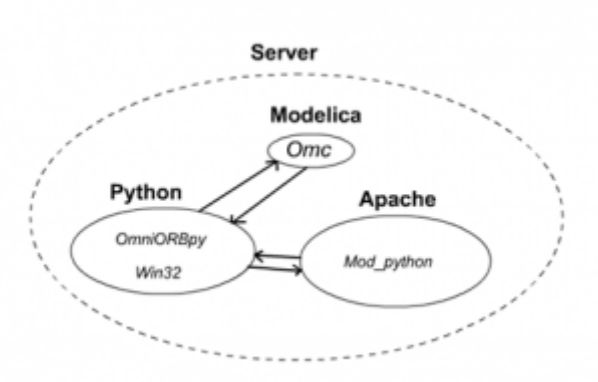
Skriptovací jazyk Python (ver 2.5), má podobný syntax ako C, ale vie vytvárať aj webové stránky. Z tohoto hľadiska je podobný aj jazyku PHP. Pomocou tohto jazyka vytvorená web aplikácia rieši rôzne úlohy. Posiela a prijíma dáta ako zo strany programu OpenModelica, tak aj zo strany klienta. Python nám teda umožňuje vytvoriť dynamickú webovú aplikáciu.

Program OmniORBpy (ver. 3.3) tvorí vnorenú časť programu Python. Umožňuje vytvoriť rozhranie medzi OpenModelicou a Pythonom. Knižnica Pythonu je rozšírená o vytvorené rozhrania.

Mod Python je tiež doplnkový program, ktorý rozširuje program Apache. Inštalácia programu je automatická, avšak nesmieme zabudnúť nastaviť miesto uloženia konfiguračného súboru Apache.

Python Win 32 extensions je program, ktorý rozširuje Python. Tento program je dobrý pre používateľov operačného systému Windows, ktorým poskytuje nové možnosti. Inštalácia je úplne automatická.

Vyššie uvedené komponenty môžu návzájom komunikovať bez vonkajšieho zásahu. Na ich správnu spoluprácu však musí byť splnených niekoľko podmienok. Sem patrí napríklad aj správna verzia programov. Návod na inštaláciu programov obsahuje aj popis kompatibility s inými programami. Vzájomná komunikácia medzi programami je znázornená na Obr.1.



Obr. 1. Komunikácia medzi programami.

2.2. Vytvorenie klientskej strany

Klientsku stranu tvorí dynamicky vytvorená webová stránka, ktorej základom je HTML dokument.

Významnú časť HTML stránky tvorí formulár, kde môže klient modifikovať parametre daného systému, nastavenie regulátora a parametre simulácie. Kvôli umožneniu riešenia čo najväčšieho spektra úloh je webový formulár navrhnutý tak, aby obsahoval, čo najviac údajov.

HTML stránka zároveň obsahuje zabudovaný SVG objekt, ktorý slúži pre grafické zobrazovanie výsledkov. Pretože lokálna inštalovaná verzia programu OpenModelica je schopná vytvárať grafikony, webová aplikácia musí mať tiež schopnosť tieto grafikony zobrazovať. Zároveň nám to umožňuje vytvoriť pre zadaný model animáciu. Animácia sa bude dynamicky meniť podľa toho, aké parametre zadá klient. Pri každej novej simulácii sa vygeneruje nová stránka, a následne sa zobrazí. Takto sme schopní simulovať chovanie reálnych systémov.

2.3. Spôsob komunikácie

V prvej časti komunikácie hrá hlavnú rolu klient. Na webovej stránke zadá potrebné údaje a odošle ich na server. Potom už len čaká a prijíma výsledky.

Po odoslaní údajov sa dá pozorovať komunikácia medzi webovou stránkou a Pythonom.

Následne Python pretransformuje údaje tak, aby boli zrozumiteľné pre OpenModelicu a odošle jej ich v podobe reťazca. Ako už bolo uvedené, OpenModelica beží v móde Interactiv CORBA. Pri komunikácii sa používa už spomínané ID programu.

Pythonom generované reťazce odpovedajú takým príkazom alebo modelom, ktoré by sa zadávali na lokálne spustenej OpenModelice pomocou klávesnice. OpenModelica tieto príkazy vykoná, v prípade zadaného modelu tento model vytvorí a výsledok prepošle do Pythonu. V prípade simulácie posiela späť len správu o úspešnom vykonaní. Údaje sa zapíšu do externého súboru. V tomto kroku nie je rozdiel medzi lokálne a interaktívne bežiacim módom, pretože aj v prípade lokálneho módu sa dáta tiež ukladajú do externého súboru.

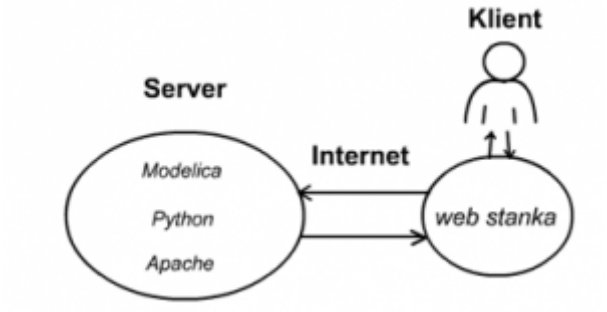
Keď sa spracováva len jednoduchý výpočet, tak sa posiela späť očakávaný výsledok. V prípade simulácie Python musí vykonávať ešte ďalšie úkony:

1. Otvorí súbor zahrňujúci výsledky simulácie. Samozrejme otváranie súboru je predom naprogramované, názvy súborov sú predom definované, ako aj ich umiestnenie.
2. Požadované hodnoty načíta zo súboru. Výsledkom zložitej simulácie môže byť viacero premenných. Súbor môže obsahovať okrem potrebných aj nepotrebné údaje, takže je potrebné v Pythone presne zadefinovať, ktoré údaje sa majú načítať.
3. Vykreslí grafické objekty. V tejto časti sa uskutoční vyhodnotenie výsledku, ktorý sa vykresľuje ako graf alebo animácia.

Vykreslením výsledku sa ukončí komunikácia. Klient má možnosť odoslať nové údaje, ktoré server môže znova prijať a spracovať.

Vytvorená aplikácia umožňuje aj kontrolu nesprávne zadaných údajov, ktoré je nemožné nasimulovať. Zlyhaniu systému sa snažíme zabrániť tak, že v Pythone neustále kontrolujeme vracajúce sa výsledky ktoré ešte pred samotným vykreslením najprv vyhodnotíme. Týmto riešením môže server bežať ďaľej, a je pripravený na ďaľšie výpočty.

Komunikačná bloková schéma sa nachádza na Obr. 2.



Obr. 2. Komunikačná bloková schéma

3. Realizácia a implementácia

Počas vývoja komunikácie medzi serverom a klientom sme vytvárali rôzne testovacie príklady. Najprv išlo len o vytvorenie jednoduchej kalkulačky, kde celý proces spočíval v tom, že klient zadal dve čísla a jeden operátor, a výsledkom bolo jedno číslo. Príkaz,

ktorý poslal program Python, pozostával z dvoch príkazových riadkov. Prvý riadok sa skladal z reťazca, ktorý obsahoval dve čísla a typ operácie, t.j. údaje načítané z formulára. Po odoslaní tohto príkazu bol odoslaný ešte príkaz pre vypísanie výsledku.

Ďalšia aplikácia bola zameraná na vykreslenie ľubovolnej funkcie Klient mal za úlohu zadať funkciu a zadefinovať jej rozsah hodnôt. V tomto prípade výsledkom bol graf, ktorý bol vytvorený na základe vypočítaných hodnôt. Stačilo len vytvoriť krátku funkciu a zobraziť je výsledok. Príkazy, ktoré prichádzali do programu OpenModelica už boli trošku zložitejšie, a preto bola nutná definícia jednej funkcie a zároveň aj výpisu výsledku.

Následný príklad už spočíval vo vytvorení dynamického modelu. Išlo o aktiváciu modelu skákajúcej lopty, ktorý je preddefinovaný v programe OpenModelica. Jedná sa o vopred vytvorený model, ktorý vypočíta polohu lopty v závislosti na čase. Lopta je spustená z istej výšky a užívateľ má možnosť zadať hodnotu tejto výšky, t.j. má možnosť meniť jeden parameter. Výsledky sa vizualizovali pomocou svg animácie. Po zadaní hodnoty užívateľom sa poloha guličky dynamicky menila a pomocou Pythonu sa dala animácia zobrazovať v ideálnych pomeroch.

Poslednou vytvorenou web aplikáciou bola animácia reálneho systému magnetickej levitácie, ktorý sa nachádza v laboratóriach Ústavu riadenia a priemyselnej informatiky FEI STU (Obr. 3). Našou úlohou bolo vytvoriť virtuálny experiment, ktorého súčasťou bude už spomenutá animácia a možnosť odsimulovania riadenia za pomoci PID regulátora. Tento príklad je na rozdiel od predchádzajúcich aplikácií o niečo zložitejší, keďže vyžaduje prepojenie viacerých modelov. Spojenie týchto modelov umožňujú spojovacie konektory, ktoré sú taktiež súčasťou programu OpenModelica.

Obr. 3. Reálny model magnetickej levitácie

Model systému sa dá realizovať v programe OpenModelica pomocou štyroch blokov:

1. 2 bloky pre PID regulátor
2. blok vytvorený pre model magnetickej levitácie
3. hlavný blok obsahujúci všetky ostatné bloky

Klientska aplikácia (Obr. 4) môže ovládať daný systém pomocou preddefinovaného PID regulátora, čím ovplyvňuje polohu kovovej guličky v magnetickom závese. Na strane

klienta sa zadávajú hodnoty, ktoré ovplyvňujú chovanie systému a pomocou Pythonu sa odosielajú obvyklým spôsobom. Ako aj v prípade predošlej animácie, aj tu sa jedná o simuláciu modelu, ktorý je opísaným matematickým aparátom. Rozdiel je v tom, že každý údaj, ktorý používa systém pri výpočtoch, sa dá modifikovať na strane klienta. Obdržaný výsledok po simulácii sa zobrazí vo forme animácie. V skutočnosti sa v tomto prípade nejedná o jednoduchú animáciu, ale o dve navzájom synchronizované animácie.

Obr. 4. Klientska aplikácia virtuálneho experimentu magnetickej levitácie

4. Vyhodnotenie

Predstavený spôsob komunikácie je samozrejme len jedným z viacerých možností. Dokumentuje však možnosť ako vytvoriť z lokálnej inštalácie programu aplikáciu, ktorú je možné implementovať na server. V budúcnosti sa plánuje ďalšie rozšírenie tejto aplikácie.

Literatúra

1. ADRIAN POP, PETER FRIZON,OpenModelica Users Guid Dostupné z http://www.ida.liu.se/~pelab/modelica/OpenModelica/
2. Dokumentácia programu Python. Dostupné z http://www.python.org/doc/
3. Dokumentácia programu OmniORBpy Dostupné z http://omniorb.sourceforge.net/docs.html
4. ADRIAN POP, PETER FRIZON, OpenModelica. Dostupné z http://www.ida.liu.se/~pelab/modelica/OpenModelica/

Spoluautorkou tohto článku je Katarína Žáková, Fakulta elektrotechniky a informatiky, Slovenská technická univerzita, Ilkovičova 3, 812 19 Bratislava