Primerjalna analiza izbire platforme za numerični model proizvodne celice (Katarina Mramor)

Igor Grešovnik, Februar 2011

V spodnji tabeli sem puskusil zajeti razlike glede na to, na kateri platformi bi delala Katarina. Prva možnost je C# na osnovi IGLib in stvari, ki so pripravljene za NAFEMS test, druga pa v C++ v kodi Gregorja Kosca. Gregorjeve kode še nisem dobil, da bi jo lahko malo pregledal, tako da nekatere ocene temeljijo na tem, kar sem videl na sestanku z njim jeseni in na tem, kar sta mi pozneje povedala Gregor Kosec in Robert.

V tabeli sem s K označil stvari, ki so pommebne predvsem za Katarino, s S pa stvari, ki so pomembne predvsem za širšo skupino.

Slabosti, če Katarina začne svoj razvoj v C#	
C#	C++, iz Koščeve kode
K, S. C# je počasnejši od C++. Po do zdaj zbranih podatkih razlike niso tako bistvene in bi imela v praksi precej večji vpliv dobra implementacija numeričnih algoritmov. Če bi bile razlike zaradi jezika kritične, se še vedno lahko v kritičnih delih kode odpove varnemu načinu in se uporablja hitrejše kazalce.	C++ je v splošnem hitrejši od C#.
K, S. V C# je napisanih precej manj knjižnic, predvsem v prosti domeni na področju numerike, jezik je precej mlajši kot C++. Kritičnih manjkajočih členov po drugi strani nismo našli. Če je potrebno, se lahko linkajo tudi fortranske ali Cjevske knjižnice. Po drugi strani je standardizirana baza jezika precej večja v C#.	V C++ je na voljo velika izbira numeričnih knjižnic, veliko od tega v prosti domeni. Po drugi strani je standardizirana baza jezika precej manjša.
K. V C# trenutno nimamo delujoče kode. Za implementacijo do nivoja Koščeve kode bi morala Katarina vložiti precej časa, to predstavlja tudi psihološko bariero. Če bi lahko z Robertom implementirala začetno ogrodje, kot je bilo načrtovano, bi to precej zmanjšalo razliko.	V C++ ima Katarina s Koščevo kodo delujočo in preizkušeno kodo. Implementiran je del fizike, ki jo bo rabila pri svojih primerih. To omogoča lažji začetek. Vseeno bo morala veliko stvari dopolniti in prirediti. Tu so lahko problemi, če koda ni prilagojena temu.
Prednosti, če Katarina začne razvoj v C#	
K, S. Hitrejše delo. C# ima zelo dodelano, čisto in jasno sintakso.	Kodiranje v C++ je malo manj pregledno in zamudnejše, čeprav je v osnovi sintaksa podobna C#. Katarina ima nekaj izkušen s C++, vendar bi prednosti zaradi tega izzvenele v nekaj tednih.
S. Programiranje v C# je manj zahtevno.	K zahtevnosti programiranja v C++ veliko prispeva potreba po upravljanju z dinamičnim spominom.
K, S. Manj razhroščevanja in zahrbtnih napak. Prevajalnik C# je zelo dodelan v tej smeri, da zmanjša možnost logičnih napak. “Varni način” (implicitno upravljanje z dinamičnim spominom, bound checks…) omogoča hitro lokacijo vzrokov napak.	Verjetnost logičnih napak je večja. V C++ je veliko možnosti za napake, ki jih je težko odkriti (memory leaks, nepooblaščeno pisanje v spominu).
K, S. Sintaksa C# napeljuje k bolj objektnemu pristopu in zato boljši strukturi kode.	Ker je neobjektni C podmnožica C++, programerji pri prehodu lažje nadaljujejo s slabimi praksami in rabijo več vodenja, da jih opustijo. Logična dolgoročna posledica je

	več ukvarjanja s kodo in manj časa za bolj pomembne stvarmi (fizikalno ozadje, numerični postopki, zahteve industrije).
S. Enostaven prenos programov v C# na druge platforme. Vzameš prevedeni program in dll-e ter ga poženeš.	Prenos spremenjene na drugo platformo pomeni ponovno prevajanje.
K. Solidna in trajna podpora z moje strani.	V C++ bi težko zagotovil primerljivo podporo.
K, S. Baziranje kode na osnovni knjižnici, ki je bila načrtovana predvsem za podporo hitrega razvoja numeričnih aplikacij za industrijske namene in za hkraten razvoj in testiranje osnovnih algoritmov. Aktivni razvoj zaradi optimizacije bo v precejšnji meri uporaben v simulacijski kodi.	
S. Če bi Umut uspel v krajšem času končati svoje stvari, bi bilo s kodo v C# možno poenotiti njegove stvari s Katarinino kodo.	
S. Pri kodi v C# bi bilo lažje pozneje priključevati ljudi na isto kodo, predvsem zato, ker je jezik lažji in bolj objekten in ker bi bila kontrola nad razvojem kode boljša.	Predvsem za ljudi, ki nimajo veliko izkušenj z objektnim programiranjem ali delom na obsežnejših projektih, je prehod na kodo v C++ težji kot v primeru C#.
S. Lažja integracija z industrijskim okoljem zaradi širokega nabora standardiziranih osnovnih knjižnic (npr. distribuirane aplikacije as komunikacijo preko TCP/IP, podatkovni in web standardi…).	Pri C++ je za veliko od teh osnovnih pripomočkov iz IT potrebno poiskati ustrezne dodatne knjižnice.

Dodano 14. maja 2011:

Od izdelave zgornje tabele sem prišel še do nekaj dodatnih zaključkov, ki jih navajam spodaj.

Neposredna prenosljivost osnovne kode v C# (brez grafike) bo velika prednost, če bomo stvari poganjali na drugih platformah (npr. superračunalnikih z na Unix-u temelječim OS). Pri kodi v C++ je prenosljivost še vedno precejšen problem, o tem sem se pogovarjal tudi s Koscem, ki je to potrdil.

V primeru, da Katarina dela v C#, bo lahko prišlo do večje sinergije med delom Katarine in Tadeja, ki dela v C#. Tadej je po začetnih težavah pri programiranju (zaradi slabih osnov, s katerimi je prišel v skupino) pokazal zavzetost in odgovornost pri delu in po mojem lahko računam, da bo sčasoma prišel na nivo, ko bo lahko samostojno produciral različna orodja za numerične kode. Ker naj bi Tadej delal tudi na branju in zapisu ter predstavitvi podatkov iz simulacijskih kod, bi bila uskladitev med njegovim in Katarininim delom koristna. Glede na to, da na projektu nimamo ravno močne ekipe (prej bi ocenil, da smo kadrovsko podhranjeni za dober zagon podpore proizvodnji fulerenov), bi bilo treba razvoj čimbolj homogenizirati. Tako bi Tadej tudi lažje prevzel del bremena pri implementaciji od Katarine ali obratno, če se pokaže potreba (pri majhnih skupinah je takšna prožnost zelo pomembna). Ena ideja, ki se mi poraja, je na primer, da bi Tadej ob moji pomoči prevzel implementacijo modula za kinetiko kemijskih reakcij in bi se lahko Katarina na ta račun takoj lotila kontinuumskega modela. Drug pomemben primer (to bi prišlo v poštev bolj dolgoročno, računam, da v letu do letu in pol) je uporaba zunanjih orodij za izgradnjo mreže pri adaptivnem mreženju (remeshing), npr. katerega od številnih orodij, ki so zelo dodelana v

skupnosti, ki se ukvarja s končnimi elementi (npr. GID ali drugih orodij dostopnih na trgu ali v obliki odprte kode). Verjetno bo Tadej po naravi svojega dela najbljižje implementaciji potrebnih vmesnikov in če bi Katarina delala v C#, bi bila uporaba njegovegih orodij zanjo precej enostavnejša. Kar se tiče sklopitve z optimizacijo, je stvar manj kritična, ker je na tem področju navadno lažje zadostiti zahtevam s šibkejšo sklopitvijo kod (tako ostane samo problem težjega razdeljevanja nalog med več članov skupine).

Pri C++ je vprašanje, kako bomo lahko skrbeli za zagotavljanje kvalitete kode (QA). S tem razen mene nihče v skupini nima veliko izkučenj. V primeru razvoja v C# (npr. nevronske mreže) mi je v veliko pomoč baziranje na trdni strukturi knjižnic. Standardna urejenost kodnih projektov, ki jo tu lahko zagotavljam, mi omogoča razmeroma dober pregled nad kodo. Pri razvoju dela kode v C++ bi bilo to veliko težje, v skupini se bodo podvojili potrebni komunikacijski napori za usklajevanje razvoja kode.

Kar se tiče prednosti, ki izhajajo iz tega, da bi Katarina izhajala iz Koščeve kode (ki za dani obseg problemov že deluje in je preizkušena), se mi zdi, da so te manj enoznačne, kot je bilo videti.

Količina čiste kode (.cpp + .h), ki jo je Katarina dobila od Kosca, je okrog 80 kB. Trenutna količina čiste kode v projektu za nevronske mreže (.cs) je okrog 700 kB, pri čistem jedru (knjižnice) pa 400 kB. K temu je treba dodati, de je bilo samo jedro knjižnice IGLib, preden sem prišel v COBIK, veliko čez 3 MB (čiste kode), v izoliranem razvojnem peskovniku je še enkrat toliko kode in veliko je še kode izven jedrnega dela knjižnice (velik del tega sem zgradil v svojem prostem času, kot osebno investicijo za lažji razvoj svojih aplikacij). Ta koda je sistematično načrtno grajena na podlagi petnajst letnih izkušenj pri razvoju zahtevnih aplikacij (predvsem numeričnih). Katarina bi gotovo nekaj profitirala s tem, ko bi izhajala iz preizkušene in delujoče Koščeve kode, in bi ji to olajšalo predvsem začetek implementacije. Dolgoročno pa bo po mojem mnenju večjega pomena lažje usklajevanje in integracija dela s tem, kar bova razvijala s Tadejem. Samostojna implemntacija (delno morda kar prepis kode) bi imel tudi koristne učineke, ker bi se Katarina pri tem bolje utrdila v osnovah programiranja takšnega modela, nad njim bi imela boljši pregled in bi ga zelo dobro obvladala, lažje povezovala z drugimi stvarmi in podobno.

Pri diskusijah o izbiri platforme je bilo veloko govora o razpoložljivosti knjižnic, kar smo omenjali kot eno od glavnih slabosti C#. Kot zunanja knjižnica je bil v Koščevi kodi uporabljen LAPACK (Linear Algebra Package), ki pa obstaja tudi v C# in ga je Robert uporabil v svojem primeru za NAFEMS. V IGLib je vključenih kar nekaj odprtokodnih knjižnic, ki so skrbno pozicionirane, dodelan je concept vključevanja. Dodatne knjižnice so vključene tudi v projektu za nevronske mreže.

V celoti se je moja ocena, da bi bilo dolgoročno precej bolje, če bi Katarina razvijala numerični model proizvodne celice v C# in uporabi Koščevo kodo delno kot model kot model (šablono), kot referenčno kodo za preverjanje pravilnosti in kot vir algoritmov. To bi zanjo pomenilo nekoliko težji začetek, vendar bi bilo to uravnovešeno s tem, da bi se pri lastni implementaciji osnovne simulacijske kode veliko naučila in bi izdelano kodo zelo podrobno poznala. Že v roku dveh let bi se morale jasno pokazati prednosti bolj homogenga razvoja v skupini. Te bodo verjetno postale še bolj očitne, če bomo ustanovili spin-off podjetje, ki bon a trgu ponujalo storitve podpore proizvodnji fulerenov z numeričnimi modeli.

1 I. GREŠOVNIK: ZADOLŽITVE IN KOMPETENCE DO 2013

Opomba: Vsebina tega poglavja je zastarela, posodobljena vsebina je v posebnem dokumentu.

1.1 COBIK

Definicija in razvoj platforme za optimizacijo celice za proizvodnjo fulerenov

- v C#
- temelji na knjižnicah IGLib
- splošna platforma, omogoča delo z različnimi simulacijskimi softveri in z različnimi optimizacijskimi algoritmi / okolji, vgrajena splošna orodja za (šibko) integracijo, poudarek na modularnosti in naprednih orodjih za praktične probleme

Sklopitev z numeričnim modelom - Katarina
Sklopitev z modelom na podlagi nevronskih mrež – Tadej Kodelja
Optimizacijska merila
Optimizacijski postopki
Grafična zasnova

1.1.1 Opombe

Šibka integracija pomeni sklopitev, pri kateri niso potrebni večji posegi v simulacijsko kodo. Parametrizacijski moduli so zelo ad hoc in brez posebnih konceptov (npr. z vnaprej definiranimi in zakodiranimi možnimi tipi odzivnih funkcij).

1.1.2 Okvirni časovni načrt za optimizacijsko platformo

Do konca 2011:

- Testno okolje za testiranje konceptov
- Optimizacijski proces izveden v testnem okolju
  - Klic zunanjega algoritma
- Nekaj osnovnih algoritmov
- Osnovna pomožna orodja (ugotavljanje lastnosti odziva, testiranje rešitev…)
- Sklopitev z modelom z nevronsko mrežo

Do konca 2012:

- Izboljšave algoritmov za industrijske primere
- Dodajanje orodij, npr. za paralelizacijo
- Povezava z zunanjim optimizacijskim softverom (IJS)
- Povezave z industrijskimi modeli
- Izvedba industrijske optimizacije

Do konca 2013:

- Dodatni algoritmi in njihove izboljšave glede na izkušnje in zahteve pri praktičnih primerih
- Nova orodja
- Pridobivanje projektov za nadaljnje delo
    - Trženje preko konferenc, člankov, obiskov, infrastrukture COBIKa
    - Splošno dostopne demo verzije
    - Dogovarjanje z industrijskimi partnerji
    - Projekti iz javnih sredstev
- Reševanje aktualnih industrijskih primerov

1.2 UNG – projekt Štore

Definicija in razvoj platforme za optimizacijo procesne verige v Štore Steel

- Platforma bo ista, kot se razvija za COBIK

Definicija vhodnih in izhodnih podatkov (= definicija optimizacijskih problemov)

- V sodelovanju z ljudmi iz industrije
- Skrb za izvedljivost (kar se tiče softvera in matematično) – potrebno sodelovanje in disciplina vseh vpletenih

Definicija grafike (kakšne predstavitvene zmogljivosti naj imajo numerični softveri)
Definicija vmesnikov med numeričnimi modeli (output enega -> input drugega), koordinacija za izvedbo
Sklopitev z evolucijskimi optimizacijskimi postopki
Sklopitev z numeričnimi modeli
Razvoj in zapis (definicija) optimizacijskih meril

1.3 Laboratorij – skupni razvoj (COBIK + UNG)

Vodenje in koordinacija timskega dela v laboratoriju pri programiranju
Pomoč pri vzpostavitvi skupnih knjižnic (svetovanje, planiranje, razporejanje nalog)

Resources:
Katarina Mramor 8 ur/teden
Gregor Košak 8 ur/teden
Quingguo Liu 8 ur/teden
Tadej Kodelja 20 ur/teden

1.3.1 Razvojni cilji in področja

C# okolje in grafika ter manipulacije za numerične modele
- Implementacija: Katarina, G. Košak, Q. Liu, T Kodelja
- I/O
- Postprocesiranje
- GUI

KONTINUIRNO ULIVANJE JEKLA IN ALUMINIJEVIH ZLITIN, TLAČNO ULIVANJE; ITD.
- Implementacija: Robert 50%, Juh 25%, Vuićević 25%, Agnieszka 100%
- Ostaja pretežno v Fortranu v okviru projekta Štore
- Nadaljni modeli
- Implementacija:

CELICA ZA PROIZVODNJO FULERENOV
- Implementacija: Katarina, morda + Igor, morda +Robert
- C++ (Igor preferira C#, če bi bili izpolnjeni pogoji za to)

MAKORIZCEJANJE IN GIBANJE TRDNE FAZE
- Implementacija: G. Košak
- C++

VALJANJE
- Implementacija: Umut
- C#

C++ PLATFORMA
- Implementacija: G. Košak, pomaga Robert
- Splošna geometrija
- Tlačna korekcija
- Problemi toka v cevi

1.3.1.1 Opombe: timsko delo

Igorju G. ima pooblastila potrebna za vodenje timskega dela – načrtovanje in razporejanje konkretnih nalog, razporejanje skupnih nalog (npr. administracija strežnikov), usklajevanje z Božidarjem glede širšega konteksta (obremenjenost članov tima, timing, strateško načrtovanje).

Božidar in Robert bosta pomagala pri premoščanju problemov, ki bodo nastajali zaradi nehomogenosti tima in razpršenosti razvoja.

1.3.1.2 Opombe: Numerični modeli v C++

Če bosta G. Košak in Katarina začela delo z isto kodo v C++, se načeloma razume, da je to razvoj dveh ločeni vej softvera. Na ta način preprečimo, da bi se ovirala zaradi precej različne narave dela. Če in dokler bo to šlo brez težav, lahko razvijata znaten del kode skupaj. Kar se tiče knjižnic, se vseeno poskusi razvoj poenotiti. Da se izognemo težavam, se lahko knjižnice razdelijo v več nivojev in uporabljata skupaj samo nekatere.

1.3.1.3 Opombe: Definicije vhodnih in izhodnih formatov za numerične simulacije

Definirali se bodo enotni osnovni formati. V okviru tega ima lahko vsaka koda svoje specifike (glede na to, kateri podatki se uporabljajo v določeni kodi).

Osnovni format določi Igor G., ostali sodelujejo glede na interesna področja in povedo, katere podatke je potrebno zajeti.

Razvijalci kod za simulacije implementirajo format (rutine za I/O).

Format bo tekstoven, do neke mere modularen in razširljiv, omogoča gnezdenje, razmeroma enostaven za strojno branje in učinkovit, berljiv ljudem (pomembno za debagiranje!), vsaj malo standarden videz. Sintaksa: zaviti oklepaji za grupiranje (“list-based”), zelo enostaven markup (npr. ključne beside, ki se začnejo z $). Koliko v globino bomo definirali stvari (npr. ali vsebuje tudi vrsto baznih funkcij) se bomo odločili sproti glede na količino težav pri implementaciji. Če bo vse OK, lahko vključimo linke na dele podatkov (predvsem eno ali večdimenzionalne tabele) v ločenih binarnih datotekah.

Nivo dodelave formatov in poprocesorskih orodij prilagajamo razpoložljivosti virov.

Reference:

[1] Igor Grešovnik: Coordination of software development in COBIK and Laboratory for Multiphase Processes. Treatiese, COBIK, 2011.

[2] Igor Grešovnik: *Programmers' guidelines for Development of Software within COBIK & Laboratory for Multiphase Processes*. Treatise, COBIK, 2012.

[3] Igor Grešovnik: *IoptLib*, electronic document at http://www2.arnes.si/~ljc3m2/igor/ioptlib/.

[4] Igor Grešovnik: *IGLib.NET*, electronic document at http://www2.arnes.si/~ljc3m2/igor/iglib/index.html.

[5] Igor Grešovnik: *IGLib Documentation*, electronic document at http://dl.dropbox.com/u/12702901/code_documentation/generated/iglib/index.html .

[6] Igor Grešovnik: *IGLib.NET Code Documentation*, electronic document at http://dl.dropbox.com/u/12702901/code_documentation/generated/iglib/html/index.html .