Posudek oponenta diplomové práce

Práce: Jazyk pro programování asynchronních serverů
Autor: Miroslav Lenčéš
Vedoucí: Leo Galamboš
Oponent: Petr Tůma

Práce navrhuje jednoduchý komponentový systém. Komponenty jsou představované Java třídami, které dědí z dodané třídy AbstractNode a pomocí anotací uvádí své rozmístění na síti a propojení s dalšími komponentami. Konektory jsou spojení schopná přenášet instance slovníkového typu Datagram, které je možné případně kopírovat více příjemcům a spojovat od více odesílatelů.

Návrh práce není v jasném souladu ani se zadáním práce, ani s uvedenou motivací. Zadání požaduje návrh programovacího jazyka, který řeší zejména problémy s předáváním dat při asynchronní obsluze požadavků. Motivace uvádí potřebu přizpůsobování aplikací, zejména změnu toku dat a změnu formátu dat, bez úprav zdrojového kódu. Navržené cíle pak jen obecně hovoří o vytváření prostředí pro psaní multiplatformních distribuovaných aplikací. Závěr práce, který konstatuje splnění navržených cílů, je pak nutné brát s rezervou, protože prostředí neřeší zásadní problémy jako zajištění bezpečnosti nebo rozkládání zátěže a multiplatformnost se omezila na platformu Java.

Text práce, ačkoliv celkově čitelný, trpí místy nepřesným vyjadřováním. Například o situaci, kdy chyba v topologii aplikace způsobí zablokování, se hovoří jako o nepříjemném zpomalování aplikace. Hlavním nedostatkem textu je ale chybějící analýza toho, co je vlastně cílem práce ve vztahu k zadání a motivaci. Práce se omezuje na povrchní rozbor témat jako je paralelismus nebo distribuce (toto je obzvláště patrné například v kapitolách 2.3.2 či 2.3.3), po nichž následuje bez potřebného zdůvodnění přímo popis implementace na úrovni jednotlivých tříd (kapitola 3). Práce tak nevysvětluje některé zásadní otázky, zejména:

- Proč byla v rozporu se zadáním opuštěna myšlenka programovacího jazyka ?
- Proč byl přidán požadavek multiplatformní distribuce, který zadání nezmiňuje ?
- Vyhovuje data flow model acyklického grafu aplikacím, pro které je návrh určen ?
  (Mimochodem nazývat tento model stavovým automatem je docela zavádějící.)
- Proč je použitý formát datagramů vhodný, řeší některé otázky popsané v motivaci ?
- Dává mechanismus spojování datagramů smysl, vyhovoval by i ve složitých grafech ?

Po technické stránce je implementace malého až středního rozsahu (3500 LOC), relativně čitelná a přiměřeně funkční, sama o sobě by představovala rozumnou součást práce. Výtky, které mám k implementaci, jsou spíše méně vážné:

- Dává smysl svazovat jméno uzlu se jménem třídy, která ho implementuje ?
- Není pravda, že Java nemá volání select, k dispozici je třída java.nio.channels.Selector.
- Jak by zotavovací mechanismus fungoval v případě současného výpadku několika uzlů ?
  (A co se stane s požadavky, které uzel při výpadku právě obsluhoval ?)

Celkově práce nabízí zajímavý prototyp prostředí, o kterém však není jasné, pro jaký typ aplikací se vlastně hodí, a který se nijak očividně nehodí ani k zadání práce, ani k uvedené motivaci. V textu práce výrazně chybí hlubší analýza (která by pravděpodobně dále zdůraznila zmíněný rozpor mezi požadavky a výsledkem). Domnívám se, že práce by v této situaci neměla být úspěšně obhájena.

V Praze 15. května 2009.