VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

Ing. Martin Vítek

DISTRIBUOVANÉ SYSTÉMY NA PLATFORMĚ .NET FRAMEWORK

DISTRIBUTED SYSTEMS ON THE .NET FRAMEWORK PLATFORM

Studijní program:	**Elektrotechnika, elektronika, komunikační a řídicí technika**
Studijní obor:	**Teleinformatika**
Školitel:	**Ing. Ivo Herman, CSc.**

Klíčová slova:

distribuované systémy, .NET Framework, sdílení a alokace prostředků, asynchronní rozhraní, generování IL kódu, aspektově orientované programování

Key words:

distributed systems, .NET Framework, resource sharing and allocation, asynchronous interface, runtime IL code generation, aspect-oriented programming

OBSAH

1 ÚVOD

Tato práce se zabývá problematikou přístupu ke sdíleným prostředkům distribuovaných systémů na platformě .NET Framework, což je platforma vyvinutá firmou Microsoft usnadňující vývoj aplikací v distribuovaném prostředí internetu a intranetu. Vzhledem k tomu, že se jedná o poměrně „mladou“ platformu (oficiálně vydanou v roce 2002), je zde stále otevřen prostor pro nové přístupy vývoje aplikací, jako je např. vývoj servisně orientovaných aplikací (SOA). To platí samozřejmě i pro oblast distribuovaných aplikací, kdy .NET platforma přichází jak s osvědčenými, tak i s novými přístupy a technologiemi urychlující vývoj a nasazení aplikací v distribuovaném prostředí.

Platforma .NET obsahuje řadu technologií využitelných pro implementaci určité oblasti distribuovaných systémů, jako je např. oblast síťové komunikace, zabezpečení, přístupu k datovým zdrojům a další. Distribuovaný systém obvykle obsahuje prostředky, které je nutno sdílet mezi několika procesy a toto sdílení může probíhat současně. V rámci aplikace je pak nutno řešit přístup ke sdíleným prostředkům. Tento princip není na platformě .NET zatím koncepčně řešen, kdy využívání nízkoúrovňových synchronizačních mechanismů není jednoduchou záležitostí a při nevhodném používání může vést až ke stavu uváznutí při běhu systému. Řešením může být vyžití některých doporučovaných programových modelů, avšak jejich nevýhodou je nutnost daný programový model implementovat, což obvykle značí narušení přístupu objektově orientovaného programování (OOP) při vývoji systému, jako je např. zprávově orientovaná komunikace.

Teoretická část této práce přichází s vlastním návrhem modelu pro popis sdílených prostředků a vzájemných vazeb, kdy je tento model postaven na OOP principech a současně přináší základní algoritmus pro zamezení stavu uváznutí.

Protože logika sdílení prostředků nesouvisí přímo s vlastní logikou služby, kterou prostředek poskytuje, nabízí se zde možnost využití techniky aspektově orientovaného programování (AOP), která umožňuje přesně oddělit jednotlivé aspekty systému a zautomatizovat jejich vzájemnou interakci.

Přístupu AOP je využito v rámci implementační fáze práce, kdy je provedena implementace navrhovaného modelu prostřednictvím techniky AOP tak, aby se s prostředky definované prostřednictvím modelu pracovalo stejně, jako se standardními objekty.

2 CÍLE DISERTAČNÍ PRÁCE

Předložená dizertační práce se zabývá problematikou přístupu ke sdíleným prostředkům v rámci distribuovaných systémů. Na základě studia problematiky distribuovaných systémů na platformě .NET, platformy obecně, praktických zkušeností získaných při realizaci zprávově orientované middleware vrstvy a vývoje multivláknové systému v rámci realizace komunikačního kontroléru radiové sítě byly postupně stanoveny následující cíle dizertační práce:

- Vydefinovat návrhové principy pro vývoj multivláknových aplikací s cílem zamezení vzniku stavu uváznutí a na základě těchto principů vytvořit univerzální rozhraní pro tvorbu multivláknových komponent.
- Navrhnout a matematicky popsat model pro přístup ke sdíleným prostředkům, který by umožňoval definičním způsobem zajistit bezpečnou práci se sdílenými prostředky bez nutnosti využívat primitivních synchronizačních technik obvykle vedoucí ke vzniku stavu uváznutí. Navrhnutý model by měl vycházet z objektově orientovaného přístupu.
- Implementovat navržený model tak, aby byl co možná nejvíce transparentní v takovém smyslu, že použití implementovaného modelu by znamenalo co možná nejmenší zásah do standardního objektově orientovaného programového modelu platformy a přispělo by k usnadnění vývoje sdílených prostředků. Implementace by měla být otevřená s možností jejího rozšíření a využívat v maximální míře samopopisných vlastností platformy .NET. Použití navrhované metody by nemělo zamezovat v použití standardních synchronizačních technik.
- Přirozeně konečným cílem je ověření správné funkčnosti navrhovaného modelu prostřednictvím implementace vybraných praktických situací obsahující sdílení prostředků postavených na navrhovaném modelu a srovnání s implementací pomocí standardních technik.

3 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Při tvorbě komplexních informačních systémů, kam distribuované systémy bezesporu patří, si dnes již nelze vystačit jen s popisem algoritmů a datových struktur, ale je nutné zabývat se strukturou systému na vyšší úrovni abstrakce [67]. Vyšší úroveň abstrakce přináší do vývoje systému dva základní prvky:

- Snížení počtu problémových oblastí, které je třeba řešit, neboť díky abstrakci odpadá rozhodnutí, a návrh detailních částí systému, což v konečném důsledku sníží složitost návrhu systému, neboť návrh je funkčně závislý na počtu idejí, které musí být řešeny v rámci návrhu systému [68].
- Zpřehlednění kódu, kdy kód obsahuje pouze vlastní logiku, kterou má systém provádět.

Platforma .NET již sama osobě poskytuje určitou úroveň abstrakce pro vývoj aplikací, která spočívá ve správě paměti, OOP přístupu. Další úrovni abstrakce lze docílit využitím technologií WCF [14], WWF [46] nebo LINQ [47], a tak se přiblížit ideálnímu stavu zmíněného v předcházejících bodech. Technologie pro přístup ke sdíleným zdrojům na vhodné úrovni abstrakce není na platformě .NET zatím přítomna.

3.1 PŘÍSTUP KE SDÍLENÝM ZDROJŮM

Přístup ke sdíleným zdrojům může být řešen programově na různých úrovních systému. Pokud tedy nelze zajistit pro každý požadavek vlastní dedikovaný prostředek, je potřeba tyto prostředky sdílet mezi jednotlivými procesy nebo vlákny zpracovávající daný požadavek. Přístup ke sdíleným prostředkům lze řešit různými způsoby.

3.1.1 Vzájemné vyloučení

Nejčastěji používanou technikou řešení přístupu ke sdíleným prostředkům je technika vzájemného vyloučení (*mutual exclusion*). Tato technika obvykle vyžívá speciálních klíčových slov k vymezení tzv. kritického úseku kódu, ve kterém je zaručeno, že tento úsek bude zpracovávat jen jedno vlákno současně.

Využívání kritických sekcí sebou nese řadu úskalí, kterou je potřeba mít na paměti a vyvarovat se jich. O některých z nich, konkrétně pro jazyk JAVA, pojednává článek [50].

Jedním z nejméně příjemných důsledků nesprávného použití této techniky je vznik stavu uváznutí, tj. stavu, kdy dojde ke křížovému čekání na uvolnění kritické sekce. K uváznutí dojde, pokud jsou splněny následující podmínky uvedené v [51].

Není-li alespoň jedna ze zmíněných podmínek splněna v žádném okamžiku běhu programu, nedojde ke stavu uváznutí. Možné techniky jsou:

- Každý proces si zažádá o všechny potřebné zdroje, a dokud je nedostane, tak bude pozastaven.
- Pokud má proces alokovaný nějaký prostředek, není mu dovoleno alokovat jiný, dokud původní prostředek neuvolní.
- Seřazení prostředků a povolení procesu alokovat pouze prostředky, které jsou následnými prostředky prostředků již alokovaných daným procesem.

3.1.2 Instrukce a API rozhraní

Další technika pro paralelní přístup k prostředkům, obecně řečeno k datům v paměti, je postavená na speciálních instrukcích nebo API rozhraní realizující přístup k paměti procesu.

Jedná se o techniky MCAS (Multi-word Compare-swap), WSTM (Word-based software transactional memory) a OSTM (Object-based software transactional memory) [52]. Tyto techniky jsou založeny na využívání speciálního aplikačního rozhraní API a v případě MCAS a WSTM pracují až na úrovni paměti. Princip těchto technik využívající atomické instrukce pro práci s paměti popisuje článek [76], kde je popsána implementace LIFO a FIFO zásobníků bez nutnosti využití instrukce pro uzamykání kritické sekce.

Vzhledem k tomu, že tyto techniky pracují až na úrovni paměti, není jejich využití v rámci OOP příliš výhodné, obzvláště v případech moderních OOP prostředích, kde řízení paměti provádí běhové prostředí, jako je platforma .NET nebo JAVA.

3.1.3 Jazykové rozšíření a podpora

Komfortní technikou pro řešení paralelního přístupu ke sdíleným prostředkům je využití některého z programovacích jazyků, který tuto podporu již v sobě obsahuje jako např. Erlang, Ada nebo jazyka se syntaktickým rozšířením stávajících jazyků (Split-C, Cilk) [58].

Tyto jazyky umožňují velmi pohodlně řídit a spravovat paralelně pracující kód, ale na druhou stranu nepatří mezi rozšířené techniky a tím pádem nedisponují takovými integračními schopnostmi s ostatními technologiemi jako standardní OOP programovací jazyky.

3.1.4 Programové modely

Programové modely definují postupy a pravidla pro sdílení prostředků. Mezi implementace těchto modelů patří např. Reo, Esterel, SIGNAL nebo Lustre [58].

Programové modely se obvykle používají ve fázi návrhu systému a pro vlastní systém je potřeba pak následně programový model nebo jeho část implementovat, což nemusí být vždy snadné, např. z důvodu, že model využívá specifických programových technik.

3.1.5 Formální verifikace systémů

Jiným přístupem při návrhu a implementaci systémů (obzvláště složitějších, kam multivláknové systémy patří) je jejich ověření prostřednictvím formální analýzy a verifikace, což jsou metody, které v poslední době patří mezi rychle se rozšiřující techniky ověřování správnosti systémů, kam samozřejmě patří i ověření, že se v systému nevyskytuje stav uváznutí [67]. Formální verifikace založená např. na formální specifikaci systému prostřednictvím temporální logiky se používá především tam, kde nelze použít mechanismus prevence nebo detekce a následného zotavení se stavu uváznutí. Jedná se tedy především o ověření korektnosti systému prostřednictvím jeho modelu nebo i systému samotného na základě analýzy, avšak obvykle ještě před jeho zprovozněním v produkčním prostředí.

Protože součástí této práce je návrh koncepce, která má usnadnit vývoj systému a následně se uplatnit při jeho činnosti, není problematika formální verifikace v této práci dále studována.

3.2 VYUŽITÍ AOP PŘÍSTUPU

Obecnou technikou, jak zautomatizovat a zjednodušit vývoj distribuovaného systému, je použití AOP (Aspect Oriented Programming) přístupu. Tento přístup umožňuje transparentně oddělit stále se opakující programátorské techniky související se samotným chodem distribuovaného systému od logiky služeb poskytovaných systémem koncovým klientům.

Zde se proto nabízí možnost využití této techniky pro realizaci bezpečného přístupu ke sdíleným zdrojům, kdy aspektem by byla logika zajišťující bezpečný přístup ke sdíleným prostředkům. Na tomto přístupu je postavena implementace navrhovaného vlastního modelu pro popis sdílených prostředků vycházejícího z OOP principů.

4 POSTUP ZPRACOVÁNÍ DISERTAČNÍ PRÁCE

Vlastní práce se skládá ze dvou částí, kdy první část se zabývá návrhem a realizací univerzálního rozhraní pro asynchronní operace, neboť distribuovaný systém tak představuje z pohledu sdíleného prostředku multivláknovou aplikaci, u které je potřeba zajistit korektní přístup vláken ke sdíleným částem systému.

Proto cílem první části práce bylo definovat obecně použitelný prostředek pro práci s asynchronními operacemi, tj. operacemi, jejichž vykonání probíhá paralelně neblokujícím způsobem vzhledem k procesu, který operaci vyvolal. Takové chování je velmi často vyžadováno při využívání služeb sdílených zdrojů, u kterých obvykle není žádoucí, aby pracovaly blokujícím způsobem. Častým scénářem je vznik požadavku na službu sdíleného zdroje, kdy o výsledku požadavku je pak volající proces informován, což bylo vyžadováno i v rámci implementace modelu pro přístup ke sdíleným prostředkům, kde operace představovala požadavek na alokaci prostředku.

Návrh univerzálního rozhraní vycházel z důkladné analýzy a vlastních praktických zkušeností získaných v rámci řešení výzkumného projektu GA102/03/1033, jehož součástí byla realizace plně paralelně pracujícího kontroléru radiové sítě obhospodařujícího až 300 aktivních mobilních jednotek. Na základě výsledků bylo vytvořeno doporučení pro implementaci multivláknových aplikací a byl proveden návrh rozhraní rozšiřující standardní asynchronní programový model platformy .NET.

Druhá část se věnuje přístupu ke sdíleným prostředkům distribuovaných aplikací tak, aby byla minimalizována možnost vzniku stavu uváznutí a současně tato navrhovaná koncepce byla transparentní při jejím využívání v rámci .NET aplikací. Teoretická část řešené problematiky obsahuje návrh modelu pro popis sdílených prostředků a jejich závislostí vycházející z principů objektově orientovaného programování. Při návrhu bylo postupováno tak, aby implementace modelu byla možná objektově orientovaných způsobem, tj. způsobem, který vychází ze základního principu platformy .NET.

Za účelem vetší názornosti a aplikovatelnosti navrženého modelu, který obsahuje jak statickou tak i dynamickou část, byla následně vytvořena i grafická interpretace modelu. S využitím grafické interpretace bylo dále ukázáno využití modelu pro řešení několik základních praktických situací pro přístup ke sdíleným prostředkům.

Posledním krokem v rámci řešení této práce byla implementace navrhnutého modelu. Tato implementace rovněž sloužila pro ověření správnosti modelu, protože součástí implementační části je implementace zmíněných praktických situací popsaných grafickou interpretací modelu.

Implementační fázi předcházela důkladná analýza zpracování řízeného kódu na platformě .NET Framework, hlavně co se týče tvorby kódu za běhu aplikace a zapouzdření komunikace do objektové podoby prostřednictvím proxy objektu.

Na základě této analýzy byla provedena implementace navrhovaného modelu tak, aby synchronizace přístupu ke sdíleným zdrojům probíhala transparentně na pozadí s využitím AOP přístupu.

5 ŘEŠENÍ A VÝSLEDKY DISERTAČNÍ PRÁCE

Tato kapitola ve stručnosti popisuje výsledky disertační práce. Zaměřuje se na vybrané kapitoly z návrhu a implementace modelu pro přístup ke sdíleným zdrojům. Jelikož navrhované univerzální rozhraní je součástí implementace modelu, není zde zvlášť uvedeno.

5.1 MODEL PRO PŘÍSTUP KE SDÍLENÝM ZDROJŮM

Navrhovaný model definuje prostředky, jejich rozhraní a vzájemné závislosti mezi rozhraními jednotlivých prostředků. Na základě vzájemných závislostí jsou pak vytvářeny synchronizační skupiny, které jsou v případě přístupu k prostředku alokovány.

5.1.1 Prostředek a rozhraní

Rozhraní je předpis, který definuje poskytované služby ve formě metody nebo vlastnosti. Prostředek představuje ucelenou jednotku poskytující svému okolí služby prostřednictvím svých rozhraní. Každý prostředek může obsahovat jedno nebo více rozhraní. Množinu rozhraní pro jeden *r*-tý typ prostředku je dána vztahem:

$$I(r) = \{i_l(r) | 0 < l \leq \mathrm{q}, i_l(r) \in I\}\,, \tag{1}$$

kde *q* je celkový počet rozhraní prostředku *r*.

5.1.2 Závislost rozhraní

Závislost mezi rozhraními definujeme následujícím symbolem:

$$\mathbf{b}_{r,l}^{\varrho,\lambda} \approx \mathrm{i}_l(r) \rightarrow \mathrm{i}_\lambda(\varrho)\,, \tag{2}$$

který říká, že rozhraní $i_l(r)$, tj. *l*-té rozhraní *r*-tého prostředku, využívá rozhraní $i_\lambda(\rho)$, tj. λ-té rozhraní ρ-tého prostředku. Množinu všech využívaných rozhraní rozhraním $i_l(r)$ definujeme jako:

$$B(i_l(r)) = \left\{\forall \mathrm{i}_\lambda(\varrho)\ \mathrm{z}\ \mathbf{b}_{r,l}^{\delta,\lambda} \middle| r, l = \mathrm{konst.}\right\}. \tag{3}$$

Mějme graf závislosti rozhraní *Q* definovaný předpisem:

$$Q = (I, B)\ , \tag{4}$$

kde *I* je množina všech rozhraní (od všech zdrojů) a *B* je množina všech závislostí mezi rozhraními obsažených v množině *I*.

5.1.3 Statická synchronizační skupina

Statickou synchronizační skupinu definujeme jako množinu spolu souvisejících rozhraní, tj. rozhraní mající mezi sebou přímou či nepřímou vazbu:

$$G = \{i | i \in I\}\ . \tag{5}$$

Jedná se v podstatě o nejjednodušší realizaci synchronizačních skupin, která nebere v potaz orientaci hran a posloupnost závislostí v grafu závislosti. Statickou synchronizační skupinu pro rozhraní *i* značíme *W*(*i*).

5.1.4 Subjekt

Prostředek představuje určitou definici určitého typu objektu mající definované rozhraní a vazby na ostatní rozhraní, ať už se jedná o závislá rozhraní nebo o rozhraní událostí. Konkrétní výskyt tohoto prostředku budeme nazývat subjektem. Subjekt je tedy instancí prostředku a budeme ho označovat písmenem *s*. Subjekt *s*(*r*) musí implementovat všechna rozhraní, které definuje prostředek *r*.

Pokud rozhraní *i* prostředku *r* obsahuje určité událostní rozhraní i_e, pak subjekt *s* umožňuje připojení libovolného počtu subjektů *s´* implementující i_e.

5.1.5 Dynamická synchronizační skupina

K událostnímu rozhraní subjektu můžeme připojit další subjekty, přičemž každý subjekt, resp. jeho rozhraní spadá do určité statické synchronizační skupiny. Statická synchronizační skupina určitého rozhraní *i* je tak rozšířena o statickou synchronizační skupinu subjektů připojených k událostním rozhraním definovaných pro rozhraní *i*. Tuto skupinu budeme nazývat dynamickou synchronizační skupinou a značíme ji *D*(*i*(*s*)), což značí dynamickou synchronizační skupinu rozhraní *i* subjektu *s*.

5.1.6 Synchronizační skupina subjektu

Kromě statické a dynamické synchronizační skupiny definujeme ještě synchronizační skupinu subjektu *p*, která může být svázána s rozhraním subjektu. Jedná se o synchronizační skupinu svázanou přímo s daným subjektem *s* a píšeme *p*(*i*(*s*)). Alokaci této synchronizační skupiny označujeme jako podmíněnou alokaci.

5.1.7 Garance dostupnosti prostředků

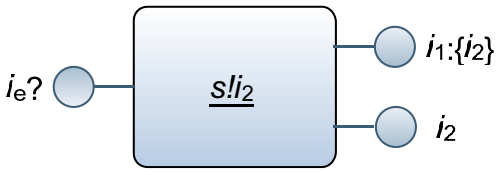
Prostřednictvím statických, dynamických a synchronizační skupiny subjektu lze zaručit dostupnost všech vyžadovaných prostředků při požadavku na určité rozhraní libovolného prostředku v systému. Tento požadavek velmi snadno splníme, pokud při požadavku na rozhraní $i(s)$ garantujeme exkluzivní přístup ke všem synchronizačním skupinám, tj.:

$$lock(i(s)) = lock(W(i) \cup D(i(s)) \cup p(i(s))) \,. \qquad (6)$$

5.2 GRAFICKÁ INTERPRATACE MODELU

Prostředek r implementující rozhraní i_1 budeme zobrazovat podle obr 1. Na tomtéž obrázku je znázorněna stejná situace, avšak pro subjekt s. Závislé rozhraní i_1 na rozhraní i_2 je zobrazeno na obr. 2. Současně je zde zobrazeno událostní rozhraní i_e a indikace, že subjekt implementuje podmíněnou alokaci pro rozhraní i_2 včetně definování podmínky pro splnění dostupnosti synchronizační skupiny subjektu.

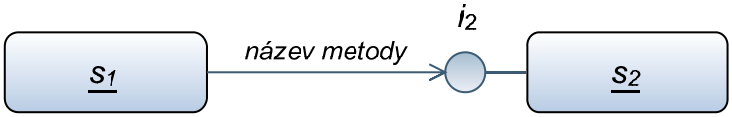
Obr. 1: Grafická interpretace prostředku *r* a subjektu *s* mající definované rozhraní i_1.



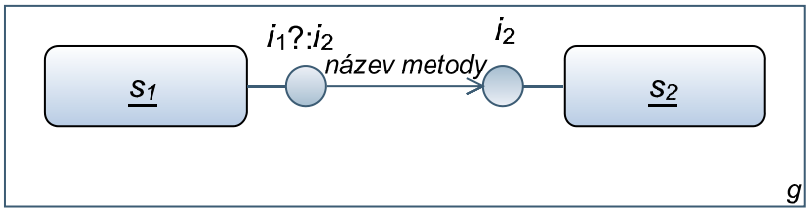
s!i_2: subjekt *s* je va stavu *ready*.

Obr. 2: Grafická interpretace závislosti rozhraní, událostního rozhraní a podmíněné alokace.

Případ, kdy subjekt s_1 volá metodu nebo vlastnost rozhraní i_2 subjektu s_2, je zobrazen na obr. 3. Vyvolání události je naznačeno na obr. 4, kde subjekt s_1 vyhazuje událost i_1, na kterou je napojeno rozhraní i_2 subjektu s_2. Na obr. 4 je současně zobrazena hranice regionu g. Název volané metody nebo vlastnosti rozhraní je uveden jako text u orientované čáry znázorňující volání.



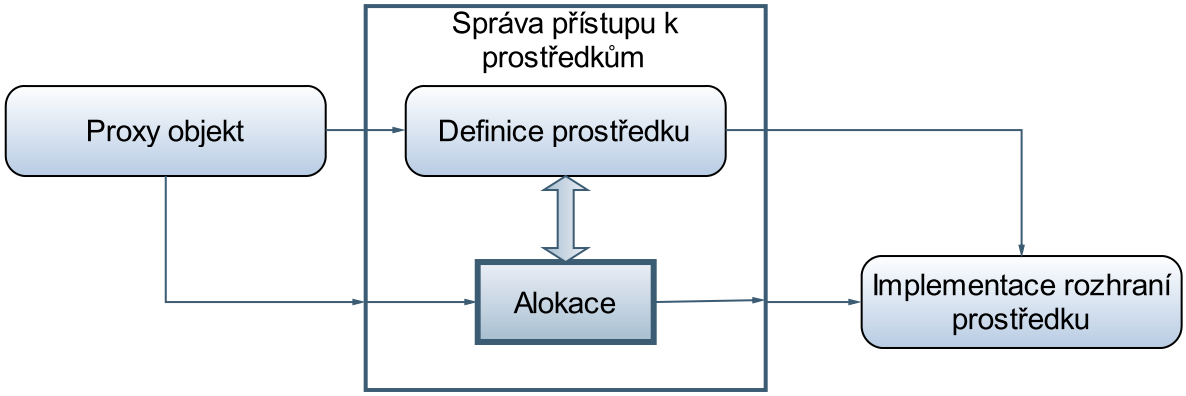
Obr. 3: Grafická interpretace volání metody nebo vlastnosti rozhraní subjektu.



Obr. 4: Grafická interpretace vyvolání událostního rozhraní.

5.3 IMPLEMENTACE MODELU

Využívání sdíleného prostředku probíhá výhradně prostřednictvím proxy objektu, který komunikuje s vlastním zdrojem přes vrstvu zajišťující správu přístupu ke sdílenému prostředku. Tato vrstva obsahuje kromě alokačního a synchronizačního mechanismu rovněž i definici prostředku obsahující popis, kde se nachází implementace konkrétního rozhraní prostředku (viz obr. 5).



Obr. 5: Princip přístupu ke sdíleným zdrojům s využitím proxy objektu.

Celkový pohled na jednotlivé vazby mezi jednotlivými „bloky“ implementující navrhovanou koncepci je zobrazen na obr. 6. Stěžejní elementy jsou zobrazeny tučně. Jedná se o objekt definující prostředek, proxy objekt zastupující implementaci rozhraní prostředku a vlastní implementace rozhraní prostředku ve formě standardního objektu. Význam vazeb u jednotlivých bloků je vyjádřen textem, čísla v rozích jednotlivých bloků odpovídají pořadí, jak jsou při přístupu ke sdílenému přístupu vytvářeny stěžejní bloky implementace. Slovně lze očíslované jednotlivé kroky popsat následovně:

1. Definuje se region.
2. Definuje se schéma virtuálních typů.
3. V rámci schématu virtuálních typu se definuje prostředek.
4. Definují se rozhraní prostředku.
5. Definuje se subjekt jako instance implementující prostředek.
6. V rámci subjektu je rozhraní prostředku přiřazen implementující objekt.

7. Při vyžádání rozhraní subjektu je vytvořen proxy objekt zastupující implementující objekt.
8. Při prvotním volání rozhraní přes proxy objekt je vytvořen kontext.
9. Pro alokaci synchronizačních skupin rozhraní prostředku je vytvořen alokační kontext.
10. Během vlastní alokace je vytvořen alokátor a provedena alokace synchronizačních skupin.

Pokud proběhne alokace v posledních kroku úspěšně, je volání nad proxy objektem přesměrováno na implementující objekt rozhraní, tj. je vykonána požadovaná služba.

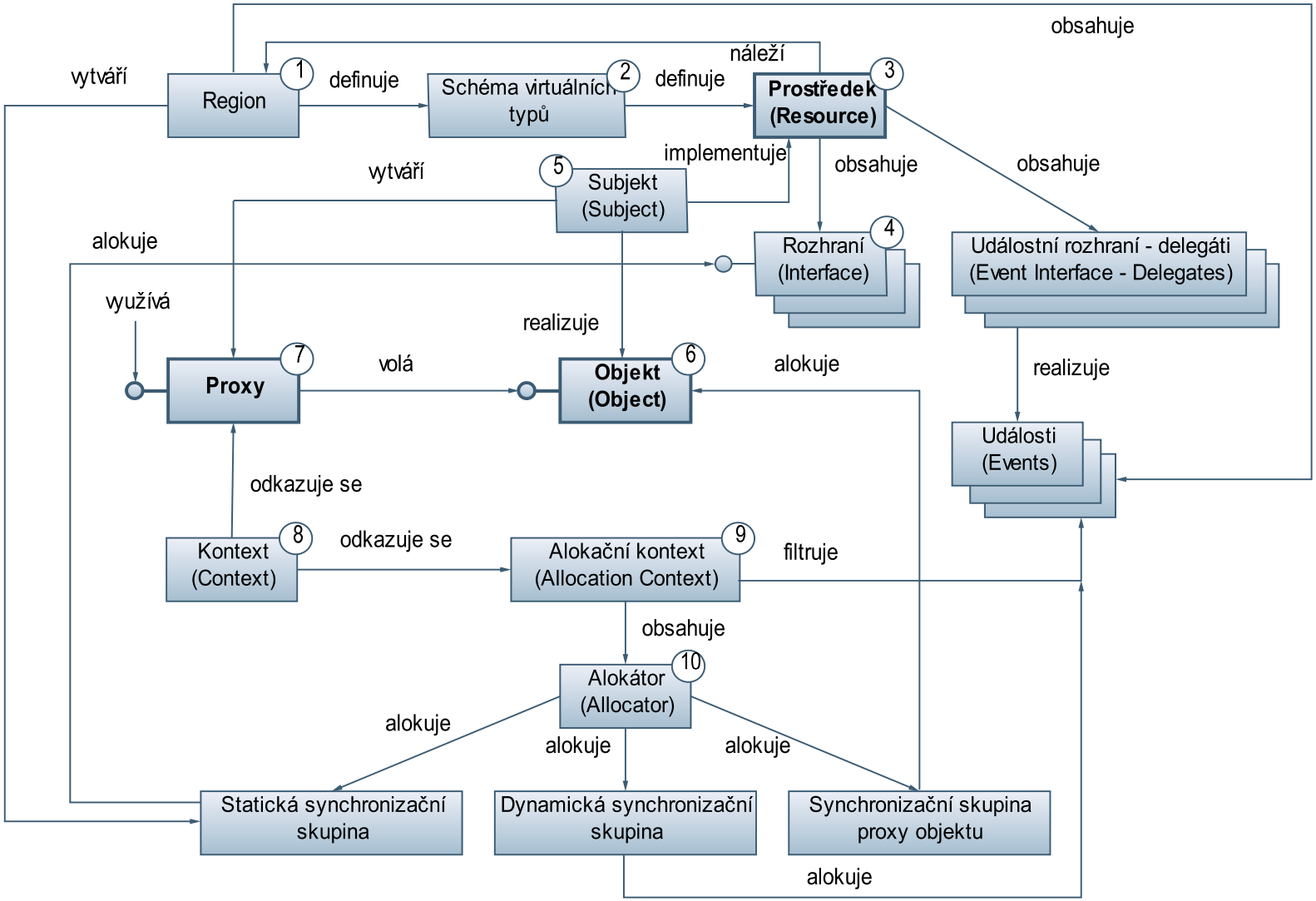
Implementace modelu obsahuje cca 50 tříd, kdy přetížením nebo vlastní implementací výchozích tříd a rozhraní lze ovlivňovat způsob chování alokace prostředků. Základní úrovně rozšiřitelnosti jsou:

- přetížení výchozí třídy proxy objektu – tímto způsobem lze plně kontrolovat průběh komunikace s prostředkem,
- implementace rozhraní alokátoru – lze implementovat vlastní alokační algoritmus pro jednotlivé synchronizační skupiny.

5.4 SROVNÁNÍ MODELU SE STANDARDNÍMI TECHNIKAMI

Z výsledků testů srovnávajících časovou náročnost přístupu ke sdílenému prostředku implementovaného prostřednictvím modelu a analogické implementace standardními technikami vyplynulo, že při libovolné lokální komunikaci navrhovaná koncepce vnáší do zpracování určité zpoždění způsobené alokací synchronizačních skupin. Toto zpoždění je v maximálním případě 1,6 násobkem oproti řešením standardními alokačními technikami.

V případě vzdálené komunikace uvedený závěr platí pouze, pokud nejsou událostní rozhraní využívaná za hranice procesu, kdy vlivem alokace synchronizačních skupin přes síťové rozhraní dochází až téměř k desetinásobnému zpoždění zpracování. Zde je třeba zmínit, že žádná ze standardních testovacích implementací neprováděla zamykání sdílených prostředků přes hranice procesů a také, že alokace synchronizačních skupin nebyla nijak optimalizována pro použití v rámci technologie .NET Remoting, která byla použita pro vzdálenou komunikaci v rámci testování.



Obr. 6: Blokové schéma implementace navrhovaného modelu pro přístup ke sdíleným zdrojům.

6 SHRNUTÍ VÝSLEDKŮ A PŘÍNOSY DISERTAČNÍ PRÁCE

Dizertační práce se zabývá oblastí týkající se vývoje distribuovaných systémů s využitím platformy .NET Framework se zaměřením se na problematiku přístupu ke sdíleným zdrojům a s tím související techniku zpracování asynchronních operací. Součástí se práce je rovněž implementační část realizující vlastní navržené techniky.

Pro účely asynchronního zpracování operací, byl proveden vlastní návrh univerzálního rozhraní rozšiřující standardní asynchronní programový model platformy .NET. Navrhované rozhraní bylo implementováno a úspěšně použito v rámci radiového kontroléru radiové sítě, což byla striktně vrstvově orientovaná implementace brány propojující radiovou síť s IP sítí. Zde bylo navrhované rozhraní použito pro popis asynchronních operací prováděných nad jednotlivými vrstvami. Díky použití tohoto rozhraní bylo docíleno úspěšné implementace multivláknového komunikačního systému umožňující realizaci komunikace na libovolném počtu radiových kanálů, který dodnes pracuje bezchybně 24 hodin denně. To dokazuje i správnost návrhu, který může sloužit i jako návrhový vzor pro implementaci objektů mající asynchronní operace. Navrhované rozhraní pak dále tvoří jednu ze základních částí v rámci implementace další řešené problematiky této práce.

Řešení problematiky sdílených zdrojů na platformě .NET je postaveno na vlastním modelu pro přístup ke sdíleným prostředkům vycházející z objektově orientovaných principů. Navrhovaný model je založen na přesném definování vazeb mezi jednotlivými prostředky prostřednictvím rozhraní. Na základě těchto vazeb jsou vytvořeny synchronizační skupiny, pomocí kterých se pak realizuje přiřazení prostředků procesům neboli alokace prostředků volajícímu vláknu. Návrh, který je postaven na aspektově orientovaném přístupu, byl koncipován tak, aby jednak umožňoval operace s prostředky známé z OOP programování, ale současně mohla být zajištěna plná kontrola nad přístupem ke sdílenému prostředku.

Pro navrhovaný model byla rovněž vytvořena grafická interpretace umožňující názorně aplikovat model na praktické situace.

Implementační část disertační práce vychází z analýzy zpracování kódu na platformě .NET a techniky pro zajištění transparence v rámci komunikace prostřednictvím .NET Remoting. Na základě této analýzy byl proveden návrh, jak realizovat alokaci prostředků transparentním způsobem, aby využívaní sdílených prostředků probíhalo stejně jako každá jiná práce s objektem na platformě .NET.

Implementace modelu současně ukazuje možnost, jak realizovat transparentní přístup k objektům postavený na principech aspektově orientovaného programování (AOP). V rámci implementace je současně ukázána dynamická tvorba kódu za běhu

aplikace a jeho okamžité využití, což je jedna z technik realizace AOP přístupu. Provedená implementace čítající cca 50 tříd tak ukazuje praktické využití AOP pro řešení komplexní techniky, jako je řízení přístupu ke sdílenému prostředku. Pro ověření správnosti nejen samotné implementace, ale především celého teoretického návrhu, jsem úspěšně vytvořil aplikační řešení několika základních scénářů vycházející z návrhu využívající grafickou interpretaci modelu.

Na časovém srovnání obdobných implementací postavených na navrhovaném modelu a s využitím standardních přístupů, které je součástí přílohy práce, lze vidět, že navrhovaná koncepce v rámci lokálního přístupu ke sdíleným prostředkům vnáší určité zpoždění způsobené logikou pro alokaci synchronizačních skupin. K razantnímu zpoždění provádění požadavku dojde, pokud dochází k alokaci synchronizačních skupin, které přesahují hranice procesu, kdy vlivem meziprocesové komunikace dochází k několikanásobnému zpoždění (při použití standardních přístupů nebylo implementováno zamykání prostředků za hranice procesů). Zde by byla zapotřebí implementace, která by plně a hlavně efektivně využívala použitou metodu meziprocesorové komunikace, což v testech nebylo učiněno, neboť šlo jen o ukázku, že funkčnost implementace není omezena jen na prostředky nacházející se v rámci jednoho procesu.

Navrhovaná koncepce obsahuje základní logiku pro zamezení stavu uváznutí, avšak při ne správném návrhu vazeb a z toho plynoucí využívání prostředků může dojít ke stavu uváznutí. Zde však je chování odlišné, neboť díky tomu, že alokace není prováděna nízkoúrovňovým přístupem, ale prostřednictvím alokačního mechanismu, neznamená stav uváznutí konec činnosti aplikace, ale dojde v rámci aplikace ke vzniku chybového stavu a v případě ošetření této chyby může aplikace pokračovat ve své činnosti.

Díky rozšiřitelné povaze implementace a formálnímu podkladu implementované knihovny modelu je možno tuto knihovnu nebo její části použít jako základní prostředí, na kterém lze prakticky realizovat různé techniky pro alokaci sdílených prostředků, nebo i jiných technik, které lze definovat ve formě aspektů ve smyslu AOP.

POUŽITÁ LITERATURA

6.1 SEZNAM POUŽITÉ LITERATURY

[1] Brookshier, D., Govoni, D., Krishnan, N., Soto, J.C., JXTA, Java P2P Programming, Sams, 2002, 432 stran, ISBN 0672323664.

[2] Bindal, R., Pei Cao, P., Can self-organizing P2P file distribution provide QoS guarantees?, *ACM SIGOPS Operating Systems Review*, 2006, ACM New York, s. 22-30, ISSN 0163-5980.

[3] Tenenbaun, A.S., Distributed Systems, Prentice Hall, 2002, 800 stran, ISBN 0-13-088893-1.

[4] Zelený, J., Nožička, J., COM+, CORBA, EJB, BEN, Praha, 2002, 310 stran, ISBN 80-7300-057-1.

[5] SDL forum, Introduction to SDL 88 [online], dostupné z: http://sdl-forum.org/sdl88tutorial/ [cit. 2009-30-3].

[6] Marques, J. M., Vilajosana, X., Daradoumis, T., Navarro L., LaCOLLA: Middleware for Self-Sufficient Online Collaboration, *IEEE Internet Computing*, April 2007, Vol. 11, Issue 2, s.56-64, ISSN 1089-7801.

[7] Glässer, U., Gurevich, Y., Veanes, M., Abstract Communication Model dor Distributed Systems, *IEEE Transactions on Software Engineering*, July 2004, Vol. 30, No. 7, s. 458-472, ISSN 0098-5589.

[8] Web Services Glossary, W3C Working Group Note 11 February 2004 [online], dostupné z: http://www.w3.org/TR/ws-gloss [cit. 2009-30-3].

[9] Carey, J. M., SOA What?, *IEEE Computer magazine*, March 2008, Vol. 41, Issue 3, s. 92-94, ISSN 0018-9162.

[10] Vogels, W., Web Services Are Not Distributed Objects, *IEEE Internet Computing*, November-December 2003, Vol. 7, Issue 6, s.59-66, ISSN 1089-7801.

[11] Lassila, O., Handler, J., Embracing „Web 3.0“, *IEEE Internet Computing*, June 2007, Vol. 11, Issue 3, s. 90-93, ISSN 1089-7801.

[12] Štumpf, J., Servisně orientovaná integrace, *Computerworld*, 2008, ročník XIX, č. 4, s.25-27, ISSN 1210-9924.

[13] Tidwell, D., Snell, J., Kulchenko P., Programming Web Services With SOAP, O'Reilly, 2001, 216 stran, ISBN 0-596-00095-2.

[14] Klien, S., Professional WCF Programming: .NET Development with the Windows Communication Foundation, John Wiley & Sons, 2007, 430 starn, ISBN 978-0-470-08984-2.

[15] Knap, P., Orchestrace a choreografie služeb, *SYSTÉMOVÁ INTEGRACE*, Česká společnost pro systémovou integraci, 2007, č. 4, s.117-126.

[16] Zervaas, Q., Practical Web 2.0 Applications with PHP, Apress, 2008, 594 stran, ISBN 978-1-59059-906-8.

[17] Mahmoud H.Q., Langendoerfer P., Service-Oriented Computing, *ACM Transactions on Internet technology*, May 2008, Vol. 8, No. 3, s. 11-13, ISSN 1533-5399.

[18] Buchler, M., Kwei-Jan Lin, Service-Oriented Computing, *IEEE Computer*, March 2006, vol. 39, no. 3, s. 99-101, ISSN 018-9162.

[19] Barnaby, T., Distributed .NET Programming in C#, APress, 2002,494 stran, ISBN 1590590392.

[20] Kačmář, D., Programujeme .NET Aplikace, Computer Press, Praha, 2001, 335 stran, ISBN 80-7226-569-5.

[21] Troelsen, A., *C# and the .NET Platform*, APress, 2001, 970 stran, ISBN 1893115593.

[22] Turner, M., Budgen, D., Brereton P., Turning Software into a Service, *IEEE Computer*, October 2003, Vol. 36, Issue 10, s. 38-44, ISSN 0018-9162.

[23] Ecma International, Standard ECMA-335, Common Language Infrastructure (CLI) [online], dostupné z: www.ecma-international.org/ publications/standards/Ecma-335.htm, [cit. 2009-30-3].

[24] Meyer, B., .NET Is Comming, IEEE Computer, August 2001, p.92-97, ISSN 018-9162.

[25] Lawton, G., Developing Software Online with Platform-as-a-Service Technology, *IEEE Computer*, p.13-15, June 2008, ISSN 018-9162.

[26] Tsai, W.T, Zhou, X., Chen, Y., Bai, X., On Testing and Evaluating Server-Oriented Software, *IEEE Computer*, August 2008, Vol. 41, No. 8, s.40-46, ISSN 018-9162.

[27] Kommalapati, H., Christian, T., Drill Into .NET Framework Internals to See How the CLR Creates Runtime Objects [online], *MSDN Magazine*, May 2005, dostupné z: http://msdn.microsoft.com/msdnmag [cit. 2009-30-3].

[28] Shared Source CLI 2.0 Release [online], dostupné z: http://www.microsoft.com/downloads, [cit. 2009-30-3].

[29] Lidin, S., Inside Microsft .NET IL Assembler, Micorosft Press, 2002, 467 starn, ISBN 0-7356-1547-0.

[30] Xuelei Wu, Chen Jia, BiLan Rong, Research and Application on .NET and COM integrated technology, *International Symposium on Intelligent Information Technolgy Application Worskshops*, IEEE Computer Society, 2008, s.1001-1004, ISBN 9781424439041 .

[31] Stutz, D., Neward, T., Shilling, G., Shared Source CLI Essentials, O'Reilly, March 2003, 378 stran, ISBN 0-596-00351-X.

[32] Microsoft Visual Studio 2010 Overview [online], dostupné z http://www.microsoft.com/visualstudio/en-us/products/2010/default.mspx, [cit. 2009-30-3].

[33] Gunnerson, E., Začínáme programovat v C#, Computer Press, Praha, 2001, 334 stran, ISBN 80-7226-525-3.

[34] Richter, J., Garbage Collection: Automatic Memory Management in the Microsoft .NET Framework [online], *MSDN magazine*, November 2000, dostupné z http://msdn.microsoft.com/msdnmag , [cit. 2009-30-3].

[35] Kommalapati, H., Christian, T., Drill Into .NET Framework Internals to See How the CLR Creates Runtime Objects [online], *MSDN Magazine*, May 2005, dostupné z http://msdn.microsoft.com/msdnmag, [cit. 2009-30-3].

[36] Neable, C., The .NET Compact Framework, *IEEE Pervasice Computing*, October-December 2002, Vol 1., No 4, s. 84-87, ISSN 1536-1268.

[37] Rammer, I., Advanced .NET Remoting, APress, 2002, 400 stran, ISBN 1-59059-025-2.

[38] Lam, Ch. T., Ding, J. J., Liu, J.-Ch., XML Document Parsing: Operational and Performance Characteristics, *IEEE Computer*, September 2008, Vol. 1, Issue 9, s. 30-37, ISSN 018-9162.

[39] Nekwirk, J., Vorontsov, A., How .NET's Custom Attributes Affect Design, *IEEE Software*, September/October 2002, Vol. 19, No. 5, s. 18-20, ISSN 0740-7459.

[40] World Wide Web Consortium, WSDL specification [online], dostupné z http://www.w3.org/TR/wsdl, [cit. 2009-30-3].

[41] Klien, S., Professional WCF Programming, Wiley Publishing, 2007, 455 stran, ISBN 978-0-470-08984-2.

[42] Chengyun, Ch., Introduction to Microsoft .NET Security, *IEEE Security & Privacy*, November/December 2008, Vol. 6, Issue 6, s. 73-78, ISSN 1540-7993.

[43] LaMacchica, B. Lange, S. a kol, .NET Framework Security, Pearson Education, 2002, 816 stran, ISBN 067232184X.

[44] Sandhu, J.R., Malhotra, S., Microsoft .NET Framework Security, Premier Press, 2002, 408 stran, ISBN 1931841829.

[45] Cai, M., Ghandeharizadeh, S., Schmidlt, R., Song S., A Comparison of Alternative Encoding Mechanism for Web Services [online], dostupné z: http://dblab.usc.edu/microsoft/, [cit. 2009-30-3].

[46] Scriber, K., Windows Workflow Foundation Step by Step, Microsoft Press, 2007, 486 stran, ISBN 073562335X.

[47] Pialorsi, P., Russo, M., Introducing Microsoft LINQ, Microsoft Press, 2007, 240 stran, ISBN 0735623910.

[48] Osrael, J., Froihofer, L., Stoifl, G., Weigl, L., Klemen, Z., Habjan, I., Goeascka, M. K., Using Replication to Build Hihgly Available .NET Applications, In *Proceedings on the 17th International Conference on Database and Expert System Applications (DEXA'06)*, Krakow, Poland, 2006, s. 385 – 389.

[49] Seidmann, T., Distributed Shared Memory Using the .NET Framework, In Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and Grid (CCGRID'03), 2003, s. 457-462.

[50] Sadén, B., Coping with Java Threads, *IEEE Computer*, April 2004,Vol. 37, No. 4, s. 20-27, ISSN 0018-9162.

[51] Coffman, E.G., Elphick, M.J, Shoshani, A., System Dedlocks, *ACM Computing Surveys*, ASSOC COMPUTING MACHINERY, Vol. 3, No.2, June 1971,s.67-78, ISSN 0360-0300.

[52] Fraser, K., Harris,T., Concurrent programming without locks, *ACM Transactions on Computer Systems*, Vol. 25 , Issue 2, May 2007, s. 1-59, ISSN 0734-2071.

[53] Meyer, B., .NET Is Coming, *IEEE Computer*, April 2001,Vol. 34, No 4, s. 92-97, ISSN 018-9162.

[54] Tröger, P., Polze, A., Object and Process Migration in .NET, In: *Proceedings of 8th IEEE Internation Workshop on Object-Oriented Real-Time Dependeble Systems*, 2003, s. 139-146, ISBN 0-7695-1929-6.

[55] MPI specification [online], dostupné z: http://www.mpi-forum.org/docs/, [cit. 2009-30-3].

[56] MPI.NET: High-Performance C# Library for Message Passing [online], dostupné z: http://www.osl.iu.edu/research/mpi.net/, [cit. 2009-30-3].

[57] Benton, N., Cardelli, L., Fournet C., Modern Concurrency Abstraction for C#, *ACM Transactions on Progranmming Languages and Systems*, September 2004, Vol. 26, Issue 5., s. 769-804, ISSN 0164-0925.

[58] Lee, A. E., The Problem with Threads, *IEEE Computer*, May 2006, Vol. 39, Issue 5, s.33-42, ISSN 0018-9162.

[59] Viniski, S., Consurrency with Erlang, *IEEE Internet Computing*, September-October 2007, Vol. 11, Issue 5, s.90-93, ISSN 1089-7801.

[60] Erlang for .NET [online], dostupné z: http://erlangdotnet.net/, [cit. 2009-30-3].

[61] Menascé, D. A., MOM vs. RCP: Communication models for Distributed Applications, *IEEE Internet Computing*, March-April 2005, Vol. 9, No. 2, s. 90-93, ISSN 1089-7801.

[62] Rahul, V. P., George, B., Tools and Techniqeus to Identify Consurrency Issues [online], *MSDN Magazine*, dostupné z: http://msdn.microsoft.com/msdnmag, [cit. 2009-30-3].

[63] Song, H., Zhao, B., Zhou, Y., Lu Yi, Meng, W., Fast Design and Construction for Network Application Solution Based on .Net 3.5 Framework, In: *Proceedings of International Conference on Computer Science and Software Engineering 2008*, IEEE Computer Society, 2008, Vol. 2, s.420-423, ISBN 978-0-7695-3336-0.

[64] Guerraoui, R., A Smooth Concurrency Revolution with Free Objects, *IEEE Internet Computing*, July – August 2007,Vol. 11, Issue 4, s. 82-85, ISSN 1089-7801.

[65] Hinchey, G. M., Sterritt, R., Self-Managing Software, *IEEE Computer*, February 2006, Vol. 39, Issue 2, s. 107-109, ISSN 0018-9162.

[66] Candea, G., Brown, B. A., Fox, A., Patterson D., Recovery-Oriented Computing: Building Multitier Dependability, *IEEE Computer*, November 2004, Vol. 37, Issue 11, s. 60-67, ISSN 0018-9162.

[67] Smrčka, A., Úvod do formální verifikace [online], dostupné z: http://www.fit.vutbr.cz/~smrcka/fav/guide/index.html [cit. 2009-05-06].

[68] Maléř, O., Softwarová architektura, *COMPETERWORLD*, 2008, č. 11, s. 42-43, ISSN 0010 – 4841.

[69] Colwell, B., Complexity in Design, *IEEE Computer*, October 2005, Vol. 38, Issue 10, s. 10-12, ISSN 0018-9162.

[70] Ferreira, P., Veiga, L., Ribeiro, C., OBIWAN: Design and Implementation of Middleware Platform, *IEEE Transaction on Parallel and Distributed Systems*, November 2003, Vol 14, No 11, s.1086-1099, ISSN 1045-9219.

[71] Towned, P. a kol., CROWN-C: A High-Assurance Service-Oriented Grid Middleware System, *IEEE Computer*, 2008, August 2008,Vol. 41, No. 8, s.30-38, ISSN 0018-9162.

[72] Laddad, R., AspectJ in Action – Practical Aspect-Oriented Programming, Manning Publications, 2003, 513 stran, ISBN 1-930110-93-6.

[73] Lopes, C. V., Kiczales G., D: A Language Framework for Distributed Programming [online], Technical report, Xerox 1997, dosupné z: http://www2.parc.com/csl/groups/sda/publications/papers/PARC-AOP-D97/for-web.pdf , [cit. 2009-30-3].

[74] Zhang, Ch., Jacobsen, H., Refactoring Middleware with Aspects, *IEEE Transactoins on Parallel and Distributed Systems*, Vol. 14, No. 11, November 2003, s. 1058-1073, ISSN 1045-9219.

[75] Richter, J., Implementing the CLR Asynchronous Programming Model [online], *MSDN Magazine*, March 2007, dostupné z: http://msdn.microsoft.com/msdnmag, [cit. 2009-30-3].

[76] Al-Jaroodi, J, Jiang, H., Middleware Infrastructure for Parallel and Distributed Programming Models in Heterogenous Systems, *IEEE Transactions on Parallel and Distributed Systems*, November 2003, Vol. 14, No. 11, ISSN 1045-9219.

[77] Fober, D., Orlarey, Y., Letz, S., Lock-Free Techniques for Concurrent Access to Shared Objects [online], dostupné z: http://www.grame.fr/pub/fober-JIM2002.pdf, [cit. 2009-30-3].

[78] Halpert, R.L., Pickett, Ch., Verbrugge, C., Component-Based Lock Allocation, In *Proceedings of 16th International Conference on Parallel Architecture and Compilation Techniques (PAST 2007)*, Brasov, Romania, IEEE Computer Society, 2007, s. 353-364.

[79] Williams, A., Thies, W., Ernst, D. M., Static deadlock detection for Java libraries, In *Proceedings of 19th European Conference on Object-Oriented Programming ECOOP 2005*, 2005, Glasgow, Scotland, s. 602-629, ISBN 978-3540279921.

[80] Zhang, Y., Sreedhar, C. V., Zhu, W., Sarkar, V., Gao R. G., Minimum Lock Assignment: A Method for Exploiting Concurrency amog Critical Sections, In *Proceedings of 21th Annual Workshop on Languages and Compilers for Parallel Computing (LCPC 2008)*, 2008, Edmonton, Alberta, Canada, s. 141 - 155, ISBN 978-3-540-89739-2.

[81] Schmidt, D., Stal, M., Rohnert, H., Buschmann, F., Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects, Volume 2, John Wiley & Sons, 2000, 633 stran, ISBN 0471606952.

6.2 SEZNAM VYBRANÝCH VLASTNÍCH PRACÍ

[82] Vítek M., Herman I., Uchytil, S., Run-time code generation on the .NET Framework platform, In *Proccedings of the Telecommunications and Signal Processing Conference, TSP-2005*, Brno University of Technology, 2005, s. 57 - 61, ISBN 80-214-2972-0.

[83] VÍTEK, M., HERMAN, I. IO Interface of Communication Components. In *Proccedings of the Telecommunications and Signal Processing Conference TSP-2004*. Brno University of Technology, 2004. s. 205-208. ISBN 80-214-2684-5.

[84] VÍTEK, M., GREGOŘICA, M., KOMOSNÝ, D. Řízení a přístup ke službám hromadné radiové sítě [online], Elektrorevue - Internetový časopis, 2004, roč. 2004, č. 57, s. 0-7. ISSN 1213-1539, dostupné z: http://www.elektrorevue.cz.

[85] Vítek M., Herman I., Uchytil S. Run-time code generation on the .NET Framework platform. In *Proceedings of the Telecommunicatioins and signal processing conference*. Brno University of Technology, 2005. s. 57-61. ISBN 80-214-2972-0.

[86] Vítek M., Uchtil S., Herman I., Stateful Web Services Using WSE, In Proceedings of the 9th WSEAS International CSCC Multiconference. 9th WSEAS International Conference on Communications, Athens, 2005, 5 stran, ISBN 960-8457-29-7.

[87] Vítek, M., Uchytil S., Miklánek, T., System for request parallel processing. In *Proceedings of Resarch In Telecommunication Technology RTT 2005*, Ostrava, 2005. s. 1-7, ISBN 80-248-0897-8.

[88] Miklánek T., Herman I., Vítek M. Stavová komunikace pod platformou .NET Framework. In *Research in telecomunication technology 2005*. Ostrava, 2005. s. 1-6. ISBN 80-01-03063-6.

[89] Herman, I., Vítek, M., Miklánek, T., Radiová síť FD NET s integrovanými datovými přenosy. Dopravní magazín, 2004, roč. 4, č. 1, s. 22-23.

[90] Vítek, M., Herman, I., Remote Communication Using WSE 2.0. In *Research in telecommunication technology 2004*. Praha, 2004. s. 1-4. ISBN 80-01-03063-6.

[91] Vítek, M., Herman, I., Miklánek, T. Stateful Communication with Web Services. In *Telecommmunications and signal processing TSP 2004*, Brno, 2004. s. 158-161, ISBN: 80-214-2684-5.

[92] Vítek, M., Herman, I., Conception of Control System of Network with Dynamic Endpoints. In *Proceedings of Telecommunications and Signal Processing 2003*. Brno, 2003, s. 233-236, ISBN 80-214-2433-8.

[93] Vítek, M. Using SOAP and Web Services Technologies for Communication Protocol Design. *In Research in Telecommunication Technology RTT 2003*. Bratislava, 2003. s. 241-244, ISBN 80-227-1934-X.

[94] Herman, I., Vítek, M., Miklánek, T., Rádiová síť FDNET pro Dopravní podnik Ostrava a.s., 2003, výzkumná zpráva pro DPO, s. 1-120.

[95] Vítek, M., Herman, I., Principle of Multifunctional Server based on Ticket Processing. In *Proceedings of Parellel and Distributed Computing and Networks IASTED 2004*, Innsbruck, 2004, s. 508-513, ISBN 0-88986-385-7.

[96] Vítek, M., Princip využití technologie webových služeb pro komunikaci typu peer-to-peer [online], *Elektrorevue - Internetový časopis*, 2002, roč. 2002, č. 48, s. 1-8. ISSN 1213-1539, dostupné z: http://www.elektrorevue.cz.

[97] Vítek, M., Herman, I., Model pro popis sdílených prostředků a jejich závislostí vycházející z OOP principů [online], *Elektrorevue - Internetový časopis*, 2009, roč. 2009, s. 1-7, ISSN 1213-1539, dostupné z: http://www.elektrorevue.cz, *(předpokládaný termín publikace článku – červen 2009)*.

CURICULUM VITAE

Jméno: Ing. Martin Vítek
Narozen: 14.4.1979
Kontakt: vitekm@feec.vutbr.cz

Vzdělání

1993-1997	Všeobecné gymnázium Moravský Krumlov
1997-2002	Fakulta elektrotechniky a komunikačních technologií, VUT v Brně
2002-	doktorské studium na Ústavu telekomunikací, VUT v Brně

Praxe

2002-2005	programátor, analytik
2005-2008	vývojář, konzultant
2009-	vývojář, projekt manažer

Účast na řešení projektů

2002	Účast na projektu FRVŠ reg. č. 1071/2002/F1 s názvem „Moderní způsoby komunikace v předmětu komunikační sítě a techniky“.
2002	Účast na projektu s názvem „Aplikovaný výzkum technologií pro multimediální a hypermediální služby“, ident. kód FD-K/040.
2002	Účast v projektu MŠMT „Vytvoření encyklopedie komunikačních technologií a její zpřístupnění pomocí Internetu“, reg č. LP01060.
2003	Účast na projektu reg. č. IS432129 „Video přednášky do kurzu Mikroprocesorová technika v telekomunikacích“.
2003	Řešení grantu FRVŠ reg. č. 2204/2003/G1 s názvem „Signalizační protokol pro H.323/Radio gateway“.
2004	Řešení grantu FRVŠ reg. č. 1645/2004/G1 s názvem „Koncepce modulů pro stavovou komunikaci webových služeb“.
2005	Řešení grantu FRVŠ reg. č. 2790/2005/G1 s názvem „Middleware na platformě .NET Framework“.
2005	Účast na projektu FRVŠ reg. č. 3004/2005/F1 s názvem „Moderní způsoby komunikace v předmětu Pokročilé komunikační techniky“.
2003-2005	Účast na projektu GAČR reg. č. GA 102/03/1033 s názvem „Komunikační protokoly s dynamickým směrováním“.

ABSTRACT

With the expansion of the Internet communication and related availability of increasing number of services built on different technologies, distributed systems represent a solution to integrate these network services and provide them to users in a coherent form. The .NET Framework which provides an environment for application development in a highly distributed environment of Internet and intranet can be used to achieve this goal.

This PhD thesis deals with access to shared resources in the context of distributed systems using the .NET platform. The first part of the work is devoted to describing the basic principles of distributed systems and .NET platform techniques, which can be used for implementation of the principles. For the purposes of request processing having asynchronous nature not only in distributed systems a universal interface for the description of asynchronous operations was designed and implemented. The interface extends standard asynchronous techniques on the .NET platform.

In order to address the issue of access to shared resources model was designed based on the principles of object-oriented programming, along with basic algorithm to avoid deadlock in the case of use resources by multiple processes (threads) simultaneously. This extendable model has been successfully implemented and its functionality verified in basic scenarios of access to shared resources. After the definition of resources and their dependencies the implemented model allows working with resources as with any other objects on .NET platform. The synchronization processes proceed transparently in background.