ЮБИЛЕЙНА НАУЧНА СЕСИЯ – 30 години ФМИ
ПУ “Паисий Хилендарски”, Пловдив, 3-4.11.2000

МУЛТИ-АГЕНТЕН ПОДХОД ЗА ИЗГРАЖДАНЕ НА РАЗПРЕДЕЛЕНИ ИНФОРМАЦИОННИ СИСТЕМИ

Станимир Недялков Стоянов

В публикацията е представен един подход за разработване на съвременни разпределени бизнес-системи. Изхождайки от потребителските изискванията към тези системи и технологичните предизвикателства, които съществуват за тяхното удовлетворяване е представена една възможна технологична рамка за разработване на приложения в телекуманикационните услуги и в електронния бизнес. Накратко е разгледана симулационната среда M-STEBS, която е предназначена за провеждане на експерименти с описаните в публикацията архитектури.

Ключови думи: *софтуерни технологии, многослойни архитектури, агентно-ориентиран подход, компонентно-ориентиран подход, симулационни среди, телекомуникационни услуги, електронен бизнес, Java*

1 Въведение

В различни приложни области (напр. електронната търговия) се наблюдава явна тенденция за все по-тясно интегриране на “класически” бизнес-услуги с телекомуникационни услуги. По силата на вътрешната си логика тези два вида услуги се развиваха във времето самостоятелно. Еволюцията на конвенционалните мрежи и мрежите за пренос на данни разкрива нови възможности за предоставяне на интелигентни телекомуникационни услуги. В момента телекомуникационните услуги са под изключителния контрол и експлоатация на мрежовите оператори. Причините за това са различни – от една страна непълните стандарти за телекомуникации, от друга - усилията за осигуряване на сигурност и цялостност на мрежите. Като резултат от това наблюдаваме централизирани комуникационни услуги и разпределителни механизми, които работят в мрежовия домейн, но не предлагат достъп до данни в други приложни области. Съществен момент в посока на децентрализиране на телекомуникационните услуги е търсенето на възможности за преместване на контрола извън областта на мрежовите оператори към външни бизнес-среди.

В публикацията разглеждаме един подход за разработване на информационни бизнес-системи от ново поколение, като се отчитат тези интеграционни тенденции. Необходимите за тази цел технологии трябва да предоставят адекватни решения на следните два въпроса:

- Как ще се поддържа обработката на разпределената функционалност на системата в една хетерогенна среда?
- Как да се предоставят на потребителите лесни и интуитивни средства за работа със системата, като се “скрие” от тях вътрешната сложност на системата?

2 Описание на подхода

В основата на предложения в публикацията подход е залегнало разбирането, че една информационна система трябва (освен другите неща) да се разглежда в два аспекта:

- *Потребителски аспект* – системата се интерпретира като една архитектура, състояща се от сървърна и клиентска част. Сървърът предлага едно множество от *услуги*, които потребителите могат да използват посредством *заявки* към системата. Обработката на услугите е прозрачна за потребителите. Клиентската част служи за връзка на потребителите със системата.
- *Технологичен аспект* – разглежда системата като ансамбъл от отделни логически свързани помежду си програмни средства, софтуерни модули, компоненти и автономни агенти, които реализират определена стандартизирана функционалност и могат да се интегрират в една архитектурна рамка. В общия случай потребителите не познават в детайли технологичния аспект.

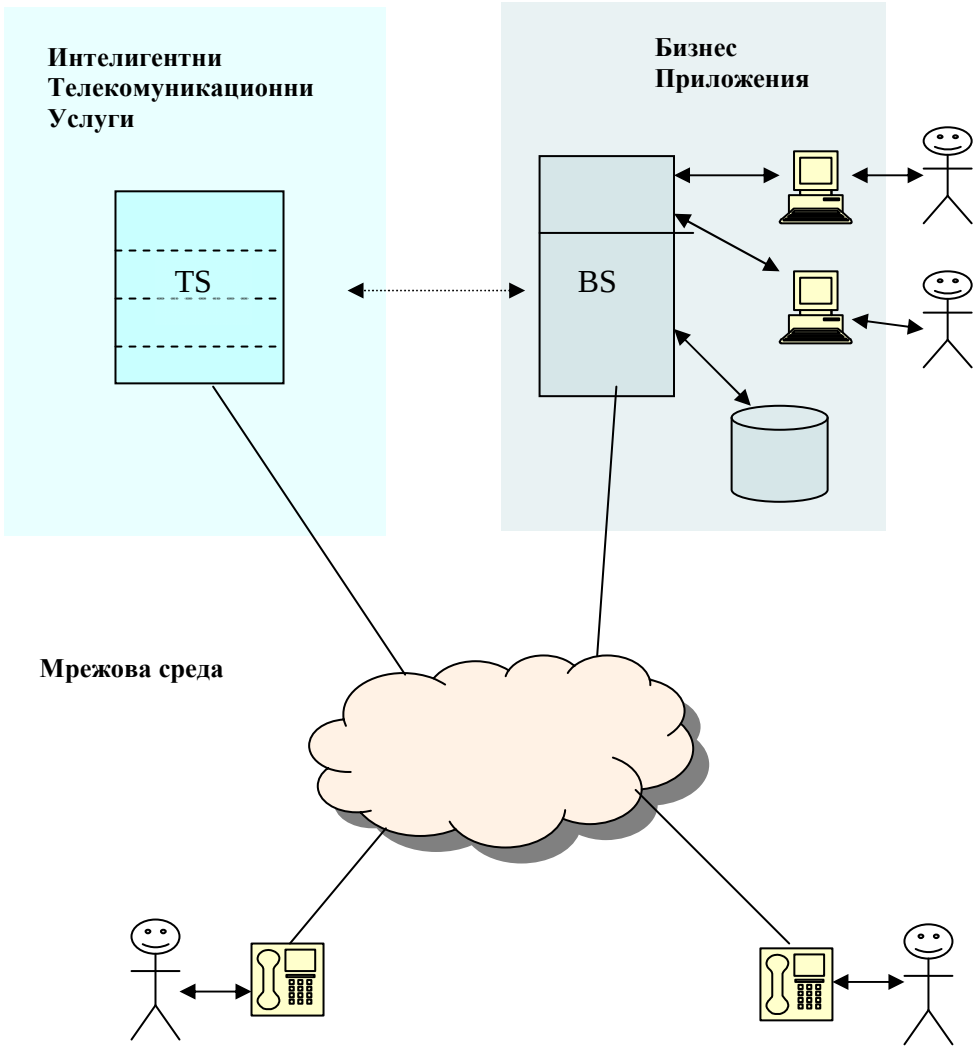
От потребителска гледна точка разбирането на същността, смисъла и използването на услуги, като че ли е ясно и не създава особени прблеми. От този аспект една услуга е една логически завършена функционалност, обработката на която носи някакъв резултат за потребителите. Не така лесно е обаче дефинирането на услугите от технологична гледна точка. Създаване на архитектура за програмна реализация на услугите, която да отговаря на съвременните изисквания е един сложен проблем. Към една такава архитектура се поставят много трудни и в определен смисъл противоречащи си изисквания, като напр. гъвкавост, преносимост и адаптеруемост, обработка в динамично променящи се околни среди, различни форми на управление – от силно централизирано до напълно децентрализирано.

В представената в публикацията технология една услуга се разглежда като структура, състояща се от два компонента: *ядро* – реализира функционалност (съдържание на услугата), *обвивка* – позволява ядрото да се вгражда в съответната оперативна среда. Така дефинирана услугата може да се реализира като обект, компонент или агент. За реализирането на този модел от решаващо значение е търсене на възможности за специфициране на стандартизирани логически обособени функционалности за дадена конкретна приложна област. В областта на електронната търговия стандартизираните функции се наричат ECBBs (Electronic Commerce Building Blocks) [4], а в областта на телекомуникациите SIBs (Service Independent building Blocks) [5].

Основната идея на нашия подход ще представим посредством следната опростена (и донякъде идеализирана) схема за работа с една бизнес-система:

- Потребителите изискват извършването на определена услуга от системата посредством задването на една заявка R. За потребителите заявката е атомарна.
- Системата декомпозира заявката в едно множество от системни услуги, т.е. $R = \{s_i, \ldots, s_j\}$, където $i,j \in [1, \ldots, n]$, $s_i \in BS \cup TS$, $BS \cap TS = \varnothing$, BS – множеството на бизнес-услугите, TS – множеството на телекомуникационните услуги. За всяка системна услуга съществува съответен програмен модул, който я реализира.
- От параметрите на заявката и от информацията за предлаганите системни услуги системата генерира динамично бизнес-логика bl, която ще управлява изпълнението на заявката. $bl(\{s_i, \ldots, s_j\}) = \langle s_1, \ldots, s_n \rangle$ е една наредба в множеството на услугите. Тази наредба ще наричаме условно *bl-transaction* (различна от транзакциите на ниво бази данни).

- Системният управляващ софтеуер (системен run-time) обработва транзакцията.
- Системата връща резултата на потребителите.



Фиг. 1

На Фиг. 1 е дадена една най-обща архитектура на информационна система, която може да се разработи с описания подход (и поддържащите го технологични и развойни средства).

Една такава архитектура ще изграждаме поетапно, като използваме следните три технологични средства:

- MALINA - технология за изграждане на мулти-агентни приложения

- Симулационна среда за обработка на интелигентни телекомуникационни услуги
- Многослойна софтуерна архитектура за интегриране на бизнес-услуги.

3 MALINA – технология за изграждане на мулти-агентни приложения

В този раздел ще представим накратко технологията за създаване на мулти-агентни приложения MALINA (Multi-Agent Local Integrated Network Associations), основните идеи на която са дадени в [6,7]. Технологията представя един bottom-up подход [8] за изграждане на мулти-агентни приложения, като поетапно специфицира една инфраструктура на разпределени софтуерни системи. MALINA специфицира следните три нива:

- *РДЗС (Разпределени Динамично Зависими Системи)* – една хипотетична инфраструктура, която представя теоретико-концептуалната рамка на технологията.
- *Концепции* – представят една декомпозиция и детайлизация на хипотетичната инфраструктура, с цел подготовка разработването на поддържащи програмни средства и съответна развойна среда.
- *Развойна среда и run-time модул* – конкретните програмни средства, поддържащи технологията.

3.1 Разпределени динамично зависими системи

РДЗС е една хипотетичната инфраструктура, основните характеристики на която ще представим накратко в този раздел.

Нека съществува едно множество от *идентичности* (реални или абстрактни обекти, субекти, ...), които притежават определена функционална завършеност. Идентичностите са автономни, проактивни и независими помежду си, което предполага, че те притежават:

- *локално управление* – предоставя възможности за самоактивиране на идентичностите (напр. те могат да поемат инициатива за решаване на определен проблем). Това е една съществена разлика от обектно-ориентирания подход, където обектите реагират само на външни събития.
- *локални ресурси* – за всяка идентичност те са ограничени по обем и във времето.

Идентичностите имат възможност да разменят информация във формата на *съобщения*.

В определени моменти във времето могат да настъпват събития (напр. създава се ситуация, поражда се необходимост от решаване на общ проблем), които ще наричаме *пораждащи събития* и спрямо които между идентичностите се появяват определени *частични зависимости*. Така в контекста на определена ситуация (напр. решаване на обща задача, действие, решение, план, мисъл, идея) идентичностите стават *динамично зависими*. Отделните идентичности могат да влияят индивидуално или съвместно (кооперирано) върху *общи (поделени) ресурси*, които са свързани с обработването на съответното събитие. При настъпването на нови събития (ситуации, конфигурации, ...) динамичните зависимости изчезват или се заменят с нови. Отделните идентичности могат да се намират едновременно в повече от една динамични зависимости.

Системи, ситуации или конфигурации, които можем да определим като РДЗС, имат обикновено сложна структура и еднозначното им характеризиране или специфициране е особено труден за решаване проблем. Анализът и синтезът на такива системи изисква

един многопластов, многокритериен и комплексен подход. За подпомагане решаването на проблемите, свързани с комплексността на системите ще дефинираме понятието *аспект*. Аспектите ще обединим в две големи групи: базови и организационни.

Базовите аспекти характеризират същността на системата. Те могат да бъдат статичен, динамичен, темпорален и модален. *Статичният аспект* характеризира разпределената функционалност и инфраструктурата на системата. С помощта на *динамичния аспект* могат да се изследват необходимите за взаимодействието операции, като напр. синхронизация, комуникация, кооперация, договаряне. Освен това динамичният аспект характеризира и същността на събитията (ситуациите), пораждащи динамични зависимости и същността на самите зависимости между статично независимите и автономни идентичности . *Темпоралният аспект* характеризира от една страна, развитието на пораждащите събития (ситуации, конфигурации, ...) във времето, а от друга страна - промяната във времето на ролите на отделните идентичности и зависимостите между тях. *Модалният аспект* характеризира различни условия, условности, изключения от правилата, възможности или необходимости. Той налага същевременно различни ограничения в действията на отделните идентичности. В процеса на развитие на събитията (ситуациите) модалностите могат също така да се променят.

Основен организационен аспект е *управлението*. В общия случай организационните аспекти са функции на определени базови аспекти. Така напр. функционалната независимост и автономност предполагат наличието на децентрализирано (локално) управление на отделните идентичности. При възникване на определена динамична зависимост се появява необходимост от някакъв вид централизирано управление. Спрямо този аспект една система може да бъде с децентрализирано управление и с възможности за някакъв вид централизирано управление.

За потребителите една РДЗС е множество от различни *услуги*. Потребителите изискват изпълнението на някаква услуга от системата чрез *заявки*. Услугите се реализират посредством взаимодействие (динамични зависимости) между отделни компоненти на системата.

3.2 Концепции

Следващата стъпка, която специфицира технологията MALINA е конкретизирането и детайлизирането на хипотетичната инфраструктура. В технологията са разработени следните четири концепции, които определят рамката на нейната развойна среда:

- *Abstract Agent Architecture* – специфицира архитектурата на един абстрактен агент, с помощта на който могат да се генерират агентите на едно приложение.
- *AgentCities* – концепцията специфицира една статичната инфраструктура за мулти-агентни приложения.
- *AgentAssociations* – спецификация на възможностите за образуване на агентни общности.
- *MobileServices* – специфицира начините за автоматично генериране на мобилни услуги.

Абстрактен Агент е една генетична абстрактна структура $A = <\pi,\phi,\mu>$, от която се генерират агентите на приложението като нейни копия (екземпляри, инстанции) и където:

- π - *агентна машина*, която включва интерфейси на агента с околната среда (сензори и ефектори) и локално управление
- ϕ - *специализация* на агента. Това е един абстрактен интерфейс към някакъв програмен компонент, който реализира функционалността на агента.
- μ - *менталност* на агента. Това е също така един абстрактен интерфейс към една силно структурирана *база знания*. Базата знания се състоя от отделни независими един от друг *сегменти*. Всеки сегмент съдържа знания и съответни обработващи механизми, които специфицират менталните свойства (напр. вяра,намерения, ангажименти) на агента.

Концепцията *Agent Cities* специфицира една отворена инфраструктура за моделиране на много-агентни системи в разпределена среда. В концепцията се дефинират две адресни пространства:

- *Логическо (концептуално) адресно пространство* – видимо за проектантите и потребителите на приложенията. Това пространство може да бъде използвано и от оперативните агенти на съответното приложение. В това адресно пространство агентите се разполагат в локални логически пространства, наречени *градове*. Определени агенти могат да бъдат мобилни (с възможности за преминаване от град в град);
- *Физическо (мрежово) адресно пространство* – прозрачно за потребителите и проектантите. Това адресно пространство се обслужва от системни агенти, които са съставна част на run-time модула на съответното мулти-агентното приложение.

Една *много-агентна система* е съвкупност от всички съществуващи градове в едно приложение.

Множеството на позиционираните в един град агенти не е още агентна общност. Взаимодействието между агентите в един град е източник на допълнителна семантика, която не може да бъде зададена в отделните агенти. Агентната общност се дефинира посредством:

- Взаимодействието между агентите
- Модела на устройство и управление на града.

В концепцията взаимодействието се разглежда като състоящо се от три слоя:

- *Комуникация* – най-простата форма на взаимодействие, която реализира обмен на съобщения между агентите.
- *Синхронизация* – по-сложна форма на взаимодействие, която може да се използва за синхронизиране на отделните агенти за изпълнение на общи задачи. Приложима е само за агенти с ментални свойства.
- *Кооперация* – най-висшата форма на взаимодействие. Използва се за подготовка и планиране на съвместни задачи. Приложима само за агенти с ментални свойства.

За реализиране взаимодействието между агентите се използват де факто стандартите KQML [9] и KIF [10].

Възможни модели на управление на градовете са следните:

- *Централизирано управление* – обикновено в рамките на един град съществува един интелигентен агент и множество агенти без ментални свойства. Интелигентният агент получава заявката и той определя задачите на другите агенти (т.е. той изпълнява ролята на планировчик и възложител на задачите). Възможни са

различни разновидности на този вид управление (силно, слабо, ...). Централизираното управление се реализира чрез комуникация и синхронизация.

- *Децентрализирано управление* – в града всички агентни са интелигентни. За изпълнение на една заявка агентите трябва да могат да се кооперират. Също различни разновидности са възможни (силно, слабо, ...).

3.3 Развойна среда

Развойна среда на технологията включва следните групи програмни средства:

- Конфигурационни агенти
- Системни агенти
- Стандартизирани помощни и приложни агенти – напр. интерфейсни агенти.

Конфигурационните агенти подпомагат проектантите на агентно-базираните приложения при:

- генериране на оперативните агенти (реализират разпределената функционалност на приложението) – в съответствие с концепцията AbstractAgentArchitecture
- конфигуриране на статичната инфраструктура на приложението – в съответствие с концепцията AgentCities.

Системните агенти образуват run-time на мулти-агентните приложения. Те реализират следните основни функции:

- Управление на логическото адресно пространство
- Управление на физическото адресно пространство
- Трансформации между двете пространства
- Управление на динамична мобилност на агентите

4 Интелигентни мрежи

Основната задача на една Интелигентна Мрежа (ИМ) е независимо от използваната мрежа за комуникация изграждане и управление на интелигентни телекомуникационни услуги. Интелигентните мрежи интегрират в себе си функционалност (съдържание на услугата) и разпределена софтуерна архитектура, която представя структурата на системата и се прилага за идентификация на използваните протоколи, стандарти и продукти. В основата на ИМ е залегнала концепцията за Service Creation Environment (SCE), която специфицира една среда за създаване, поддържане и развитие на услуги. В тази среда потребителите могат да дефинират различни интелигентни телекомуникационни услуги, които удовлетворяват различни техни изисквания и условия. С помощта на SCE телекомуникационните услуги могат бързо да се концептуализират, разработват и изпълняват като комбинации от стандартизирани функционални софтуерни пакети (напр. SIBs).

За управление на ИМ се използват основно два функционални блока:

- *SMAF (Service Management Access Function)* – функция, управляваща достъпа до услугите.
- *SMF (Service Management Function)* – функция за управление на услугите.

Докато в началния етап на разработване на ИМ се акцентираше предимно върху изграждане на приложната логика на услугите, напоследък все по-често се третират различни аспекти, свързани с въвеждането на нови услуги в реални условия. В процеса на все по-тясното интегриране на компютърните и телекомуникационните технологии

мрежовите оператори се конфронтират все повече със съвместимостта на околната среда. Все по-голяма става необходимостта от извеждане на контрола и управлението от телекомуникационния домейн в актуалните бизнес-среди. При създаването на нови услуги е необходима както бърза и гъвкава реакция, съобразена с пазарните изисквания, така също и осигуряване на нейните оперативни аспекти. Способностите за самостоятелен контрол върху услугите увеличават също така тяхното абонаментно обслужване. Средствата за улесняване използването на услугите предоставят на доставчиците възможности за разработване на собствено ноу-хау за интелигентни телекомуникационни услуги, което ги прави независими от доставчиците на оборудване.

Жизненият цикъл на услугите в една ИМ се състои от два основни етапи:

- *Изграждане на услугата* – покрива всички фази, през които тя трябва да премине - от първоначалната идея до експлоатацията ѝ в реална мрежа.
- *Изпълнение на услугата* – покрива всички аспекти на нейното изпълнение, след като тя е разгърната в мрежата.

Приложната логика на интелигентните услуги се дефинира с помощта на граф, възлите на който са SIBs. Изграждащите блокове капсулират в себе си определена логически завършена функционалност и съответните структури данни. Функционалните блокове могат да бъдат параметризирани. Логиката на обработка на функционалностите е отделена от данните. По този начин се постига една задоволителна гъвкавост на услугите и лесно адаптиране към специфичните нужди на различните потребители. SIBs се специфицират с помощта на съответни шаблони, които се задават в специализиран език (STL – Sib Template Language). В обсега на ИМ се включват дейности за:

- Потвърждение и управление на клиенти
- Определяне на свойствата на услугите посредством комбиниране на различни SIBs
- Дефиниране на пълни услуги чрез комбиниране на свойства
- Регулиране и промяна според нуждите на правата за достъп и т.н.

Бързото развитие на телекомуникациите през последните години предопределя необходимостта от високоскоростен обмен на данни. Това, от своя страна, води до необходимосттта от използване на високо скоростна преносна среда и мощни изчислителни машини. Изпълнието на тези изисквания поражда нов проблем – несъвместимост между някои от досега използваните протоколи. Това налага естествената необходимост от дефиниране на нови платформено-независими стандарти, обединяващи функционалността на досега съществуващите.

В момента се разработват два основни стандарта за интелигентни телекомуникационни услуги:

- *Parlay* - разработван от консорциума Parlay group [11].
- *JAIN (Java API for Intelligent Networks)* - разработван от корпорацията SUN Microsystems [12]. Силата на JAIN се състои в това, че наред с решаването на проблема за независимост от платформите, този стандарт е съвместим със съществуващите и използвани вече стандарти за изграждане на телекомуникацианни архитектури. JAIN внася нови измерения в изграждането на разпределени услуги и приложения, като повишава ефективността на инструментите, използвани за изграждане на ИМ.

Двата стандарта използват различни подходи за решаване на едни и същи проблеми и в определена степен са противоречиви. Времето ще покаже дали ще бъдат обединени в един по-общ модел.

Освен тези стандарти съществена роля за изграждане на приложения, включващи телекомуникационни услуги, играе и Parlay API (Application Program Interface) [13. Това е един отворен, технологично и мрежово независим интерфейс, който доставя сигурен и свободен достъп до възможностите на телекомуникационните мрежи.

5 Мнослойни софтуерни архитектури

Повечето от съвременните бизнес-приложения използват многослойни софтуерни архитектури. Една типична клиент/сървер система може да се разглежда като двуслойна архитектура, при която приложението е разположено върху персоналния компютър на клиента, а базата данни – върху сървера на доставчика на услуги. Въпреки, че този подход ни позволява да използваме поделена информация, той има също така много недостатъци [14].

При двуслойните архитектури обработката се извършва върху клиентския компютър, докато много по-мощният сървер функционира като контролер на трафика между приложението и базата данни. От това страда производителността на приложението и се увеличава трафикът в мрежата. Друг проблем е поддръжката. Дори най-малките промени в приложението ще изискват пълно реорганизиране на базата данни.

Тези проблеми се решават с разработването на трислойни и многослойни архитектури. При тях приложението е обособено в три отделни логически слоя, всеки от които има добре дефинирано множество от интерфейси. Първият слой (*представителен*) обикновено съдържа някакъв графичен потребителски интерфейс. Средният слой (*приложен сървер*) реализира логиката на приложението. Третият слой (*сървер на база данни*) съдържа необходимите за приложението данни.

Преминаването от трислойна към многослойна архитектура става като средният слой (цялото приложение) се декомпозира на повече отделни слоеве. Това е типично за обектно-ориентираните, компоненто-ориентираните и агентно-ориентираните подходи.

При използване на многослойни архитектури значително се подобрява производителността, разтоварва се мрежовия трафик и се улеснява поддръжката на приложенията.

6 Симулационна среда M-STEB

M-STEBS (Malina Simulation of Telecommunication and Electronic Business Services) е един проект за симулационна среда, с реализирането на който преследваме следните цели:

- Модифициране и тестване на технологията MALINA за изграждане на компонентно-базирани мулти-агентни системи.
- Разработване на прототип на приложение, поддържащо интелигентни телекомуникационни услуги.
- Създаване на многослойна архитектура за интегриране на различни бизнес-услуги.
- Разработване на стандартизирани интерфейси, свързващи бизнес-услугите и телекомуникационните услуги.

Поради сложността на проблемите, които трябва да се решават, M-STEBS е декомпозиран на четири по-малки самостоятелни проекти:

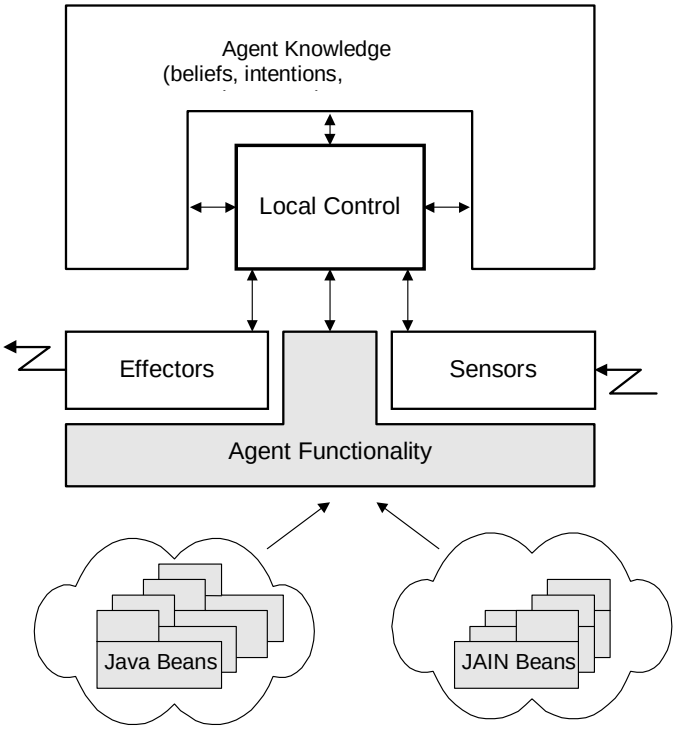
- *MALINA-IS* – модификация на технологията за приложение в интелигентните мрежи и електронната търговия.
- *Ti-JEINA* – симулация на интелигентни телекоминикационни услуги, използваща JAIN стандарта.
- *MLABA* – мнослойна архитектура за интегриране на интелигентни бизнес-услуги.
- *PAPI* – едно множество от стандартизирани интерфейси за интегриране на бизнес-услуги и телекомуникационни услуги.

6.1 MALINA-IS

MALINA-IS е модификация на базовата технология за реализиране на телекомуникационни услуги и бизнес-услуги като агенти с различна степен на интелигентност. Основните промени са свързани с вътрешната архитектура на агентите (Фиг.2). Развойната среда ще се разшири, като в нея се интегрират следните два вида стандартни програмни библиотеки:

- *Service Independent Components Library* – модулите, които реализират компонентно-ориентирани версии на SIBs.
- *Electronic Commerce Building Blocks Library* – компоненти, реализиращи ECBBs.

Тази архитектура позволява разработването на бизнес-приложения като автономни агенти, стандартни компоненти (напр. Java Beans) и JAIN компоненти (JAIN Beans).



Фиг. 2

6.2 Ti-JEINA

Ti-JEINA [1] е симулационна среда за провеждане на експерименти с различни интелигентни телекомуникационни услуги, базирани на стандарта JAIN. За пренос на данни се използва спецификацията TCAP и стандартът за кодиране на съобщения в телекомуникационни мрежи ITU. Предвижда се средата да се изгражда на няколко етапа.

През първия етап е разработена основната рамка на системата, която включва симулация на компонентите на ниско ниво (JAIN TCAP Stack, Provider Ti-JEINA), необходими за реализирането на интелигентни услуги. За верификация на разработката се използват седемнадесет теста, предоставени от SUN.

Във втория етап системата беше надстроена с интелигентната услуга Green Phone (Time Window), която предоставя възможности за специфициране на различни условия (преместване на разговора във времето, условия за плащане, забрана за разговор и др.) към телефонните разговори на абонатите.

6.3 MLABA

MLABA (Multi Layered Architecture for Business Applications) е един научно-изследователски проект за разработване на системна архитектура за интегриране на различни бизнес-услуги. Архитектурата поддържа обработка на заявките на потребителите на системата в съответствие представената по-горе схема. Едно по-детайлно представяне на проекта може да се намери в [2]. Тук ще дадем само кратко описание на функционалността на системата.

В рамките на проекта се разработва една многослойна архитектура. Клиентската част служи за получаване на заявки към системата и за визуализиране на резултатите. Върху сървера за бази данни се съхранява постоянната информация на системата.

Най-интересният компонент на архитектурата е приложният сървер. Самият той се изгражда като една трислойна структура, която включва в себе си описаните по-долу слоеве.

Първият от тях е *управляващ слой*. В него са включени различни стандартни модули, които реализират необходимите за изпълнението на заявките управляващи и контролни функции. Управлението изпълнението на заявките се извършва от следните стандартни за архитектурата компоненти:

- *RequestAnalyser* - заявките на потребителите (доставени от клиентския слой) се анализират и от тях се извличат конкретните стойности на определени параметри.
- *BL-Generator* – от конкретните стойности на параметрите на съответната заявка и от информацията за услугите, предлагани от приложния сървер, този модул генерира динамично бизнес-логиката, която е необходимо за обработка на заявката.
- *BL-TransactionGenerator* – генерира транзакциите, които трябва да се обработят за изпълнение на заявките.
- *BL-TransactionManager* - управлява изпълнението на bl-транзакциите.

Вторият слой съдържа модули, реализиращи стандарните бизнес-услуги, предлагани от системата. Архитектурата предоставя едно множество от стандартизирани услуги, което може да бъде допълвано от потребителите за

конкретните приложни области, където ще се използва приложението. Някои от стандартизираните услуги са следните:

- *Класификации* – тези услуги могат да решават класификационни проблеми, използвайки *теорията на грубите множества* [15,16] и алгоритми, използващи *CBR-подходи* [17].
- *Data Mining* – генерират изводи и анализи на основата на фактологията, съхранявана в базата данни.
- *Генетични алгоритми* – използват се за решаване на оптимални задачи.
- *EDI* – услуги, поддържащи електронен обмен на данни по определени стандарти.

Третият слой реализира различни видове интерфейси към сървера на базите данни.

6.4 PAPI

Целта на проекта PAPI [3] е да се реализират някои от основните спецификации на интерфейси между телекомуникационни системи и бизнес-приложения. В симулационната среда тези интерфейси ще се използват за:

- предаване на данни
- връщане на резултати
- достъп до базите данни от телекомуникационните услуги

при управлението на bl-транзакции, които включват както бизнес-услуги, така също и телекомуникационни услуги.

7 Заключение

Симулационната среда M-STEBS се реализира на Java. Различните функционални модули са разработени като компоненти, което дава по-големи възможности за стандартизация, гъвкаво използване и параметризиране.

До момента са проектирани и програмирани отделни части на средата. Предстои тяхното интегриране, с което ще завърши първия етап от изграждането на симулационната среда.

Някои от най-съществените разширения, които се предвиждат за втория етап са следните:

- Реализиране на пълната функционалност на бизнес-архитектурата
- Включване на Parlay разширение в проекта Ti-JEINA
- Реализиране на някои от основните стандартизирани интерфейси между телекомуникационната симулационна среда и многослойната бизнес-архитектура.

ЛИТЕРАТУРА

[1] А.Казанджиев, Архитектура на интелигентна телекомуникационна мрежа, Юбилейна научна сесия “30 години ФМИ”, 3-5 Ноември, Пловдив, (2000), (приета за печат)

[2] Е. Хаджиколев, Многослойна архитектура за бизнес-приложения, Юбилейна научна сесия “30 години ФМИ”, 3-5 Ноември, Пловдив, (2000), (приета за печат)

[3] Т. Георгиев, Един подход за интеграция на интелигентни мрежи и бизнес-приложения, Юбилейна научна сесия “30 години ФМИ”, 3-5 Ноември, Пловдив, (2000), (приета за печат)

[4] EBES/EWOS Project Team on Building Blocks for Electronic Commerce (PRB), Building Blocks for Electronic Commerce, Final Report, Brussels, (1997)
[5]IEEE Intelligent Network Worksshop IN'96, The Carlton Crest, Melbourn, Australia, April 21-24, (1996)
[6] S.Stojanov, M.Kumurddjieva. MASTT-Technology and Its Application in Electronic Commerce Systems, Workshop "Concurrebcy, Specification & Programming", Humboldt-University of Berlin, September 28-30, (1998)
[7] S.Stojanov, M.Kumurddjieva, E. Dimitrov, A. Schmietendorf, Technological Framework for Development of Agent-based Applications, Workshop "Concurrebcy, Specification & Programming", Humboldt-University of Berlin, October 9-12, (2000)
[8] H.D.Burkhard, Theoretische Grundlagen (in) der Verteilten Kuenstlichen Intelligenz, in: Verteilte Kuenstliche Intelligenz, BI, 157-189, (1993)
[9] Y.Labrou, T.Finin, A Proposal for a new KQML Specification, University of Maryland Baltimore County, (1997)
[10] http://logic.stanford.edu/kif/specification.html
[11] http://www.parlay.org
[12] http://java.sun.com/products/jain/index.html
[13] Parlay Group Press, Parlay API specification 2.0, (1999)
[14]D. Ayers, H. Bergsten, … Professional Java Server Programming, Wrox Press Ltd, (1999)
[15] Z.Pawlak, Rough Sets. Theoretical Aspects of Reasoning about Data. Institute of Computer Science, Warsaw, (1990)
[16] A. Kazandjiev, S. Stojanov, A program interpretation of the Rough Set Theory, Workshop "Concurrebcy, Specification & Programming", Humboldt-University of Berlin, October 9-12, (2000)
[17] M.Lenz, B. Rartsch-Spoerl, H.D. Burkhard, S.Wess. Case-Based Reasoning Technology. From Foundations to Applications. LNAI 1400, Springer-Verlag, (1998)

Станимир Стоянов
Пловдивски университет "П.Хилендарски"
Факултет по математика и информатика
Бул. България 236, Пловдив
e-mail: csstani@pu.acad.bg