ПЛОВДИВСКИ УНИВЕРСИТЕТ

„ПАИСИЙ ХИЛЕНДАРСКИ”

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Катедра „Компютърни системи”

Развойна и симулационна среда за DeLC

Автореферат

за присъждане на образователна и научна степен „доктор”
в област *4. Природни науки, математика и информатика*,
професионално направление *4.6 Информатика и компютърни науки*,
докторска програма *Информатика*

Докторант: Дамян Димитров Митев

Научен ръководител: акад. проф. дтн. Иван Попчев

Пловдив
2018

Дисертационният труд е обсъден и насрочен за защита пред научно жури на катедрен съвет на катедра „Компютърни системи“ при Факултета по математика и информатика на ПУ „Паисий Хилендарски“ на 27.04.2018 г.

Дисертационният труд съдържа 134 страници. Използваната литература включва 141 източника. Списъкът с авторските публикации по темата съдържа 5 заглавия, от които 3 на английски и 2 на български език.

Защитата на дисертационния труд ще се състои на 13.07.2018 г. от 13:00 часа в заседателната зала на Факултета по математика и информатика на ПУ „Паисий Хилендарски“, гр. Пловдив.

Материалите по защитата са на разположение на интересуващите се в секретариата на ФМИ, нова сграда на ПУ, каб. 330, всеки ден от 8:30 до 17:00 часа.

Автор: Дамян Димитров Митев

Заглавие: Развойна и симулационна среда за DeLC

Тираж: 50 бр.

Пловдив 2018 г.

СЪДЪРЖАНИЕ

ИНДЕКС НА ФИГУРИТЕ

ИНДЕКС НА ТАБЛИЦИТЕ

ОБЩА ХАРАКТЕРИСТИКА НА ДИСЕРТАЦИОННИЯ ТРУД

Използването на Интернет, за подпомагане образователния процес, се превърна в трайна тенденция в съвременните висши учебни заведения. Електронното обучение става съществен елемент в образователните стратегии на университетите. В световен мащаб експлозивно нараства търсенето на ефективно работещи платформи за електронно обучение.

В отговор на съвременните потребности за подпомагане на обучението посредством използване на съвременни информационни и комуникационни технологии, във Факултета по математика и информатика (ФМИ на Пловдивския университет „Паисий Хилендарски“ (ПУ) стартира проектът Distributed eLearning Center (DeLC). Целта на проекта е изграждане на инфраструктура за доставка на електронни образователни услуги и електронно учебно съдържание, разположени върху физически разделени сървъри [65] [68] [79] [69].

Концептуално DeLC е динамична мрежова инфраструктура, състояща се от възли [55] [59] [62] [74], върху които могат да бъдат разполагани електронни образователни услуги, хранилища за електронно съдържание и релации, които специфицират определени зависимости между възлите.

Спрямо предназначението си възлите могат да бъдат от различен тип, като напр.:

- *Образователни възли* – предоставящи електронни образователни услуги и електронно учебно съдържание посредством подходящ потребителски интерфейс;
- *Помощни възли* – прозрачни за потребителя възли, задачата на които е да подпомагат функционирането на образователните възли;
- *Специализирани възли* – предоставят специфични услуги за потребители със специален статус;
- *Интерфейсни възли* – предоставят специфични интерфейси за връзка с външни системи.

Спрямо достъпа до предлаганите информационни ресурси в модела на DeLC различаваме два вида образователни възли:

- *Възли с фиксиран достъп* – предназначени за доставка на услуги и учебно съдържание предимно през фиксирани комуникационни мрежи, без специална поддръжка на мобилен достъп. Потребителският интерфейс на тези възли се разработва под формата на образователен сайт или портал;
- *Възли с мобилен достъп* – предназначени за мобилна доставка на услуги и съдържание през InfoStation-базирани комуникационни мрежи посредством посредничеството на специализиран софтуер (мидълуер).

Възлите могат да оперират самостоятелно или динамично да се свързват помежду си, образувайки комплексни виртуални структури, наречени образователни клъстери. Понастоящем са разработени два клъстера.

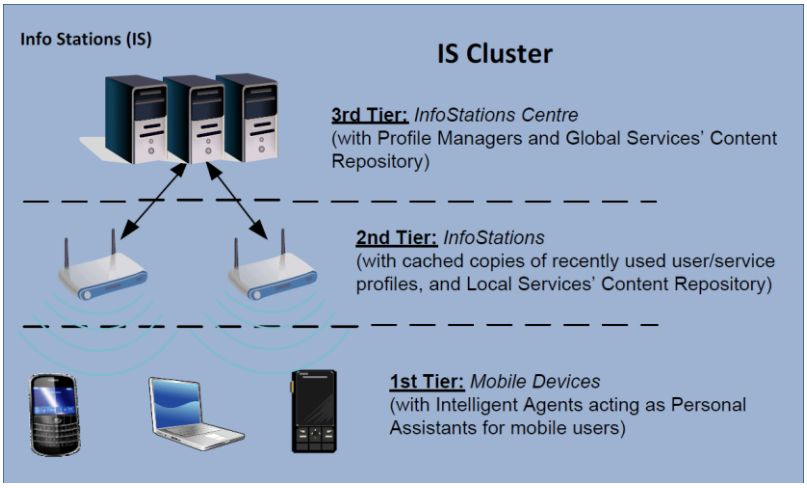
Първият, наречен MyDeLC, се използван за организация и провеждане на реално електронно обучение. Клъстера MyDeLC включва следните възли (Фигура 1):

- *DeLC портал* – типичен образователен възел, реализиран като портал, който предоставя фиксиран достъп до услугите и електронно учебно съдържание [82]. Доставката на електронни услуги и учебно съдържание се осигурява от три стандартизирани софтуерни модула - SCORM 2004 Engine, eTest Engine и Event Engine;

- *Agent Village (AV)* –възел, доставящ специализирана помощ на услугите, предоставяни от образователния портал на DeLC, под формата на „асистенти" [86] [67] [64];
- *e-Learning in Software Engineering (eLSE)* –възел, предоставящ средства за подпомагане на обучението по дисциплината „Софтуерни технологии" [85];
- *Focus* – интерфейсен възел, осъществяващ връзка с информационната система на университета [82].
- *Education Portal for Secondary School in Brezovo (EPB)* - образователен портал на СОУ "Хр. Смирненски", гр. Брезово [81].
- DeLC Test Center (DeTC) – клъстер с отворена инфраструктура, която предлага услуги за електронно тестване [84] [49] [50];
- CADeLC (Creativity Assitant) е специализиран възел за изучаване и идентифициране на креативното мислене и действие на обучаемите [80] ;

Фигура 1 Архитектура на MyDeLC клъстер

Вторият клъстер клъстер, наречен InfoStation клъстер [63], предоставя трислойна архитектура (Фигура 2), осигуряваща мобилен достъп до услуги и информационни ресурси посредством интелигентни безжични точки на достъп, наричани Information Stations, разположени около сградата на университета [22] [57] [61] [23]. Като комуникационна мрежа за този клъстер се използва InfoStation архитектура, състояща се от три слоя – InfoStation Center, InfoStations и мобилни устройства.



Фигура 2 Архитектура на InfoStation клъстер

Върху InfoStation Center са разположени образователни услуги и необходимите за тяхната обработка информационни ресурси. Различен брой InfoStation се използват като посредници между InfoStation Center и мобилните устройства на потребителите. В концепцията на DeLC ролята на InfoStations е разширена, като върху тях се разполагат също образователни услуги, които се изпълняват и управляват локално. Софтуерното осигуряване на мобилния клъстер се разработва като контекстно-зависим и адаптивен агентно-ориентиран мидълуер, който е в състояние да открива промените в средата и в зависимост от тях да се адаптира за ефективно изпълнение заявките на потребителите. За подпомагане тестването на мидълуера е разработена симулационна среда, в която неговото поведение може да се анализира в рамките на отделни експерименти [63].

Разширената концепция за използване на InfoStation мрежата поражда нови проблеми, свързани с идентификация и локализиране на събитията, предизвикващи промени в средата [78] [56]. Особено съществени стават идентификацията на събитията и синхронизацията на необходимите адаптивни активности във времето. Интервалната темпорална логика е подходящ формализъм за описание на времево-зависими процеси, за който съществува интерпретиращ механизъм – системата Tempura [1] [41].

Цел и задачи на дисертационния труд

Обобщавайки краткия анализ може да се направи следния извод: **необходими са нов тип софтуерни архитектури, развойни и поддържащи средства, които да предоставят достатъчна мощност за удовлетворяване на изискванията на системи за електронно обучение.**

В съответствие с направения извод, основната цел на дисертацията се формулира както следва: **изследване и прототипна реализация на развойна и симулационна среда на DeLC. Средата ще подпомогне изграждането на нова гъвкава и динамична версия на DeLC, с адаптивни и колаборативни възможности.**

За постигане на целта са дефинирани следните **четири задачи**:

1. Разработване на концепция, софтуерна архитектура и прототип на среда за създаване на електронно съдържание, удовлетворяващо стандарта SCORM 2004;
2. Реализиране на втора версия на SCORM машина – интерпретатор на правилата за последователност и навигация (SCORM Sequencing and Navigation Engine), удовлетворяваща сертифициращите изисквания на организацията ADL;
3. Разработване на концепция, софтуерна архитектура и прототип на симулационна среда за тестване на InfoStation-базирания и агентно-ориентирания специализиран мидълуер, поддържащ доставката на мобилни услуги в DeLC;
4. Реинженеринг на отделни компоненти на формалната среда Tempura за изграждане на нова, напълно обектно-ориентирана Java версия, която може да бъде вградена в DeLC.

Изследването, представено в дисертационния труд, е свързано с нашето участие в представения накратко международен проект под наименованието „Software Engineering: Computer Science Education and Research Cooperation“ [15].

Структура и обем на дисертацията

Предложеният дисертационен труд се състои от увод, пет глави и заключение. Обемът на основната част е 127 страници, а на приложената библиография – 7 страници

ивключва 141 източника. Списъкът с авторските публикации по темата съдържа 5 заглавия, от които 3 на английски и 2 на български език.

В първа глава са описани теоретичните основи и разработки, на които се базира настоящия труд. Обърнато е внимание на значението на стандарта за електронно обучение SCORM и са дискутирани възможностите за използване на агентно-ориентирани архитектури, онтологии и интервална темпорална логика за изграждане на персонализирана система за електронно обучение.

Във втора глава е предложена архитектура за изграждане на среда за разработване на SCORM съвместимо електронно съдържание, изброени са принципите, стоящи зад нейното създаване и е представена средата Selbo 2 като реализация на тази архитектура.

В трета глава е описана SCORM машина – интерпретатор на правилата за последователност и навигация на стандарта. Дискутирана е ролята на машината в DeLC портала, показана е нейната архитектура и е дадена статистика за проведените сертификационни тестове.

В четвърта глава е представена симулационната среда SimEnv. Нейната роля е да улесни тестването на агентно-ориентирания мидълуер на DeLC и поддържаните от него сценарии на взаимодействие между потребители и InfoStation възли, като симулира потребителски заявки и позволява проследяване на обработката на тези заявки на различни нива в мулти-агентната система и верифициране на коректността на изпълнението им.

В пета глава е описан подхода за реинженеринг на Tempura – интерпретатор на интервална темпорална логика (ITL). Представени са стъпките по анализ на системата, написана на езика C, и създаване на функционално еквивалентна версия на Java, с цел използване на ITL за верификация на процеси в DeLC.

В заключението са дадени резултатите от апробацията на SCORM машината и са обобщени резултатите от дисертацията, като се дават някои перспективни насоки за продължаване на изследванията по темата. Даден е и граф на дисертационния труд.

ГЛАВА I. ТЕОРЕТИЧНИ ОСНОВИ НА ДИСЕРТАЦИЯТА

Дисертацията се базира на следните подходи, технологии и стандарти:

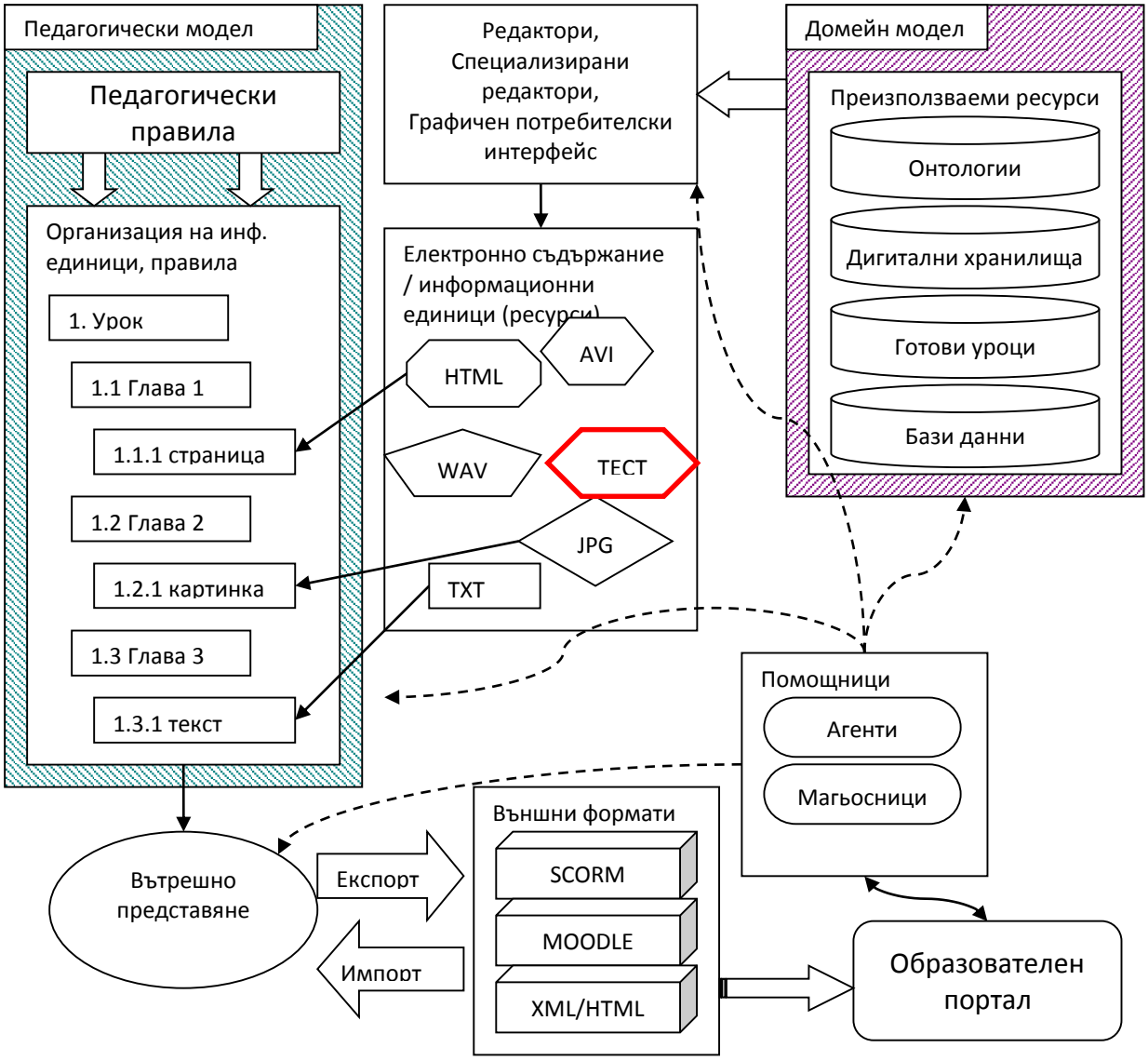
- SCORM (Sharable Content Object Reference Model) - референтен модел, създаден от агенцията ADL (Advanced Distributed Learning) [4] [3]. SCORM дефинира правила за създаване на повторно използваеми единици като отделя самата обучителна информация от правилата, по които тя ще бъде представена на обучаващия се [33].
- Агентно-ориентирант подход - представлява разработване на приложения, в които агенти реализират някои от основните функционалности на системата. Автономните агенти са системи, които могат да правят самостоятелни решения и да извършват действия върху средата за постигане на целите си [42].
- Семантичен Уеб - обща рамка, която позволява формалното описание на смисъла на описаната информация [10] [6] [9]. Семантичният Уеб се основава на стандартите Resource Description Framework (RDF) [51] и Web Ontology Language (OWL) [46].

- Архитектури, ориентирани към услуги (SOA) - предоставят интерфейс към функционалността на дадено приложение, използвайки стандартни интернет протоколи, които са независими от използваните платформи и езици за програмиране [76].
- Интервална темпорална логика (ITL) - гъвкава нотация, както за съждителната логика, така и за логиката от първи ред за моделиране времево-зависими процеси. *Tempura* е интерпретатор на едно изпълнимо подмножество на ITL.

ГЛАВА II. РАЗВОЙНА СРЕДА ЗА РАЗРАБОТВАНЕ НА SCORM СЪВМЕСТИМО ЕЛЕКТРОННО СЪДЪРЖАНИЕ

2.1 Архитектурен модел на развойна среда за електронно обучение

Архитектурният модел отразява нашето виждане за това какви характеристики трябва да притежава една развойна среда за електронно съдържание.



Фигура 3 Архитектурен модел на развойна среда за електронно обучение

Принципната архитектура на развойна среда за електронно обучение, показана на Фигура 3, включва следните компоненти:

- *Средство за организиране на информационните единици* в последователност, превръщащи съвкупността им в електронен урок. Върху създадената структура се дефинират педагогическите изискванията и се задават правилата за последователност на обхождане и навигация между ресурсите;
- *Редактори на основните информационни единици*, изграждащи електронния урок. Те могат да бъдат редактори на метаданни, на стандартни файлови формати, на специализирани редактори, имащи смисъл в определен домейн, и редактори на интерактивно съдържание;
- *Средство за експортиране* на вътрешното представяне на структурите от данни във външни формати.
- *Средство за импортиране* на готови уроци от стандартни външни формати.
- *Средства, позволяващи задаването на шаблони* с предварително зададени педагогически цели;
- *Средства, предлагащи предварително създадени ресурси* за използване наготово;
- *Помощници*, автоматизиращи създаването на електронния урок.

2.2 Развойна среда Selbo 2

Обект на настоящата глава е разработването на прототип на развойна среда за създаване на електронни уроци Selbo и адаптирането ѝ за дисциплината „Софтуерни технологии“. Средата поддържа шаблони с дефинирани педагогически цели, които съдържат в себе си и набора от SCORM правила за постигането на тези цели. Предвидени специални онтологии съдържащи концепции, термини, примери към тях и примерни уроци, които могат да бъдат използвани наготово. В средата са изградени агенти, които да асистират на преподавателя в изготвянето на самия урок.

Целите на средата са да създава на стандартни SCORM 2004 съвместими пакети с електронно съдържание, да бъде лесна за използване от хора, които не са професионалисти в компютърните технологии, потребителите ѝ да работят в термините на своята приложна област, която познават и разбират добре, без да има нужда да познават стандарта SCORM, HTML или други от наложените технически стандарти, да е лесно разширяема с нови функционалности, редактори и компоненти на базата на система за управление на приставки.

При разработката на Selbo 2 бе изградена концепцията за интелигентни компоненти.

2.3 Принципи при разработката на средата Selbo 2

Няколко основни принципа са залегнали при изграждането на средата Selbo 2.

Цялостна среда за SCORM съдържание

Selbo 2 се стреми да затвори цикъла на създаването на електронно съдържание, като предоставя средства за манипулиране на организацията на ресурсите и SCORM правилата за последователност и навигация (SCORM sequencing and navigation), позволява редактирането на самите ресурси чрез набор от вградени в средата

редактори и съдържа материали от различни дисциплини, които могат да се използват при създаване или редактиране на ресурси.

Персонализирано електронно съдържание

За организирането на персонализирано електронно съдържание в проекта DeLC се предлага ясно разграничение и изрично представяне на три вида знания - специализирани знания по изучаваната дисциплина, знания за използвания педагогически подход и знания, характеризиращи индивидуалните предпочитания на учащите и учителите. В предложената инфраструктура на DeLC тези знания се представят и управляват като три независими модела: потребителски модел, педагогически модел и домейн модел [66].

Домейн модела описва предметната област или дисциплината, за която се разработва електронно съдържание. Педагогическият модел включва в себе си елементите от системата, които отговарят за задаване на педагогически цели, които трябва да бъдат удовлетворени от обучаемия при използване на електронната лекция. В рамките на DeLC под педагогически цели се разбира набор от логически правила, които обучителната система спазва при взаимодействието си с обучаемите. Един елемент от средата, поддържащ този модел, са педагогическите шаблони [81]. Потребителският модел е свързан с персонализацията на системата към своите потребители и средствата, които ги подпомагат при работата им с нея. Този модел се подържа и от интелигентните помощници в средата - агенти и магьосници (wizards).

Интелигентно поведение на средата

В контекста на средата Selbo 2 интелигентността е термин, обхващащ поведението на множество компоненти, което цели да автоматизира или улесни автора при създаването на образователните единици. Една от функционалностите на системата е наличието на проверки за консистентността на данните, записвани в структурата на дървото на съдържанието на урока. Друга функционалност е абстракцията над SCORM, която представлява опростен модел на дървото на съдържанието, в който средата автоматично управлява метаданните на възлите от организацията.

Колаборативност на средата

Архитектурния модел предвижда колаборативността между средата и автора на съдържание да се извършва с помощници, които биват два вида: *магьосници* и *агенти*. *Магьосниците* са пасивни помощници - те се извикват изрично от потребителя, и не са активни по време на разработката на самия урок. *Агентите* са непрекъснато активни помощници. Те притежават цели, които се стремят да постигнат и работят в автономен режим на работа, като непрекъснато взаимодействат с останалите компоненти в системата [7]. Предвижда се също и колаборация със системата за обучение. Тя може да пази детайлна информация за даден урок. Тази информация може да бъде използвана от автора за да рафинира и подобри своите електронни уроци.

Интерактивност

Агентите взаимодействат с останалите компоненти в системата, комуникират един с друг, вземат решения и на базата на тези решения да правят промени върху структурата на урока. Тези промени задължително трябва да са одобрени от потребителя. Агентите трябва да представят плана си за промяна в диалогов режим с

потребителя, като му предоставят възможност да модифицира предложените промени или да се откаже от тях.

Модулност и адаптивност

Принципите за адаптивност и модулност са много близки един на друг, като различията между тях са логическото ниво, на което са дефинирани. *Модулността* се определя на техническото ниво на изграждане на системата. Тя дефинира наличието на plug-in система в средата, чрез която да могат лесно да се добавят нови компоненти. *Адаптивността* разглежда логическата структура на средата и позволява лесното дефиниране нов домейн и добавянето нужните за домейна редактори, онтологии, помощници, педагогически шаблонни.

Интуитивна работа със средата

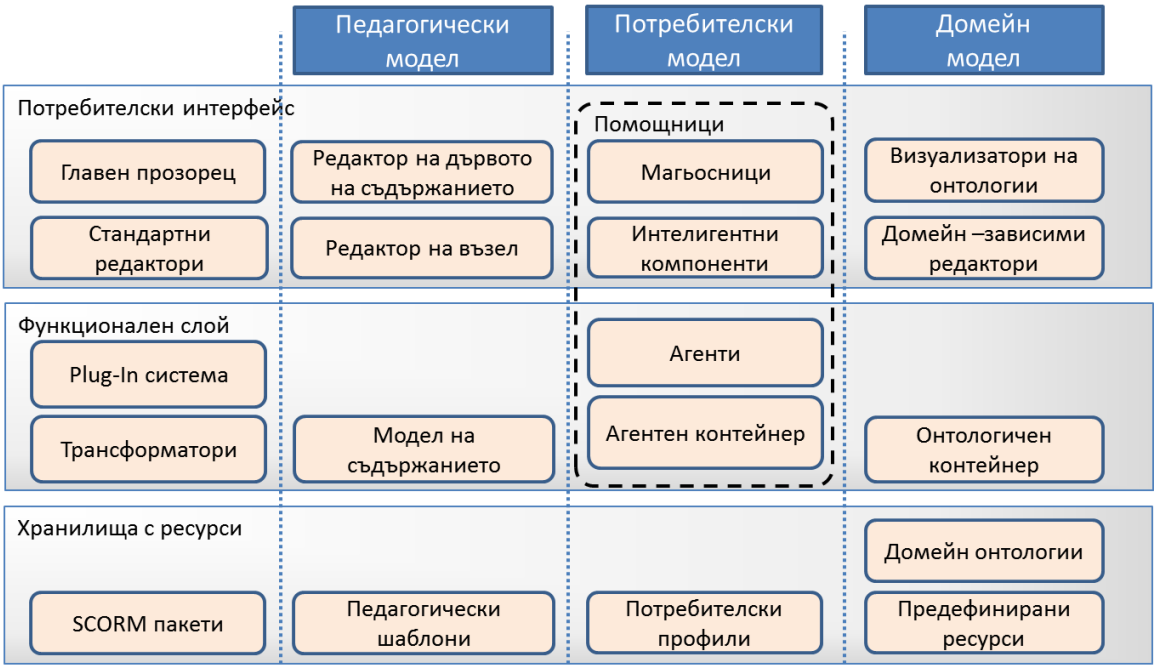
Системата предоставя стандартен и естетически издържан графичен потребителски интерфейс, удобно разположение на компонентите и възможност за тяхното местене и оразмеряване. Също така реализира адекватна и атрактивна визуализация на помощниците.

2.4 Технологии и стандарти, използвани в Selbo 2

Средата Selbo 2 се разработва на езика за програмиране Java. За изграждане на структурата на Selbo 2 бе избрана популярната OSGi [45] базирана платформа Eclipse [18]. Ядрото на Eclipse представлява plug-in системата Equinox [17], чрез която се интегрират различни компоненти. Потребителски интерфейс на средата е изграден с помощта на графичната библиотека SWT [19]. Като средство за реализация и достъп до онтологиите е избран редактора на онтологии Protégé [58]. За реализиране на агентите в Selbo 2 бе избрана платформата JADE (Java Agent DEvelopment Framework) [70].

2.5 Архитектура на Selbo 2

Архитектурата на развойната среда е представена на Фигура 4.



Фигура 4 Архитектура на Selbo 2

Система за управление на приставки

Средата не е едно монолитно цяло, а се състои от отделни модули, всеки от които реализира отделен компонент. Системата има грижата за управлението и зареждането на компонентите и осигуряване на общ канал за комуникация между тях.

Главен прозорец

Главният прозорец съдържа в себе си всички останали визуални компоненти на средата. Той управлява жизнения цикъл на цялата среда.

Модел на съдържанието

Структурата, която съдържа организацията на учебния материал, е обособена в самостоятелен модул. Моделът има дървовидна форма, като отразява структурата на манифестния файл на SCORM. Върху тази структура се задават правилата на последователност и навигация. Визуализацията ѝ е обект на редактора на дървото на съдържанието.

Редактор на дървото на съдържанието

Този модул служи за визуализиране на дървовидната структура на урока. Също така чрез него могат да се добавят нови, да се изтриват съществуващи или да се местят възли. Този графичен компонент е винаги синхронизиран с модела на дървото на съдържанието.

Редактор на възел от дървото

Този компонент има за цел да редактира съдържанието на конкретен възел от дървото. Върху възлите са дефинирани метаданните, които управляват поведението на системата за обучение при интерпретиране на пакета.

Шаблони с педагогически правила

Този компонент представлява хранилище на шаблони, които дефинират уроци с различни педагогически цели. Такива шаблони съдържат в себе си препоръчителна структура на урока, както и набор от правила за последователност и навигация, които да се приложат към тази структура.

Редактори на ресурси

Редакторите на ресурси са основните единици, изграждащи потребителския интерфейс. Тяхната задача е да редактират ресурсния файл, свързан с даден възел от дървото. Основния ресурс, генериран от средата, е HTML страница, поради богатите възможности за текстово форматиране и вграждане на мултимедия. Selbo 2 се възползва и от включените в Eclipse редактори за стандартни формати. За целите на средата се разработва и домейн-ориентиран редактор на UML диаграми за дисциплината „Софтуерни технологии“.

Визуализатор на онтология

Визуализаторът на онтологията има задачата да представи на потребителя съдържанието на онтологията, заредена в модула с помощта на библиотеката на Protégé. Компонента се състои от две части: Визуализатор на модела на онтологията и

визуализатор на конкретен екземпляр от нея. Също така предоставя и магьосник, чрез който учителя може да задава критерии за търсене на информация в онтологията.

Трансформатор към SCORM

Този модул има задачата да трансформира вътрешното представяне на дървото на съдържанието към структурите на стандарта SCORM

Хранилища с предефинирани ресурси

При разработката на средата бе разгледана възможността за комуникация с външни хранилища с предварително генерирани ресурси или готови SCORM пакети. В прототипната версия на средата е разработен частен интерфейс за достъп до хранилището на SCORM съдържание в разработвания за нуждите на DeLC портал [82].

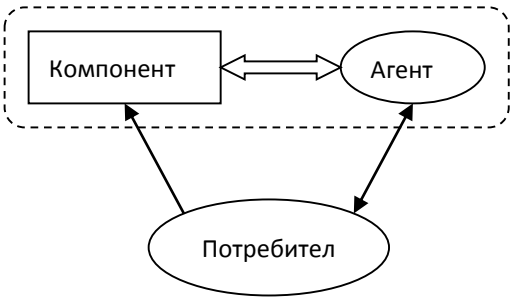
Потребителски профили

Selbo 2 позволява многопотребителски достъп. За всеки потребител се създава профил, който отразява личните му предпочитания за изглед и функционалност на средата..

2.6 Интелигентни компоненти

Различните редактори, които представляват интерфейса на средата, са визуални компоненти. Те могат успешно да изграждат потребителския интерфейс, но им липсва интелигентност. От друга страна агентите са проактивни, могат да извършват автономни действия и имат специфична архитектура на самоуправление, което ги прави неудобни за директното им вграждане в графичния потребителски интерфейс.

Нашият подход за решаването на гореспоменатите проблеми се основава на използването на интелигентните компоненти (Фигура 5). Виждането ни за интелигентен компонент представлява комбинация от компонент на визуалния интерфейс към който е прикрепен агент, специално създаден да взаимодейства с компонента. Комбинацията представлява слаба свързаност и силна съгласуваност между две самостоятелни единици, така че единия елемент от двойката може да бъде заменен лесно, без да се променя другия.



Фигура 5 Интелигентен компонент

Жизненият цикъл на компонента се управлява от се управлява от единствена нишка на потребителския интерфейс (GUI thread). Други нишки нямат право на директен достъп и единствения позволен начин за комуникация е чрез изпращане на съобщения в опашката на даден компонент. Всеки агент в JADE платформата се изпълнява в собствена нишка.

За успешната комуникация между компонента и агента трябва да се решат два проблема: предаване на съобщения от агента към компонента и обратно. Както видяхме и двете единици се управляват от различни нишки, което създава нуждата от синхронизация между тях. Един от проблемите е, че нишката на потребителския интерфейс отговаря за всички компоненти в приложението и обработката на събитията се извършва в нейния контекст. За това обработката на всички съобщения трябва да става за минимално време. Това означава, че ако агент иска да направи комплексна промяна в състоянието на компонент, тя трябва да е разбита на атомарни съобщения, съдържащи в себе си елементарни заявки към компонента. Основен недостатък на тази стратегия е невъзможността за гарантиране на промяна в компонента в реално време – ако опашката със съобщения на нишката-диспечер е пълна или заета с обработката на други събития, промяната може да бъде отразена след няколко секунди. За постигане на диалог в реално време, се приема забавяне от не повече от секунда [58].

Комуникацията между компонента и агента също има своите предизвикателства. Компонентите са реактивни - те реагират на действията на потребителя, като генерират събития. Когато агент иска да получава съобщения за настъпила промяна в компонент, той трябва да се регистрира като получател на съответните събития. Независимо от това, че агента има собствена нишка на управление, обработчиците на събития се извиква в контекста на основната нишка на приложението. Освен това е възможно едно логическо действие на потребителя да генерира множество (еднакви или различни) събития. За управлението им в агента трябва да се реализира машина на състоянието. Сложността на тази машина зависи от вида на свързания компонент и от желаното прихващане на действия на потребителя.

2.7 Домейн-зависими онтологии

Заради възможностите да формализират семантиката, свързана с различни обекти и отношения между тях, онтологиите са едно отлично средство за съхранение на многократно използваеми ресурси от знания. За нуждите на средата бе разработена онтология, съдържаща концепции, дефиниции и описания на процеси от дисциплината „Софтуерни технологии”. Моделът на онтологията може да се раздели на три групи класове: поддържащи класове, основна йерархия и домейн-зависима йерархия

Поддържащи класове моделират стандартни концепции, които са извън йерархията на конкретния домейн. Дефиницията им включва и тяхната семантика – данните, които съдържат, имат ясна структура и значението им остава непроменено при използването им в различни контексти. В предложената онтология са разработени три поддържащи класа: *Memo*, *Example* и Image.

Основната йерархия от класове е независима от конкретната приложна област. Тези класове определят базовата семантика на класовете в онтологията. Метакласът *OntologyClass* дава необходимата информация за всички класове, които описват онтологията за съответната учебна дисциплина. Класът *OntologyObject* стои в основата на онтологиите. Той е инстанция на метакласа OntologyClass и е родителски клас за всички класове, изграждащи специфичните онтологии за различните домейни.

Домейн-зависимите класове представляват конкретизация на основната йерархия и са специфични за дадена предметна област. Техните атрибути има стандартни типове с добре дефинирано представяне, което позволява на парсер да извлече информация от онтологията, без предварително да е запознат със структурата й. Онтологията за дисциплината „Софтуерни технологии“ включва в себе си четири

основни класа. Тези класове наследяват класа OntologyObject и представят основните термини, които се използват в домейна: *Phase (фаза)* – описва етапите (фазите), през които преминава разработката на софтуерния продукт, *Role (роля)* – описва участниците в разработката, *SystemDocument (системен документ)* – описва документацията, *SoftwareConception (софтуерна концепция)* – описва концепциите и използваните нотации

ГЛАВА III. ИНТЕРПРЕТАТОР НА ПРАВИЛАТА ЗА ПОСЛЕДОВАТЕЛНОСТ И НАВИГАЦИЯ В SCORM

Използването на платформи за електронно обучение и системи за управление на обучението (LMS) и проектирането на онлайн курсове със собствени образователни технологии значително се увеличиха в образованието и бизнес обучението. Учебни единици, комбинирани с LMS, са използвани за изразяване на знания и предоставяне на информация и образователни услуги. Дейността на инструктор или преподавател е от съществено значение за ръководеното обучение: учителят избира проблемите, които да покаже на обучаемия, определя как да анализира отговорите, представя решаването на определени проблеми или решава да покаже някои примери. Учителят също така управлява учебните материали и отговаря за избора на най-подходящия материал в зависимост от съответната ситуация.

Всичко това може да бъде включено в модел, подобен на предлагания от Shareable Content Object Reference Model (SCORM), за да се покаже материалът в оптимална структура, определена от инструктора. SCORM е най-широко използваният стандарт за разработване на курсове за електронно обучение. Този стандарт дефинира независими информационни единици, които могат да се стартират на всяка SCORM съвместима LMS платформа [3].

3.1 Модели на SCORM

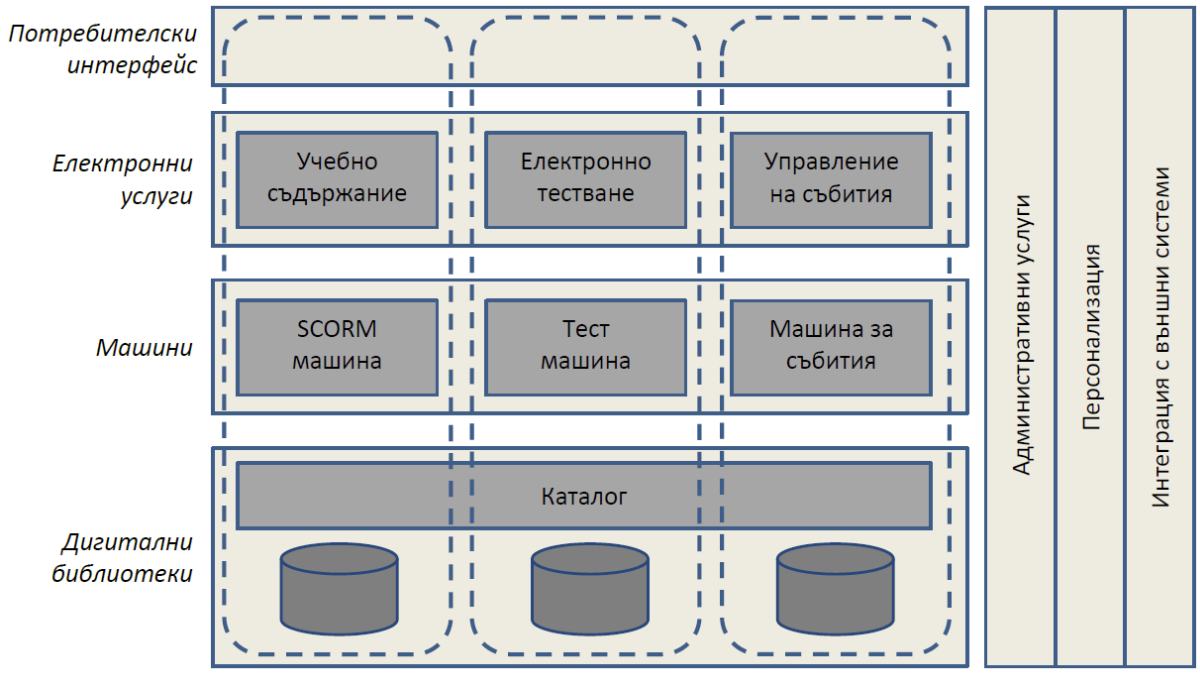
SCORM помага да се дефинира техническата база за онлайн среда за обучение [3]. Референтният модел съдържа правила за последователност и навигационна за динамично представяне на учебното съдържание. SCORM последователността предоставя на разработчиците на курсове за електронно обучение необходимите инструменти за създаване на сложни проекти, които могат да бъдат адаптирани към индивидуалните нужди на обучаемите. В стандарта се поддържат следните модели: Модел на проследяване (Tracking Model), Модел на възли (Activity State Model), Модел на последователността (Sequencing Definition Model), Навигационен модел (Navigation Model) [2] [54].

Моделите са съвкупности от правила и обекти, върху които тези правила са дефинирани. Тези правила обуславят поведението на една SCORM съвместима система за обучение в различни аспекти от взаимодействието на обучаемия със системата. Тези правила са дефинирани върху възлите от дървото на съдържанието. Дървото на съдържанието представлява йерархична структура, в която всеки възел може да е или терминален (листо) или да съдържа в себе си други възли (клъстер). Върху възлите се дефинират структури от данни, които отразяват текущото състояние на моделите. Стойностите на елементите на тези структури определят набор от възможните действия обучаемия, а промяната на тези стойности обуславя динамичното поведение на системата.

3.2 DeLC портал

Клъстерът MyDeLC, представен в уводната част (Фигура 1), представлява съвкупност от няколко взаимосвързани възли от архитектурата на DeLC. Образователният портал на DeLC е най-съществения компонент от клъстера, защото болшинството образователни функции се предоставят от него. Функционалностите на портала могат да се разглеждат в съответствие с основните групи потребители: обучаващи, обучаеми и администратори. Архитектурата на портала е матрична, многослойна и сървисно-ориентирана, състояща се от три хоризонтални слоя: потребителски интерфейс, услуги и дигитални библиотеки (Фигура 6). Порталът поддържа две големи групи услуги:

- Образователни услуги (електронни услуги);
- Системни услуги (машини).



Фигура 6 Архитектура на DeLC портал

Образователните услуги се предоставят за изпълнение на заявки на потребителите. В портала тези услуги са свързани обикновено с обслужване на образователния процес.

Системните услуги са прозрачни за потребителите и са предназначени за подпомагане изпълнението на образователните услуги. Тези услуги се наричат също машини (engines). Машините са вградени в архитектурата на портала като неразделна чат на неговата run-time система. В актуалната версия на портала са реализирани следните машини:

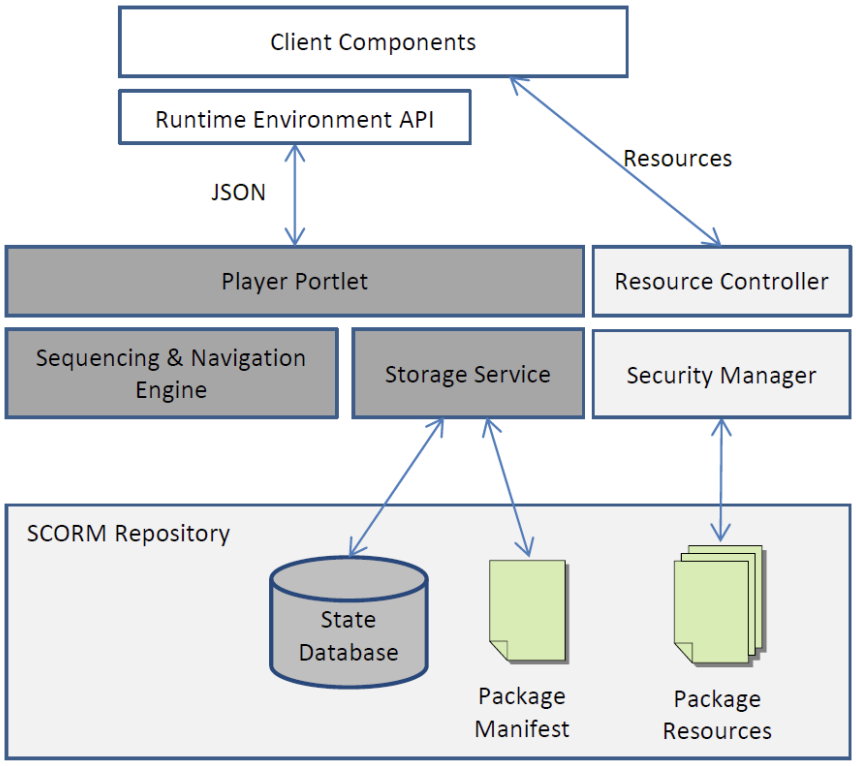
- SCORM Sequencing and Navigation Engine (SCORM машина);
- Test Engine;
- Event Engine;

3.3 SCORM Player

SCORM Player предоставя услуга за визуализиране на SCORM съвместимо електронно съдържание. Тази услуга се причислява към образователните, като при

изпълнение на функциите се обръща към системните услуги в портала, и по-специално към SCORM машината. SCORM Player в DeLC е изграден съгласно спецификацията SCORM 2004 4th ed. и се състои от два вида портлети:

- Портлети за управление на съдържанието
- Портлет за доставяне на SCORM съдържание до обучаемите (същинската част на SCORM Player)



Фигура 7 Архитектура на SCORM Player

Портлета за доставяне на SCORM съдържание до обучаемите (т.нар. Player portlet) реализира основната функционалност услугата (Фигура 7). Основен елемент в този портлет е SCORM машината (Sequencing and Navigation Engine), който извършва интерпретацията на правилата дефинирани в съответния SCORM пакет и на базата на поведението на обучаемия определя конкретното съдържание, което трябва да му бъде доставено във всеки един момент.

3.4 SCORM машина

Този модул е ядрото на DeLC SCORM Player. Реализиран е като отделна библиотека – интерпретатор на правилата на последователност на SCORM.

Алгоритмите, заложени в интерпретатора, се базират на псевдо-кода, описан в раздела Sequencing and Navigation Model на SCORM стандарта [54]. Псевдо-кода описва по езиково-независим начин логиката и поведението на системата и е разделен на над 40 псевдо-функции, групирани според функционалните си поведения.

Разработката на SCORM машината се извърши в четири фази. В първата фаза се реализира превод на функционалността на псевдо-кода на езика Groovy. За всяка псевдо-функция се направи анализ на структурите данни и се създадоха съответни класове, които да ги представляват. След това се направи превод на операторите от псевдо-функциите във функционално еквивалентни конструкции на Groovy. Всяка

псевдо-функция бе имплементирана като метод на основния клас на машината SNProcess.

Втората фаза се състои от оптимизирането на получения код. Бяха анализирани повтарящи се конструкции и бе направен рефакторинг на получената имплементация с цел получаване на обектно-ориентиран и структурно и смислено разделен код.

В третата фаза бе извършена интеграция на машината със SCORM Player. Бяха уточнени структурите от данни и механизмите за прехвърляне на данни между двете системи. Това включва набор от интерфейси, чиито имплементации реализират услугите по съхранение на текущото състояние на дървото на съдържанието за конкретен потребител в база данни и услугите на потребителския интерфейс за получаване на навигационни заявки и определяне на следващ ресурс за визуализиране. В тази фаза бяха създадени и среди за ръчно и програмно тестване на получената имплементация.

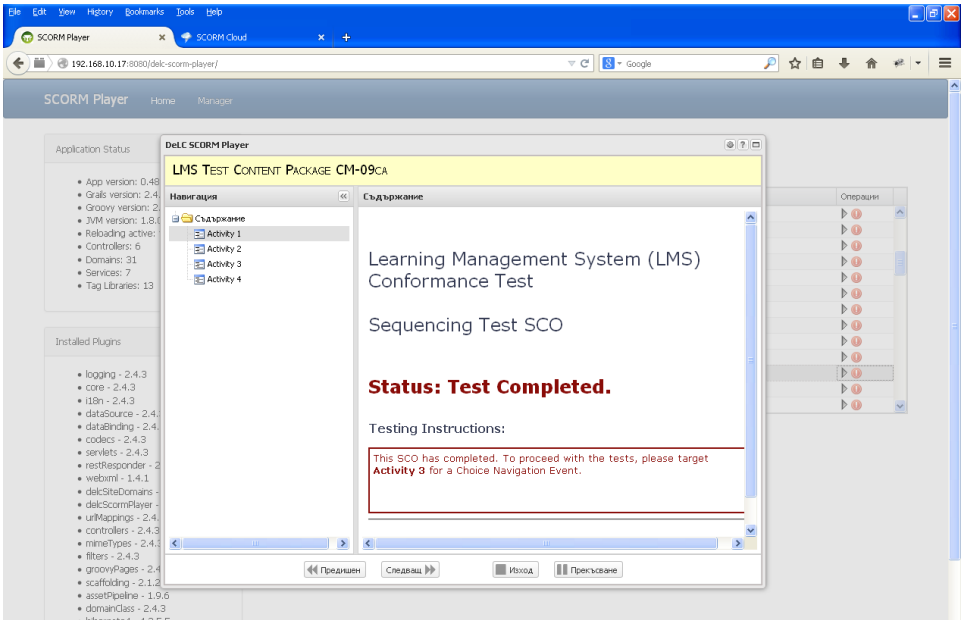
Четвъртата фаза представлява тестването на интерпретатора, чрез създадените тестови среди и чрез ADL SCORM 2004 TestSuite.

3.5 Тест на машината

Машината е реализирана като самостоятелна библиотека. По време на имплементацията на машината бяха разработени Stub-среди за ръчно и програмно тестване. Средата за програмно тестване представлява много тънка обвивка около машината и позволява изпращане на серия от навигационни заявки и запис на върнатите отговори в журнален файл с цел последващ анализ. Средата се използва основно за отстраняване на грешки в debug режим. Средата за ръчно тестване представлява елементарен потребителски интерфейс, който визуализира дървото на съдържанието и позволява на потребителя да избере възел от него (генерирайки choice navigation request) и предоставя контроли за избор на предишен/следващ възел.

3.6 Проведени тестове с ADL SCORM 2004 TestSuite

ADL, организацията, стояща зад разработката на референтния модел, е създала набор от полу-автоматизирани тестове, с които може да се верифицира всяка SCORM съвместима система за електронно обучение. Текущата версия на набора от тестове е



Фигура 8 Тестов пакет в SCORM Player

SCORM 2004 4th ed. Conformance TestSuite v1.1.1. В набора се съдържат близо 200 тестови пакета, групирани по категории.

Пример за стартиране на тестов пакет е даден на Фигура 8.

При тестването на нашата реализация на SCORM машина не се разчиташе само на генерираните съобщения от тестовата система. Поведението на DeLC SCORM Player бе сравнено с поведението на SCORM Cloud, като едни и същи тестови пакети бяха инсталирани и тествани и на двата плеъра.

SCORM Cloud (известен и като Online SCORM Player) [53] е web-базиран плеър на SCORM съвместимо електронно съдържание. Използваме SCORM Cloud като база за сравнение, защото той е сертифициран от ADL и поведението му е гарантирано коректно и генерира подробен журнал при изпълнението си.

Резултатите от проведените тестове могат да се обобщят в Таблица 1.

Група	**Всички**	**Успешни**	**Неуспешни**	**Оставащи**
API	1	0	0	1
CM	32	22	7	3
CO	23	22	1	0
CT	7	7	0	0
DDM	4	0	4	0
MS	8	6	2	0
OB	40	6	0	34
RU	48	0	0	48
SX	24	7	6	11
T	2	2	0	0
Общо	**189**	**72**	**20**	**97**

Таблица 1 Резултати от ADL SCORM 2004 TestSuite

От всички 189 теста в тестовия пакет, успешно са стартирани 92, което е 49% от общия брой. От тях 72 (или 78%) са приключили успешно работата си, в 20 е било открито несъответствие между очакваното и реалното поведение DeLC SCORM Player.

ГЛАВА IV. СИМУЛАЦИОННА СРЕДА ЗА ТЕСТВАНЕ НА INFOSTATION-БАЗИРАНИЯ МИДЪЛУЕР НА DELC

4.1 Сценарии в InfoStation архитектура

InfoStation архитектурата предполага различни сценарии за изпълнение на потребителските заявки, поради поддръжката на два вида мобилност - мобилност на потребителя (промяна на обслужващия InfoStation) и мобилност на устройството (промяна на крайното устройство). В контекста на двата вида мобилност са възможни следните базови сценарии:

- Без промяна – мобилното устройство и InfoStation не се променят, т.е. потребителя е неподвижен;
- Промяна на InfoStation – потребителя излиза от обхвата на първоначално обслужващия InfoStation и влиза в обхвата на друг InfoStation;

- Промяна на мобилното устройство – потребителя сменя мобилното си устройство по време на сесията;
- Промяна на InfoStation и мобилното устройство – мобилното устройство и обслужващия InfoStation се сменят по време на сесията.

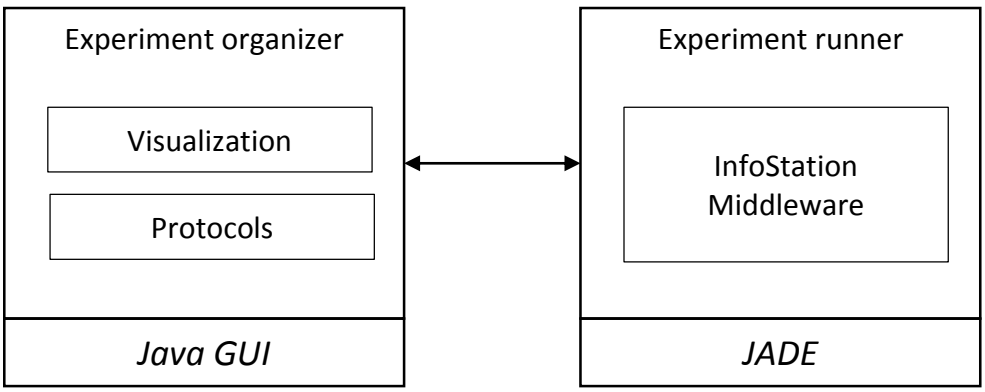
4.2 Симулация на изпълнението на сценариите в InfoStation архитектурата

Мобилността на устройството на потребителя предполага неговото влизане във и излизане извън обхвата на различни InfoStations, докато консумира образователна услуга. Тази промяна на местоположението изисква синхронизиране на сесията на потребителя между различните InfoStations. Като интелигентни и контекстно информирани, InfoStations реагират на тази промяна на местоположението с набор от поведенчески протоколи, общо известни като сценарии. Целите на сценария са синхронизиране на потребителската сесия, пренасочване на отговор на услугата и осигуряване на прозрачно доставяне на поисканите ресурси.

Разработването и тестването на интелигентния агентно-ориентиран мидълуер на реален хардуер е финансово неефективно, затова се разработи прототип на симулационна среда SimEnv, с която да бъдат тествани функционалностите на мидълуера.

Модули на симулационната среда

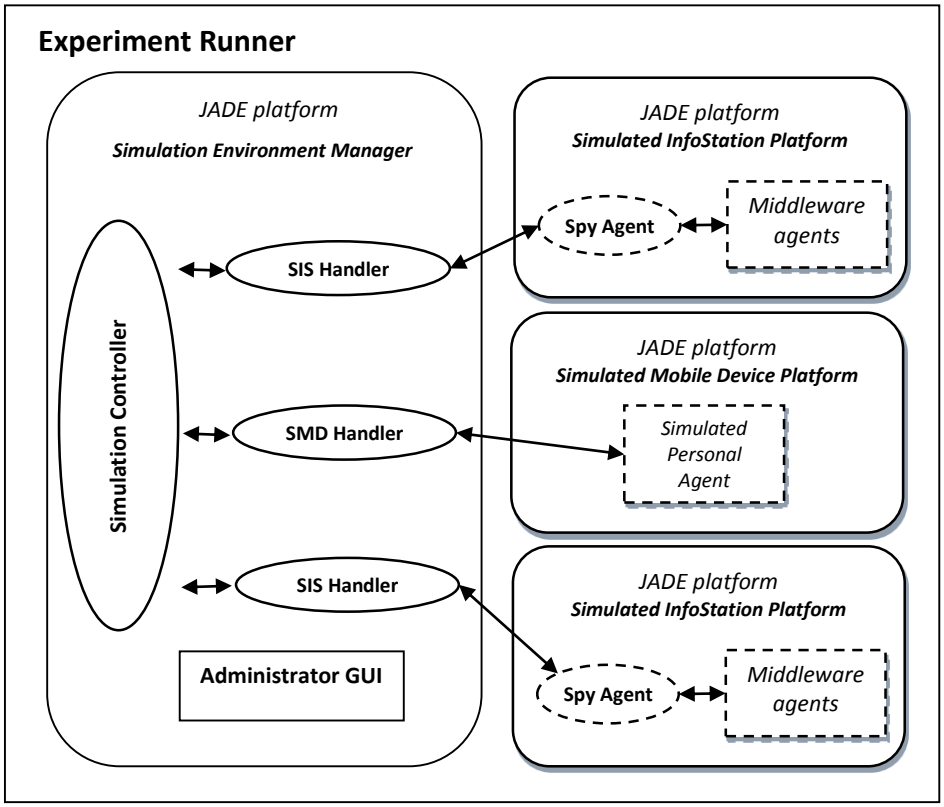
Симулационната среда включва два основни модула: Experiment Organizer (EO) и Experiment Runner (ER) (Фигура 9). Experiment Organizer представлява специализиран потребителски интерфейс, използван за подготовка на експерименти и представяне на резултатите. Експериментите са представени като спецификация (скриптове); резултатите се записват и се предоставят за последващ анализ. ER е агентно-ориентирана обвивка на тествания мидълуер, който контролира изпълнението на експеримента в съответствие с дадената спецификация. Междинните резултати, получени по време на конкретен експеримент, се предават на EO, където може да се визуализира изпълнението на експеримента. Симулационната среда може да представя всички събития, настъпващи в мидълуера, под формата на лог файл.



Фигура 9 Модули на симулационната среда

Experiment Runner

По време на изпълнението на експеримент, всеки InfoStation е представен като немодифицирано копие на мидълуера, който се изпълнява в отделен агентен контейнер (Фигура 10). Симулацията се управлява от набор от агенти, наричани контролери на симулацията, които се намират в отделен специализиран контейнер. Контролерите управляват средата на InfoStation, като имплантират специален Spy агент във всеки мидълуер контейнер. Тези Spy агенти получават команди за манипулиране на сензорите на мидълуерските агенти (ако е необходимо) и предоставят пълна обратна връзка на симулационните контролери за това, което се случва в тяхната InfoStation. Тестовете могат да се управляват от оператора, като се използва графичен интерфейс или могат да се задават като скриптове.



Фигура 10 Архитектура на Experiment Runner

Experiment Runner се състои от следните агенти:

- *Симулационен контролер (SC)* - централният координатор на експеримента, който се изпълнява. С негова помощ се симулира желаната конфигурация на InfoStation мрежата;
- *SIS Handler (SH)* - контролира отделна Simulated InfoStation (SIS) платформа;
- *Spy Agent* - изпълнява командите, подадени от симулационния контролер към симулираната IS платформата и осигурява обратна връзка със симулационната среда за събитията, настъпили в SIS.
- *SMD Handler* - управлява симулираното мобилно устройство;
- *Симулиран персонален агент* - играе ролята на личен агент на потребителя.

Агентите на ER са разположени на следните платформи:

- Системата за управление на симулационната среда *Simulation Environment Manager (SEM)* е JADE контейнер, на който са разположени симулационния контролер и Handler агентите. Тя изпълнява ролята на централна конзола за контрол на експериментите;
- *Симулирана InfoStation платформа (SIS)* е JADE контейнер, на който са разположени немодифицираните агенти на стандартния мидълуер. Единственото допълнение е наличието на Spy агента, който предоставя обратна връзка към контролера;
- *Платформата за симулирани мобилни устройства (SMD)* е мулти-агентен контейнер, на който е разположен симулирания личен асистент. Тази платформа представлява мобилното устройство на един потребител.

Spy Агент

Ролята на Spy агента е да открива промените в мулти-агентната платформа и да изпраща съответната информация на съответния SIS Handler в средата на симулацията. Spy агент могже да се абонира за всички събития, които се случват в една JADE платформа или може да бъде конфигуриран така, че да прихваща промените в състоянието само за определена група агенти. Информацията, събрана от Spy агента, може лесно да бъде филтрирана за последващи визуализация и анализ.

4.3 Нива на анализ на симулацията

Експериментите, извършени с помощта на тази среда, позволяват изследване и анализ на поведението на мидълуера на четири нива:

- Сценарийно ниво - На това ниво се проследява глобалното поведение на мидълуера по време на изпълнението на сценария, определен в експеримента. Фокусът е върху поведението на местните агенти и процеса на генериране на оперативни (временни) агенти, които зависят от промените в средата;
- Контейнерно ниво - на това ниво за всеки симулиран компонент (InfoStation Center, InfoStation или мобилно устройство) симулационната среда показва контейнерите, които са активни в мидълуера [70];
- Агентно ниво - на това ниво симулационната среда показва кои агенти живеят в контейнерите на всеки InfoStation. На агентно ниво се обръща особено внимание на подробното представяне на състоянията на агентите. Това ниво събира и информация, която позволява анализ на взаимодействията между агентите;
- Поведенческо ниво - целта на това ниво е да осигури възможност за анализ и оценка на локалното поведение на отделните агенти;

ГЛАВА V. РЕИНЖЕНЕРИНГ НА ИНТЕРПРЕТАТОРА НА ИНТЕРВАЛНА ТЕМПОРАЛНА ЛОГИКА TEMPURA

За да можем да използваме възможностите на ITL и нейния интерпретатор Tempura, трябва да намерим подход за неговата интеграция в архитектурата на ВОП, като от една страна запазваме хомогенността на пространството, а от друга – съхраняваме пълната функционалност на интерпретатора. За целта са разгледани и

анализирани три основни подхода за трансформация на интерпретатора и неговата интеграция във ВОП [72]:

- Разработване на подходяща „обвивка" на съществуващия интерпретатор, съдържаща изпълними входно-изходни Java класове;
- Реализиране на изцяло нов проект, използващ теоретичния модел на ITL и програмиран на езика за програмиране Java;
- Реинженеринг на оригиналния интерпретатор, програмиран на езика за програмиране C, при съхраняване на пълната функционалност на доказано работещия и широко използван в различни приложения интерпретатор.

Обвиването на съществуващия интерпретатор във входно-изходни Java класове бе сметнат за неподходящ метод, заради очакваната неефективност на run-time системата. Създаването на изцяло нов ITL интерпретатор на езика Java би отнел много време и ресурси.

В дисертацията е избран третия подход, т.е. прилагане на реинженеринг върху C версията на интерпретатора и пренаписването му на езика за програмиране Java като най-подходящ. Нашето решение може да бъде обосновано със следните причини:

- Всички софтуерни модули на ВОП са реализирани на езика за програмиране Java;
- Също така една Java версия би дала възможност за по-лесно създаване на агентно-ориентирана версия на интерпретатора, която да бъде лесно вградена в мулти-агентната архитектура на ВОП;
- От гледна точка на развитието на самия интерпретатор, версия, реализирана на езика за програмиране Java, може да се използва в бъдещи проекти, използващи съвременни обектно и агентно-ориентирани архитектури;
- При този подход изцяло се използва проверената и доказалата своята ефективност в много приложения функционалност на Tempura. Оригиналната версия е използвана в реални системи и разполага с много тестови примери, които са добра основа за първоначални тестове на новата версия ***jTempura***.

5.1 Реализация на реинженеринга на Tempura

Реинженерингът на интерпретатора Tempura бе извършен чрез итеративен подход, по подобие на стандартните модели, като бяха реализирани две ясно различими итерации. Всяка итерация се състои от три основни фази:

Проучващата фаза се състои от серия експерименти с ревърс-инжинерингови техники, приложени върху части от програмния код на интерпретатора [71,37,14].

Като резултат от разработващата фаза получихме аналогична на оригиналната C-Tempura версия, написана на програмния език Java. Тестовете в третата фаза на процеса бяха използвани за проверка коректността на Java-версията на интерпретатора и запазване функционалността на оригиналния код.

При реинженеринговия процес искахме да запазим оригиналния синтаксис и поведение на интерпретатора, което бе проверено в тестващата фаза. Тестовете се проведоха чрез изпълняване на скриптове от оригиналната C-Tempura. Паралелното изпълнение на тестови примери и сравняването на резултатите ни даде възможност да потвърдим коректността на новата версия и съответствието и като формат на входно-изходните данни с оригиналната Tempura [73].

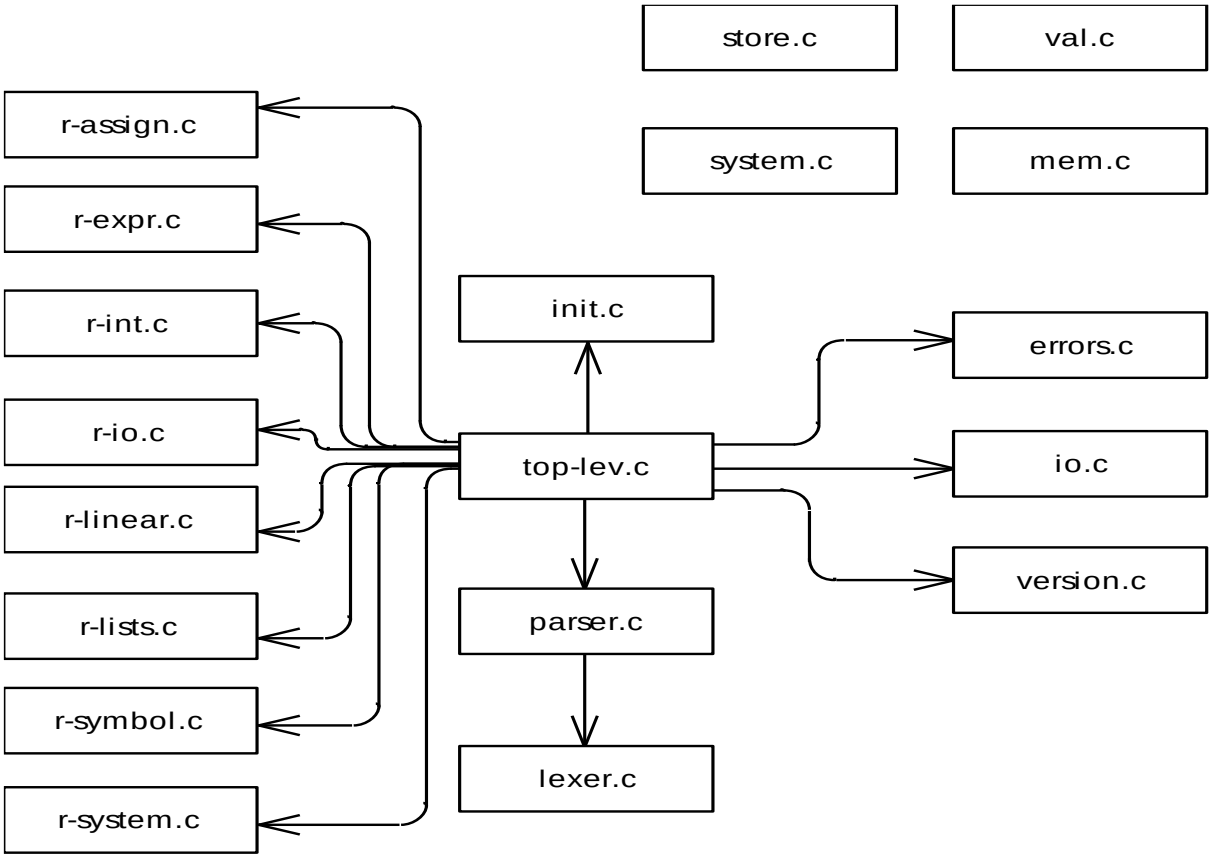
5.1.1 Архитектура на оригиналния Tempura

Програмният код на интерпретатора (C-код) се състои от деветнадесет файла първичен код и пет библиотечни (header) файла.

Отчитайки теоретичния алгоритъм за интерпретиране на ITL формули и твърдения, от резултатите на извършения анализ на програмния код могат да се направят следните изводи:

- Основният програмен цикъл на интерпретатора е реализиран във файл top-level.c;
- Базовите модели и правила за редуциране на темпорални формули и твърдения, според типа на използваните оператори, са реализирани в осем „r" файла;
- Парсерът, модулът за лексикален анализ и входно/изходни операции са включени в три отделна файла;
- Формулите, които ще бъдат интерпретирани, се приемат като текстови низ от входно/изходните процедури.

Принципната структура на първичния код на оригиналния интерпретатор и компонентите му, включени в отделните итерации е представена на схемата на Фигура 11.



Фигура 11 Принципна схема на оригиналния първичен код на Tempura

5.1.2 Реинженерингови итерации

Първата итерация включва ядрото на интерпретатора - Фигура 11. Целта на *проучването* в първата итерация на реинженеринговия процес е, използвайки знанията

за теоретичните основи и концепции на ITL, както и анализа на първичния код на основния програмен цикъл, на структурите и на променливите на оригинала, да се подготвят описание и модел на съществуващата архитектура на Tempura. Същевременно бяха проучени възможности и средства за автоматична трансформация на C-код в Java-код.

По време на *разработването* оригиналният C-код беше трансформиран в кореспондиращ Java-код. За успешното реализиране на разработващата фаза и преодолявайки някои базови различия на C и Java бяха създадени специални трансформационни правила:

- Създаване на Java клас за всеки C файл и неговия съответен header файл;
- Създаване на статичен метод в Java класа за всяка съответна функция в C файла със същото име, като конвенциите на Java за именуване на класове беше игнорирана;
- Създаване на статична променлива в Java класа за всяка глобална променлива в header файла;
- Създаване на статичен метод в Java класа за всяка макро-функция в header файла (където беше необходимо);
- Превръщане на всички булеви изрази и конструкции в съответните такива в Java;
- Заменяне на всички указатели към функции с инстанции на специален интерфейс, който извиква съответните Java-методи.

Задачата на *тестването* е да бъдат проведени интегрирани тестове на ядрото на jTempura, използвайки Tempura Standard Test Suite, доставена от разработчиците заедно с оригиналния код на интерпретатора. Тестовите примери бяха разделени на групи с нарастваща сложност и обем. Целта е да се тества поотделно всяка преведена функционалност, за да се откриват по-лесно технически и логически грешки.

Втората итерация стартира след създаването на работещо ядро на jTempura.

5.2 Резултати от реинженеринга на Tempura

С помощта на теоретичните постановки, описани в труда на Бен Московски [39], успяхме да извлечем семантиката на различните типове оператори и структури, поддържани в Temura.

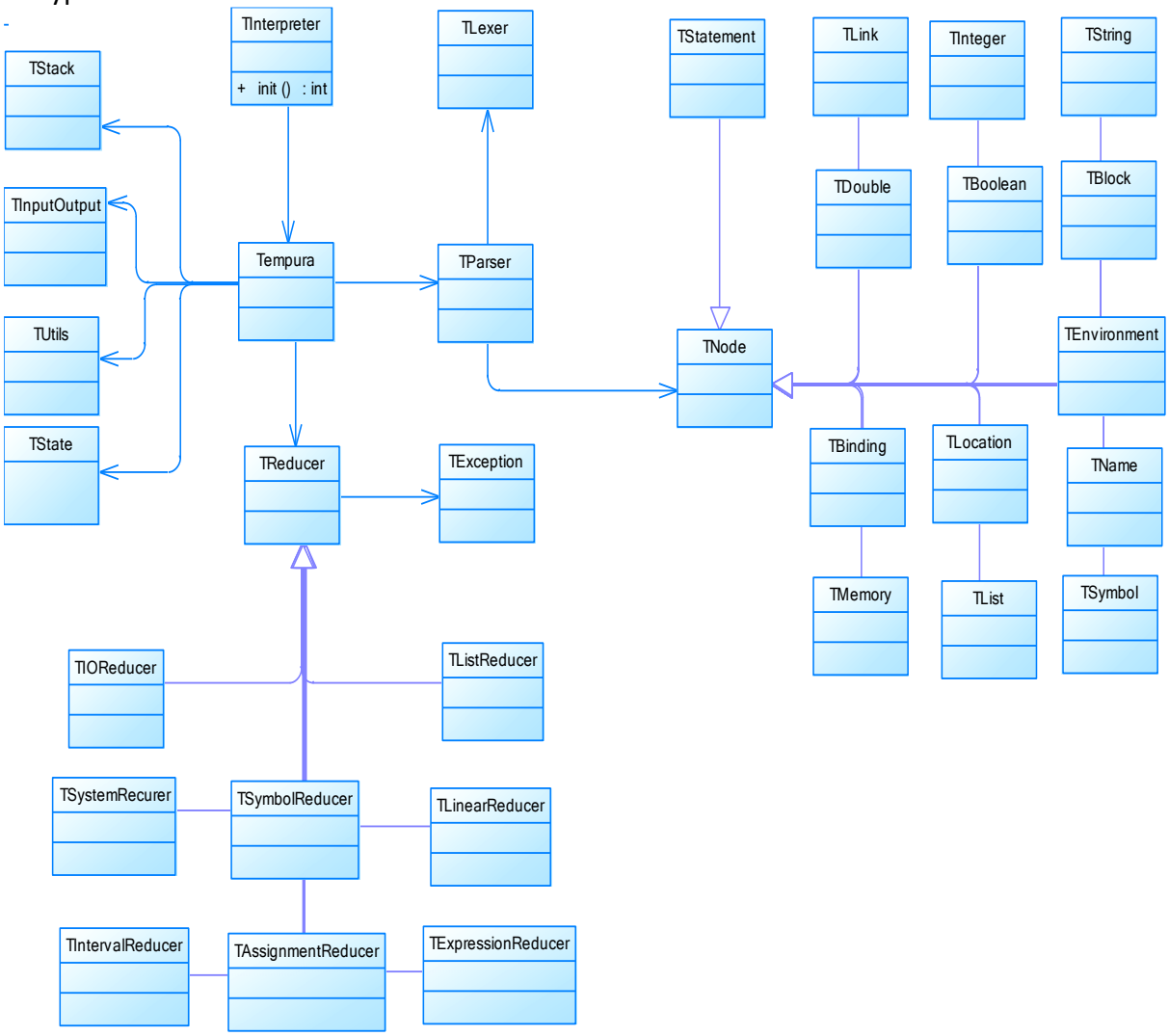
Следвайки плана за фаза „Разработване" на двете итерации, поетапно беше извършен превод на първичния код, като всички типични C конструкции бяха заменени с еквивалентни на Java. Най-често срещаните различия и трудности при превода бяха свързани с подмяната на указателите, използвани от оригиналните разработчици, с конструкции на Java.

След завършването на първоначалния превод започнахме тестове на jTempura. За да сме сигурни в правдивостта на тестовите примери и техните резултати бяха използвани тестове и скриптове, предоставени заедно с програмния код от оригиналните разработчици. Самите тестове се извършват чрез паралелно стартиране на тестови скриптове на двете версии на интерпретатора и сравняване на върнатите резултати.

След успешното приключване на първоначалния набор от тестове (Tempura Standard Test Suite) беше инициирана кореспонденция с поддържащите официалната C-версия на интерпретатора. Целта на кореспонденцията беше да получим по-комплексни тестови случаи (Tempura Extended Test Suite), с които окончателно да се потвърди

коректната работа на jTempura. Преминаването на голям брой допълнителни тестове, съпроводено със съответните корекции в кода на системата, доведе до създаването на стабилен прототип на интерпретатора, който беше официално признат за първа Java версия на ITL интерпретатора Tempura.

Архитектурата на новия интерпретатор jTempura е представена на диаграмата на Фигура 12.



Фигура 12 Архитектура на jTempura

АВТОРСКА СПРАВКА ЗА РЕЗУЛТАТИТЕ В ДИСЕРТАЦИОННИЯ ТРУД

Апробация

Реализацията на SCORM машината е интегрирана в SCORM Player – един от компонентите на образователния портал на DeLC.

Първата версия на портала DeLC е пуснат за реална употреба през 2010 г. Към 2018 г. е използван от 1457 потребители, участващи в 154 групи и притежаващи 176 роли. Потребителите са от 5 факултета на Пловдивския университет и един на Бургаския свободен университет от общо 25 специалности и 46 курса. Към момента на приключване на тази дисертация се разработват 5 лекционни курса в SCORM формат:

- Софтуерни технологии
- Изкуствен интелект
- Програмиране на Java
- Обектно-ориентирано програмиране на Java
- Шаблони за проектиране

Лекционният курс по софтуерни технологии до 2018 г. е бил стартиран от 398 студента, като 69 от тях са го завършили.

Резултати

Приносите на автора, представени в дисертационния труд, могат да се обобщят както следва:

1. Предложен е общ архитектурен модел на среда, поддържаща създаването на електронно съдържание в съответствие със стандарта SCORM 2004. Имплементирана е прототипна версия на развойна среда за електронно съдържание, наречена Selbo 2. Описани са принципите, залегнали при изграждането на средата. Средата осигурява пълни възможности за създаване и редактиране на SCORM съвместимо електронно съдържание. Поддържа се също възможност за персонализация, основаваща се на явно представяне на три основни модела – педагогически модел, модел на знанието (научната област, дисциплина) и потребителски модел. Средата показва определено интелигентно поведение посредством своята проактивност, адаптивност, абстракция над техническата реализация на SCORM. Модулността на Selbo 2 осигурява лесно добавяне на нови компоненти. Потребителският интерфейс е лесен за използване, позволяващ интуитивна работа със средата. Поддържат се също библиотеки с многократно използваеми ресурси.
2. При разработката на средата бе изградена концепцията за интелигентни компоненти. Интелигентният компонент е видоизменен компонент от потребителския интерфейс, към който е прикрепен агент, специално създаден да взаимодейства с компонента. Концепцията предлага модел на слаба свързаност и силна съгласуваност между тях, като агента играе роля на контролер на компонента, а компонента – модел на данните за агента.
3. Предложена е концепция за домейн-зависими онтологии. В тях могат да се съхраняват многократно използваеми ресурси от знания от определен домейн. Онтологиите позволяват формализиране на семантиката свързана с различни обекти и отношенията между тях. Използвайки тези връзки и конкретни насоки за търсене, агент може да състави полезен маршрут от теми, понятия, примери, които автора да използва като скелет за електронния си урок или като допълнение към него. Онтологиите могат да се използват не само като хранилища за конкретни ресурси, но и в системи, каталогизиращи хранилища с вече съществуващи електронни уроци.
4. Реализирана е втора версия на SCORM машина – SCORM 2004 Sequencing and Navigation Engine, удовлетворяваща сертифициращите изисквания на организацията ADL. Машината е интегрирана в образователния портал на DeLC като един от модулите на SCORM Player. Функционалностите на системата изпробвани в тестова среда, в която са планирани, подготвени и изпълнени близо 200 теста от набора SCORM 2004 4th ed. Conformance

TestSuite v1.1.1. Резултатите от тестовете са оценени и са извършени съответни корекции в архитектурата и имплементацията на машината.

5. Разработена е концепция, софтуерна архитектура и прототип на симулационна среда за тестване на InfoStation-базирания и агентно-ориентирания специализиран мидълуер, поддържащ доставката на мобилни услуги в DeLC. Експериментите, извършени с помощта на симулатора позволяват изследване и анализ на поведението на мидълуера на четири нива. На първото – сценарийно ниво – може да се проследи глобалното поведение на мидълуера по време на изпълнението на сценария, определен в експеримента. Фокусът е върху поведението на системните (постоянни) агенти и процеса на генериране на оперативни (временни) агенти, които зависят от промените в средата. Тези промени са описани в описанието на експеримента и са симулирани по време на изпълнението му. На второ – контейнерно ниво - за всеки симулиран компонент (InfoStation Center, InfoStation или мобилно устройство) симулационната среда показва активните контейнери, които са активни в мидълуера. На агентно ниво симулационната среда показва кои агенти живеят в контейнерите на всеки InfoStation. На четвъртото ниво се осигурява възможност за анализ и оценка на локалното поведение на отделните агенти. В резултат на експеримента могат да бъдат анализирани различни характеристики на мидълуера, като напр., проследяване на движението на потребителите и съответната реакция на мидълуера, статистика за генерираните и премахнати оперативни агенти, начало и край на комуникация, пренос на данни между устройства, преглед на състоянието на агентите на всяко хардуерно устройство.
6. Реализиран е реинженеринг на отделни компоненти на формалната среда Tempura за изграждане на нова, напълно обектно-ориентирана Java версия, която може да бъде вградена в DeLC. За целта са разгледани и анализирани следните три основни подхода за трансформация и интеграция на интерпретатора: разработване на подходяща „обвивка" на съществуващия интерпретатор, съдържаща изпълними входно-изходни Java класове; реализиране на изцяло нов проект, използващ теоретичния модел на ITL и програмиран на езика за програмиране Java; реинженеринг на оригиналния интерпретатор, програмиран на езика за програмиране C, при съхраняване на пълната функционалност на доказано работещия и широко използван в различни приложения интерпретатор. В дисертацията е избран третия подход, т.е. прилагане на реинженеринг върху C версията на интерпретатора и пренаписването му на езика за програмиране Java като най-подходящ.

Таблица 2 показва връзката между резултатите, структурата на дисертацията и направените публикации.

Резултат	Тип	Задача	Публикация	Глава
1	Научно-приложен	1	1, 5	1
2	Научно-приложен	1	5	1
3	Научно-приложен	1	1	1
4	Приложен	2	2	2
5	Приложен	3	3	3

6	Приложен	4	4	4

Таблица 2 Граф на дисертационния труд

Перспективи

За бъдещото развитие на получените в дисертацията резултати са предвидени следните насоки:

- Развитие и пълно имплементиране на разработения концептуален модел на средата за подготовка на електронно съдържание. Пълно окомплектоване на средата Selbo 2 с програмни компоненти, предоставящи възможности за работа на различни групи потребители и различни области на знанието (приложни области, дисциплини).
- Планиране, подготовка и провеждане на експерименти с новата версия на SCORM 2004 Engine за нейното стабилизиране. Разработване на предложение и концепция за тясно интегриране на стабилизираната версия на машината с окомплектованата версия на средата Selbo 2.
- Адаптиране на интегрираната среда (Selbo 2 – SCORM 2004 Engine) за различни форми на електронното учене, като например използване в училищна среда, поддържане на учене през целия живот.
- Интегриране на SCORM 2004 Engine в сървърната конфигурация на различни персонални асистенти, подпомагащи различни групи обучаеми.

ПРИЛОЖЕНИЯ

Публикации по дисертационния труд

- D. Mitev, S. Stoyanov, I. Popchev, Selbo 2 – an Environment for Creating Electronic Content in Software Engineering, Cybernetics and Information Technologies (CIT), Vol.9, No 3., ISSN: 1311-9702, Bulgarian Academy of Sciences, 2009, pp. 96-105

- С. Стоянов, И. Попчев, Е. Дойчев, Д. Митев, Г. Чолаков, Архитектура на образователния портал DeLC, 4-та Национална конференция „Образованието в информационното общество", 26-27 май, 2011, Пловдив, 129-138, ISSN 1314-0752

- I. Minov, D. Mitev, Agent-Oriented Middleware Supporting Delivery of Mobile eLearning Services, REMIA 2010: Anniversary International Conference, 10-12 December, 2010, pp. 279-318, ISBN 978-954-423-648-9

- V. Valkanov, D. Mitev, Tempura Reengineering, REMIA 2010: Anniversary International Conference, 10-12 December, 2010, pp. 311-318, ISBN 978-954-423-648-9

- Д. Митев, О. Рахнева, И. Минов, Интелигентни Компоненти в Развойната Среда за Електронно Обучение Selbo 2, Науката, Образованието и Времето Като Грижа, 30 ноември – 1 декември, 2007, Смолян, 82-87, ISBN 978-954-8767-24-8

Участие в проекти

„Software Engineering Education and Reverse Engineering". Проектът е реализиран по линия на Пакта за стабилност в Югоизточна Европа и подкрепен от Германската служба за академичен обмен – DAAD, 2006 – 2016 [15].

БИБЛИОГРАФИЯ

[1] A. Cau, B. Moszkowski. Interval Temporal Logic. <http://www.antonio-cau.co.uk/ITL/>

[2] ADL. SCORM 2004 4th Editon Sequencing and Navigation. <http://www.adlnet.gov/research/scorm/scorm-2004-4th-edition/>

[3] ADL. SCORM 2004 Specification. <http://www.adlnet.gov/scorm/scorm-2004-4th/>, последно посещение 14.07.2015

[4] ADLnet,. (2014, March) ADL. <www.adlnet.gov/>

[5] Amor M., Fernández L.F., Mandow L., Troya J.M., "Building Software Agents from Software Components" in *Current Topics in Artificial Intelligence*.: Springer, Berlin, Heidelberg, 2004, ISBN: 978-3-540-22218-7.

[6] Antoniou, G. и F. van Harmelen, *Semantic Web Primer, second edition*. Cambridge: MIT Press, 2008.

[7] Bellamy R., Andrist S., Bickmore T., Churchill E., Erickson T., "Human-Agent Collaboration: Can an Agent be a Partner?" в *2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* , Denver, Colorado, 2017 , pp. 1289-1294.

[8] Bellifemine, F. L., G. Caire и D. Greenwood, *Developing Multi-Agent Systems with JADE*.: Wiley, 2007.

[9] Berners Lee, T. , J. Handler и O. Lassila, "The Semantic Web", *Scientific American*, vol. 284, pp. 34-43, May 2001.

[10] Berners-Lee, T. , "What the semantic web can represent", W3 org., Scientific report 2000.

[11] Briot, J.P. , Meurisse, T. , Peschanski, F., "Architectural Design of Component-based Agents: A Behavior-based Approach" in *Programming Multi-Agent Systems - ProMAS 2006*., 2007, vol. 4411, pp. 73-92, ISBN 978-3-540-71955-7.

[12] Burstall, R. , "Program proving as hand simulation with a little induction" в *Proceedings of IFIP Congress 74*, Valbonne,France, 1974, pp. 308-312.

[13] Card S., Newell A., Moran T., *The Psychology of Human-Computer Interaction*. Hillsdale: L. Erlbaum Associates Inc, 1983.

[14] Casais, E. , "Automatic reorganization of object-oriented hierarchies: a case study", *Object Oriented Systems*, vol. vol. 1, pp. 95-115, 1994.

[15] DAAD. (2006-2016) Software Engineering: Computer Science Education and Research Cooperation. <https://www2.informatik.hu-berlin.de/swt/intkoop/daad/>

[16] de Jong, Steven , "Fairness in multi-agent systems" в *AAMAS `08 Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: doctoral mentoring program*, 2008, pp. 1742-1743.

[17] Eclipse Foundation. (2010) Eclipse Equinox. <https://projects.eclipse.org/projects/eclipse.equinox>

[18] Eclipse Foundation. (2010) Eclipse Platform. <https://www.eclipse.org/>

[19] Eclipse Foundation. (2010) SWT: The Standard Widget Toolkit. <https://www.eclipse.org/swt/>

[20] Ganchev I., M. O'Droma, F. McDonnell, "Component-Based Platform for a Virtual University Information System" в *6th WSEAS Multiconference CSCC*, Crete, Greece, ISBN 960-8052-63-7 , 2002, pp. 7571-7576.

[21] Ganchev I., M. O'Droma, F. McDonnell, "Component-Based Platform for a Virtual University Information System", *Recent Advances in Computers, Computing and Communications*, pp. 185-190, 2002, ISBN 960-8052-62-9.

[22] Ganchev I., S. Stojanov, M. O'Droma, I. Popchev, "Enhancement of DeLC for the Provision of Intelligent Mobile Services" в *2nd International IEEE Conference on Intelligent Systems (IS'2004)*, vol. 1, Varna, Bulgaria, ISBN 0-7803-8278-1 , 2004, pp. 359-364.

[23] Ganchev, I. , S. Stoyanov, V. Valkanova и M. O'Droma, "Service-oriented and Agentbased Architecture Supporting Adaptable, Scenario-based and Context-aware Provision of Mobile e-Learning Services", *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 3, pp. 771-779, 2011, ISSN 2150-7988.

[24] Gilbert, D. , *IBM Intelligent Agent Strategy*.: White Paper, 1995.

[25] Griss, M.L., Pour G., "Accelerating development with agent components", *Computer*, vol. 34, no. 5, pp. 37 - 43, May 2001, ISSN: 0018-9162.

[26] Guessoum, Z. , "Adaptive agents and multiagent systems", *Distributed Systems Online, IEEE* , vol. vol 5, no. 7, 2004.

[27] Hayes-Roth, и Barbara, "An Architecture for Adaptive Intelligent Systems", Department of Computer Science, Stanford University, Scientific report STAN-CS-TR-93-1496, 1993.

[28] Hoare, C. A.R., "An axiomatic basis of computer programming.", *Communications of the ACM* , pp. 576-580, Octomber 1969.

[29] Hoare, C. A.R., *Communicating sequential processes*. London: Printise Hall International, 1985.

[30] Hoare, C. A.R., *Towards a theory of parallel programming in C*, C.A.R. Hoare and R.H. Perrot, Ed. London: Academin Press London , 1972.

[31] IMS Global. Learning Resource Meta-data Specification. <https://www.imsglobal.org/metadata/index.html>

[58] J., Nielsen. (2010) Website Response Times. <https://www.nngroup.com/articles/website-response-times/>

[32] Kaminka G., Tambe M., "A Synergy of Agent Components: Social Comparison for Failure Detection" в *Second international conference on Autonomous agents AGENTS '98*, Minneapolis, Minnesota, ISBN:0-89791-983-1 , 1998, pp. 459-460.

[33] LSAL, Carnegie Mellon University. (2010) Best Practices Guide for Content Developers. <www.lsal.cmu.edu/lsal/expertise/projects/developersguide>

[34] Manna, Z. и A. Pnueli, "Verification of concurent programs: Temporal framework", *The Correctnes Problem in Computer Science*, pp. 215-273, 1981.

[35] McDonnell F., I. Ganchev, M. O'Droma, "Evolving the Virtual University" в *4th Annual Irish Educational Technology Users' Conference EdTech2003*, Waterford, Ireland, 2003.

[36] Mitrovich, Dejan , Mirjana Ivanovic, Zoran Budimac и Milan Vidakovic, "An overview of agent mobility in heterogeneous environments" в *Proceedings of the Workshop on Applications of Software Agents*, 2011, pp. 52-58.

[37] Moore, , "Automatic inheritance hierarchy restructuring and method refactoring" в *in Proc. Int'l Conf. OOPSLA, ACM SIGPLAN Notices*, 1996, pp. 235-250.

[38] Moszkowski, B. , *A temporal analysis of some concurrent systems*. Berlin: Springer Verlag, Appear in Analysis of Concurrent Systems, in the series Lecture Notes in Computer Science.

[39] Moszkowski, B. , *Executing Temporal Logic Programs*. Cambridge, England: De Montford University, 1985.

[40] Moszkowski, B. , *Ph.D. Thesis: Reasoning about Digital Circuits*.: Department of Compuer Science, Stanford University, 1983, Avaible as technical report STAN-CS-83-970.

[41] Moszkowski, B. и Z. Manna, "Reasoning in interval temporal logic" в *Proceedings of the ACM/NSF/ONR Workshop on Logic of Programs*, 1984, pp. 371-383.

[42] Mueller, J. , *The Design of Intelligent Agents: A Layered Approach (Lecture Notes in Artificial Intelligens)*.: Springer, 1996.

[43] Nwana, H. S., L. Lee и N. R. Jennings, "Co-ordination in software agent systems", *BT Technology Journal*, vol. vol 14, no. 4, pp. 79-88, 1996.

[44] O'Droma, M., I. Ganchev, S. Stojanov, I. Popchev, "On Virtual Universities, Distributed eLearning Centers and Standardised Reusable eContent", *Contemporary Issues in Higher Education*, 2005.

[45] OSGi Alliance. (2010) OSGi Architecture. <https://www.osgi.org/developer/architecture/>

[46] OWL Working Group. Web Ontology Language. <https://www.w3.org/OWL/>

[47] "Proceedings of the International Conference, „FROM DELC TO VELSPACE“, Dedicated to the 10th Anniversary of the Start of the Project Distributed eLearning Center (DeLC)" , Plovdiv, 2014. <http://fmi-plovdiv.org/index.jsp?id=2054&ln=1>

[48] R. Hale, A. Cau, B. Moszkowski. Tempura. <http://www.antonio-cau.co.uk/ITL/itlhomepagese6.html#x7-70006.1>

[49] Rachneva, O. , A. Rachnev, N. Pavlov и ', "Functional Workflow and Electronic Services In a Distributed Electronic Testing Cluster – DeTC" в *Proceedings 2nd International Workshop on eServices and eLearning*, Magdeburd, 2004, pp. 147-157.

[50] Rachneva, O. , A. Rachnev, N. Pavlov и N. Valchanov, "Authoring and Automatic Generation of Circuitries and Drafts in Distributed e-Testing Cluster (DeTC)" в *ELECTRONICS'05*, vol. 2, Sozopol, 21-23, September, 2005, pp. 133-138.

[51] RDF Working Group. Resource Description Framework. <https://www.w3.org/RDF/>

[52] Russel, , J. Stuart и Peter Norvig, *Artificial Intelligence: A Modern Approach*.: Prentice Hall, 2009.

[53] Rustici Software. SCORM Cloud. <https://scorm.com/scorm-solved/scorm-cloud-features/>

[54] Rustici Software. SCORM Sequencing and Navigation. <https://scorm.com/scorm-explained/technical-scorm/sequencing/>

[55] S. Stoyanov, I. Ganchev, I. Popchev, M. O'Droma, R. Venkov, "DeLC -Distributed eLearning Center" в *1st Balkan Conference in Informatics*, Thessaloniki, Greece, ISBN: 960-287-045-1 , 2003, pp. 327-336.

[56] S. Stoyanov, I. Ganchev, I. Popchev, I. Dimitrov, "Request Globalization in an InfoStation Network", *Compt. Rend. Bulg. Acad. Sci.*, vol. 63, no. 6, pp. 901-908, 2010.

[57] S. Stoyanov, I. Ganchev, M. O'Droma, H. Zedan, D. Meere, V. Valkanova, "Semantic Multi-Agent mLearning System" in *Semantic Agent Systems: Foundations and*

Applications, Book Series: Studies in Computational Intelligence vol. 344, M. T. Kone, M. A. Orgun A. Elci, Ed.: Springer Verlag, 2011, ISBN: 978-3-642-18307-2.

[58] Stanford University. (2013, April) Protege. <http://protege.stanford.edu>

[59] Stojanov, S., M. O'Droma, I. Ganchev, "A model for integration of electronic services into a distributed eLearning center" в *14th EAEEIE International Conference*, Gdansk, Poland, ISBN 83-918622-0-8 , 2003.

[60] Stojanov S., I. Ganchev, M. O'Droma, "An Adaptation of the DeLC Architecture to the SCORM Standard" в *4th International Conference on Informatics and Information Technology (CIIT2003)*, Bitola, Macedonia, 2003, pp. 11-14.

[61] Stoyanov, S. , I. Ganchev, I.Popchev, M.O'Droma и V. Valkanova, "Agent-Oriented Middleware for InfoStation-based mLearning Intelligent Systems" в *5th IEEE International Conference on Intelligent Systems IS'10*, London, IEEE Catalog Number: CFP10802-CDR, ISBN:978-1-4244-5164-7, Library of Congress:2009934065 , 2010, pp. 91-95.

[62] Stoyanov, S. , *Context-Aware and Adaptable eLearning Systems - PhD Thesis*. Leicester, UK: De Monfort University, 2012.

[63] Stoyanov, S., I. Ganchev, D. Mitev, V. Valkanov, M. O'Droma, "Service-oriented and Agent-based Architecture Supporting Adaptable, Scenario-based and Context-aware Provision of Mobile e-Learning Services", *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 3, pp. 771-779, 2011, ISSN 2150-7988.

[64] Stoyanov, S. , G. Cholakov, V. Valkanova и M. Sandalski, "Personalized, Reactive and Proactive Providing of e-Learning Services" в *EdiLin Conference, AACE E-Learn 2011 - World Conferenc on e-Learning, Government, Healthcare & Higher Education*, Honolulu, Hawaii, 2011, pp. 2527-2534.

[65] Stoyanov, S. , I. Ganchev, I. Popchev, M. O'Droma и V. Sgurev, "An Approach for the Development of a Context-Aware and Adaptive eLearning Middleware" в *Intelligent Systems: From Theory to Practice*, vol. SCI 299, Berlin, 2010, pp. 519-535.

[66] Stoyanov, S. , I. Popchev, I. Ganchev и M. O'Droma, "From CBT to e-Learning", *Information Technologies and Control*, vol. 4, pp. 2-10, 2005, ISSN 1312-2622.

[67] Stoyanov, S. , V. Valkanova, G. Cholakov и M. Sandalski, "Education Portal for Reactive and Proactive Service Provision" в *COGNITIVE 2011: Third International Conference on Advanced Cognitive Technologies and Applications*, Rome, ISBN: 978-1-61208-155-7 , 25-30 Sept 2011, pp. 99-103.

[68] Stoyanov, S. , V. Valkanova, I. Popchev и I. Minov, "A Scenario-Based Approach to Creating a Virtual Environment for Secondary School Instruction", *Cybernetics and Information Technologies - CIT*, vol. 8, no. 3, pp. 86-96, 2008.

[69] Stoyanov, S. и др., "Virtual Education Space", *Applied Science Journal*, vol. 1, 2014.

[70] TILAB. (2013, June) JADE. <http://jade.tilab.com>

[71] Tokuda, L. и D. S. Batory, "Evolving object-oriented designs with refactoring", *Automated Software Engineering*, vol. vol. 8, no. 1, pp. 89-120, 2001.

[72] Valkanov, V. , "Темпура реинжинеринг", ИИТ-БАН, София, Вътрешен доклад Изграждане на висококвалифицирани млади изследователи по съвременни информационни технологии за оптимизация, разпознаване на образи и подпомагане вземането на решения" - BG051PO001-3.3.04/40, 16 ФЕВРУАРИ, 2010.

[73] Valkanov, V. и D. Mitev, "Tempura Reengineering" в *REMIA 2010: Anniversary International COnference*, Plovdiv, 2010, pp. 311-320.

[74] Valkanov, V. и др., "AjTempura-first Software Prototype of C3A model" в *7th International Conference Intelligent Systems (IS'14)*, vol. 1 Mathematical Foundations, Theory, Analyses, Warsaw, Poland, 24-28 sept., 2014, pp. 427-435.

[75] Wooldridge, M. и N. R. Jennings, *Intelligent agents: theory and practice*.: The Knowledge Engineering Review, 1995.

[76] World Wide Web Consortium. Web Services. <http://www.w3.org/2002/ws>

[77] Zedan, H. , A. Cau, K. Buss и S. Westendorf, *Mapping Human Creativity*. Leicester, UK: De Montford University, 2008.

[78] Вълканов, В. , *Контекстно-ориентирано управление на електронни услуги*. София, България: Академично издателство "Проф. Марин Дринов", 2013, Дисертация, ISBN 978-954-322-701-3.

[79] Вълканова, В. , "Виртуално образователнно пространство за средното училище" в *From DeLC to VelSpace*, Пловдив, България, 26-28 март, 2014, р. (приета за печат).

[80] Вълканова, В. , С. Стоянов, Х. Зедан и И. Попчев, "Модел за изследване креативното мислене и действие на учениците" в *39та конференция на СМБ*, Албена, България, ISSN: 1313-3330 , 2010, pp. 274-280.

[81] Глушкова, Т. , *Адаптивен модел за електронно обучение в средните училища*. Пловдив: Университетско издателство "Паисий Хилендарски" Пловдив, 2011, Дисертация.

[82] Дойчев, Е. , *Среда за електронни образователни услуги*. Пловдив, България: Университетско издателство "Паисий Хилендарски" Пловдив, 2013, Дисертация.

[83] (2010 , юли) Практическо ръководство: Създаване и управление на SCORM учебно съдържание. <http://tu-sofia/bul/EO/EO.htm>

[84] Рахнева, О. , *Разпределен клъстер за електронно тестване*. Пловдив, 2006, Дисертация.

[85] Стоянова-Дойчева, А. , *Дефиниране на процес и средства за рефакторинг в обучението по софтуерни технологии*. Пловдив, България: Университетско издателство "Паисий Хилендарски" Пловдив, 2011, Дисертация.

[86] Чолаков, Г. , *Хибридна архитектура за изграждане на Разпределен център за електронно обучение (DeLC)*. Пловдив, България: Университетско издателство "Паисий Хилендарски" Пловдив, 2013, Дисертация.