SPICIALIZED SCIENTIFIC COMMITTEE ON COMPUTER SCIENCE AND MATHEMATICAL MODELING AT HIGHER ATTESTATION COMMISSION

SOFTWARE SYSTEM FOR COMPUTER SIMULATION OF METAL VAPOR LASERS

Anna Atanasova Malinova

Author's summary of a dissertation
for the acquisition of the educational and scientific degree "PhD"

Scientific specialty: 01.01.12 Computer Science

Supervisor: Associate Prof. PhD Snezhana Gocheva-Ilieva

Sofia, 2009

СПЕЦИАЛИЗИРАН НАУЧЕН СЪВЕТ ПО ИНФОРМАТИКА И МАТЕМАТИЧЕСКО МОДЕЛИРАНЕ ПРИ ВАК

Анна Атанасова Малинова

СОФТУЕРНА СИСТЕМА ЗА КОМПЮТЪРНА СИМУЛАЦИЯ НА ЛАЗЕРИ С МЕТАЛНИ ПАРИ

АВТОРЕФЕРАТ

на дисертация
за присъждане на образователна и научна степен "доктор"
по научна специалност: 01.01.12 Информатика

Научен ръководител: доц. д-р Снежана Гочева-Илиева

Рецензенти: .....
.....

София, 2009 г.

Дисертационният труд е обсъден и насочен за защита на разширено заседание на катедра “Компютърни технологии” при Факултета по математика и информатика на ПУ “Паисий Хилендарски”.

Защитата на дисертацията ще се състои на ..... г. от ... часа в мултимедийната зала на Института по математика и информатика на БАН, ул. “Акад. Георги Бончев”, бл. 8, гр. София, на открито заседание на СНС по информатика и математическо моделиране при ВАК.

Материалите по защитата са на разположение на интересуващите се в библиотеката на Института по математика и информатика на БАН, ул. “Акад. Георги Бончев”, бл. 8, гр. София.

Дисертационният труд съдържа 139 страници в основната си част. Използваната литература включва 144 източника, от които 128 на английски език, 13 на български и 3 на руски език.

Списъкът на авторските публикации се състои от 7 заглавия.

Автор: Анна Атанасова Малинова

Заглавие: Софтуерна система за компютърна симулация на лазери с метални пари

Тираж: .... бр.

Обща характеристика на дисертационния труд

Актуалност на проблема

Интензивното развитие на компютърната техника, информатиката и информационните технологии дадоха нов смисъл на понятията научен експеримент и съвременна физична лаборатория. Важно направление във физичните изследвания е решаването на задачите на физиката на плазмата, която е в основата на изучаването на процесите на широк кръг технически устройства и в частност на лазерите с метални пари.

През последните години към газовите лазери и лазерите с метални пари има засилен интерес, поради многобройните им приложения в нанотехнологиите, плазмените технологии, модерната лазерна медицина, при изследване на екологични замърсявания, изследвания на атмосферата и морското дъно, и др.

В същото време тяхното проектиране среща известни трудности. Измерването на характеристиките на лазерното излъчване по класическите методи е силно затруднено поради смущенията от високочестотното поле, а новите фотометрични методи дават незадоволително ниска точност.

Във връзка с това, необходимостта от разработка и използване на специализиран софтуер за високочестотни и импулсно-периодични лазери с метални пари се обуславя от:

- значително по-ниската цена на компютърната симулация, вместо скъпоструващите и продължителни експериментални изследвания;
- непосредствената възможност за изучаване на физическите процеси и прогнозиране на поведението на лазерната среда;
- лесното въвеждане и промяна на големия брой параметри (над 50), влияещи на процесите;
- визуализация на лазерните характеристики, таблично и графично представяне на основните резултати от моделирането;
- цялостно скъсяване на времето за проектиране.

Друга необходимост от разработка на настоящия дисертационен труд произтича от възможността за практическата приложимост на резултатите. След определен спад в периода от 1989 до 2000 год., в момента в България се възобновява производството на лазери и лазерни устройства.

В резултат на засиления интерес към симулацията на процесите в лазерите с метални пари са получени нови резултати за подобряване на лазерните характеристики, в това число са разработени нови математически и статистически модели, както и съответните алгоритми за числени пресмятания [5-18]. По-голямата част от тези кодове решават отделни конкретни задачи и станаха основа за търсене на решения за интегриране и повторно използване на наследени физични приложения, които са многократно тествани и оптимизирани.

Настоящата дисертация е в областта на приложението на информационните технологии за създаване на софтуер за числени симулации и по-точно на лазери с метални пари. Работата е свързана с извършване на реинженеринг чрез обвиване на наследен физичен софтуер и изграждане на архитектура, ориентирана към използването на услуги, даваща възможност чрез интегриране на модули, разработени от различни научни екипи, да се постигне автоматизация на процеса на проектиране, изчисляване и оценка на основните характеристики на лазерната среда, изследване и контрол на лазерната генерация на базата на известни математически, статистически и оптимизационни модели.

Цел и задачи на дисертационния труд

Основната цел на дисертационния труд е да се разработи архитектура на софтуерна система за симулация на лазери с метални пари, позволяваща интегрирането и повторното използване на наследени физични приложения и да се реализира прототип на базата на съвременни информационни технологии.

Основните задачи са:

Задача 1. Да се предложи и реализира процедура за интегриране на съществуващ софтуер в областта с цел повторно използване на наследени сложни математически и физични кодове, които са многократно тествани и оптимизирани в продължение на последните две десетилетия.

Подзадачи:

1.1 Да се анализират наличните технологии и стандарти, приложими в областта.

1.2 Да се предложат процедури за интегриране на наследени модули, съобразно характеристиките на наличния наследен софтуер за симулации в областта.

1.3 Да се предложи архитектура на обвиващия код от гледна точка на обектно-ориентираното проектиране и прилагане на образци за проектиране.

1.4 На базата на предложените процедури и дизайн да се създадат обвивки на наследени модули.

Задача 2. Да се предложи и реализира архитектура, позволяваща съвместна работа на наследени (чужди и собствени) и нови модули в разработваната софтуерна система, и да се създадат прототипи на инструментални средства, обслужващи различните етапи на реалната симулация.

Подзадачи:

2.1 Да се направи анализ на избраната архитектура, като се обоснове нейната приложимост за целта на настоящата дисертация;

2.2 Да се направи анализ и избор на съвременни обектно-ориентирани технологии и рамки за разработка на приложения в съответствие с избраната архитектура;

2.3 Да се разработи удобен потребителски интерфейс и съвременни инструменти за въвеждане на данните със и без допълнителна обработка;

2.4 Да се реализира процедура за основната моделна задача за компютърна симулация на потенциала и интензитета на електричното поле, които по-нататък са основа за пресмятане на останалите параметри на газовия разряд в лазерното устройство;

2.5 Да се разработи удобна интерактивна среда за представяне и контрол на резултатите - в табличен и графичен формат.

Структура и обем на дисертационния труд

Дисертацията е организирана в увод, три глави, заключение, списък на използваната литература и списък на авторските публикации по темата.

Обемът на основната част е 139 страници, а на използваната литература – 10. Използваната литература включва 144 заглавия - 128 заглавия на английски език, 13 на български и 3 на руски език, като част от тях са адреси на уеб сайтове. Списъкът на публикациите на дисертанта по темата се състои от 7 заглавия (5 публикувани и 2 приети за печат).

В Първа глава е направен обзор на съществуващия софтуер за симулация на плазма и газов разряд и в частност на софтуера по лазери и лазерни системи. На базата на извършения преглед са направени изводи за основни характеристики на използвания софтуер. В резултат на това са дефинирани целта на настоящата дисертация и задачите, които трябва да се реализират за нейното постигане.

Във Втора глава е описан процесът на обвиване на наследен физичен код с цел вграждането му в Java-базирана среда за числена симулация на лазери с метални пари. Въз основа на направения в Глава 1 преглед на съществуващия софтуер, са анализирани наличните технологии и стандарти, приложими в областта. В резултат са предложени процедури за интегриране на наследени приложения, написани на C, C++ и FORTRAN. Създаването на обвиващ код включва прилагането на основни принципи на многоезичното

проектиране, използването на методи, при които Java приложението комуникира с наследен код, който е част от същия процес - Java Native Interface, както и на методи, при които се обвива приложение като цяло чрез стартирането му в процес, външен за виртуалната машина. Изграждането на обвиващия код е разгледано от гледна точка на обектно-ориентираното проектиране и прилагането на образци за проектиране, въз основа на което е предложена и реализирана архитектурата на обвиващия код. На базата на предложените процедури и архитектура са създадени обвивки на наследени модули, като са описани:

- обвиване на солвър за пресмятане на разпределението на потенциала и интензитета на електричното поле;
- обвиване на цяло приложение – генератор на мрежи по метода на крайните елементи;
- обвиване на базовата функционалност на PLASIMO – софтуерна система за симулиране на нискотемпературна плазма;

В Трета глава е представено приложението на архитектура, ориентирана към използването на услуги, за изграждане на софтуерна система за симулация на лазери с метални пари. Разгледани са основните аспекти на този архитектурен подход и неговата реализация чрез уеб услуги, като е мотивирано прилагането му за научен софтуер в областта на физиката. Представени са технологията и архитектурата на уеб услугите. По-нататък е разгледан езикът BPEL като едно от най-съвременните средства за реализиране на архитектура, базирана на услуги. Представено е приложението на BPEL за изграждане на научни потоци за симулация. Разисквана е BPEL спецификацията в контекста на изискванията към един научен поток. В края на Глава 3 е описано приложението на представените подходи за реализиране на симулация на лазери с метални пари:

- изграждане на BPEL процес за симулация на разпределението на потенциала и интензитета на електричното поле;

- приложение при софтуерната система за симулиране на нискотемпературна плазма PLASIMO – създаване на прототипа WebPlasimo. Представени са компонентите и функционалността на WebPlasimo, както и създадените за целите на прототипа инструментални средства.

В **Заключение** се посочва къде са представени и публикувани резултатите от работата и се разглеждат перспективи за бъдещо развитие на представената работа.

Кратко съдържание на дисертационния труд

Глава 1. Обзор на съществуващия софтуер по лазери и лазерни системи

В тази глава е направен опит да се представят различните типове лазерен софтуер, който е намерил най-широко разпространение в последното десетилетие. Съществуват десетки частични или пълни софтуерни системи, като броят им непрекъснато се увеличава. Това показва значителен интерес на потребителите от една страна, както и все по-голямо приложение на информационните технологии в тази област от друга.

В обзора са подбрани над 20 софтуерни продукта (лицензиран и свободен софтуер), като са посочени производителите им и е дадена препратка към техните интернет страници. Продуктите се отнасят основно за симулиране на плазма, включително високочестотен газов разряд, което има пряко отношение към симулирането на процесите, протичащи в лазерите с метални пари.

Обзорът е направен в две направления:

- софтуерни пакети за симулация на процесите в различни лазерни системи;
- широкопрофилни софтуерни системи, приложими в предметната област.

Част от изводите след проучването на съществено количество наличен софтуер са:

- По-голямата част от лазерния софтуер е разработван повече от десет години;
- Необходими са мултидисциплинарни екипи от специалисти: основно математици, физици и информатици;

- Широкопрофилните софтуерни системи могат да послужат само за част от процеса на една числова симулация за разлика от специализирания симулационен софтуер за лазери. Тези системи също така са скъпи и изискват продължително време за изучаване;
- Преобладаващите езици за програмиране са C++ и FORTRAN;
- Приложенията са настолни;
- Няма възможност за интеграция с други системи;
- Изолирани екипи, което води до повтарящи се усилия;

В резултат на направения обзор и на аргументите за актуалност и значимост на темата, са дефинирани основната цел на дисертацията и задачите за нейното постигане.

Глава 2. Интегриране на наследен физичен софтуер

2.1 Използвана методика

2.1.1 Мотивация за използване на наследен физичен софтуер

Значително количество ресурси са инвестирани в създаването на високо производителен научен софтуер за числови симулации. Тези приложения често се определят като наследен софтуер, тъй като са разработени с технологии, предшестващи съвременните подходи и най-добри практики. Тъй като при проектирането на тези кодове в общия случай не е предвидена възможност за взаимодействие и повторно използване, то най-често тези пакети с висока научна стойност не могат да бъдат използвани от множество приложения като резултат от разлика в езиците за разработка, на интерфейсите или на стила на програмиране като цяло [1]. Например, голяма част от софтуера, разработен и използван за числови симулации, представлява конзолни приложения, написани на FORTRAN, C, C++, работещи на Unix или Windows машини, т.е. написани са на различни езици и за различни платформи.

От изключителна важност е такива мащабни и сложни софтуерни пакети да могат да бъдат повторно използвани, без да се налага инвестирането големи количества време и труд за техния реинженеринг.

От друга страна, изискванията към съвременните симулации включват множество физични и химични процеси и са на практика изключително сложни мултидисциплинарни симулации. Пример за това е симулацията на процесите, протичащи в газовете в състояние на плазма, което представлява основна част от процеса на симулация на лазери с метални пари.

Екипите за разработка много рядко могат да имат необходимия опит и компетентност във всички научни области, които обуславят една съвременна симулация в областта на физиката на твърдото тяло. Всичко това естествено води до извода, че е невъзможно един разработчик, или дори само една единствена институция или екип, да създадат подобен род научен софтуер в изолация. Следователно изграждането на научен софтуер за симулации от многократно тествани във времето и доказали стойността си компоненти е много по-надеждно и ефективно от опита тези симулатори да се изграждат изцяло отначало.

2.1.2 Реинженеринг чрез обвиване

Реинженерингът на наследени системи е дефиниран в [4] като : „*Проучване и промяна на една софтуерна система за реконструирането ѝ в нова форма*”.

Промяната и проучването на софтуерната система може да се наложи по различни причини като:

- разделяне на монолитни системи на отделни части с цел по-лесната им продажба;
- повишаване на производителността;
- промяна на проекта с цел подобряване на поддръжката, преносимостта и др.;
- преместване към друга платформа;
- използване на нови технологии.

Техниките за реинженеринг са:

- Реструктуриране (restructuring) – трансформация на една форма на представяне в друга в относително едно и също абстрактно ниво, като се запази външното поведение на системата (функционалност и семантика). В този смисъл пример за реструктуриране са подобряването на стила на програмиране, редактиране на документацията, изграждане на функционални структури и др.
- Рефакторинг (refactoring) – дисциплинирана техника за промяна на вътрешната структура на съществуващ обектно-ориентиран код без да се променя неговото външно поведение. В [31] е разгледан процесът на рефакторинг и са представени редица инструменти за неговото автоматизиране.
- Обвиване (wrapping) – техника за създаване на нов интерфейс, изолиращ наследения софтуер от външните системи, обекти или потребители, без да се променя наследения код. По този начин се изгражда един нов слой върху наследения софтуерен компонент или система, като се скриват вътрешния код и логика. При това новият интерфейс включва само тази функционалност от наследения софтуер, която е необходима за външните системи.

Реинженерингът чрез обвиване е единствената техника, не променяща наследения софтуер и отговаряща на мотивацията за интегриране и повторно използване на наследен физичен софтуер, представена в предходния параграф. Създаването на обвивки дава възможност за възползване от съществуващ сложен математически и физичен софтуер и достъп до най-съвременни технологии.

2.1.3 Създаване на Java обвивки на модули с наследен код чрез Java Native Interface

В този параграф се описва създаването на обвиващ Java код за наследени модули чрез Java Native Interface като модел за интегриране, използван от Java платформата.

В цялата Глава втора се използва **терминът „нативен”** за краткост и поради масовото му използване за означаване на специфичен за машината (платформено зависим, native) код. Според спецификацията Java Language Specification [19] един метод може да бъде деклариран като *native*, което означава, че е *„имплементиран като платформено зависим код, написан обикновено на друг език за програмиране като C, C++, FORTARN или асемблер”*. Но спецификацията не дава предписание как да бъде реализиран този платформено зависим код и как да се свърже с Java кода.

2.1.3.1 Java Native Interface – основни характеристики, алтернативи, предимства

Като част от виртуалната Java машина, JNI дава възможност взаимодействието да се осъществява и в двете посоки.

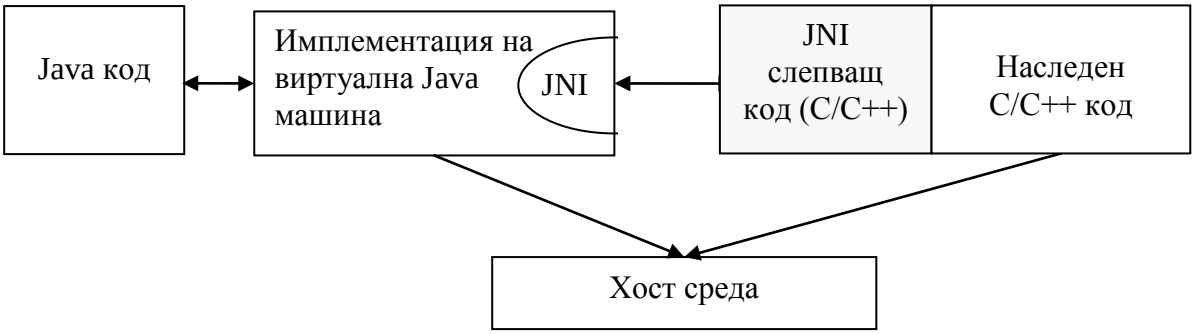
В настоящият параграф са разгледани такива аспекти на създаването на обвивки чрез JNI като:

- Java програма извиква нативни методи;
- нативен метод достъпва членовете на Java клас/обект;
- начинът, по който се извършва съответствието на типовете;
- обработката на изключения;

Разгледани са евентуалните недостатъци от използването на JNI, както и алтернативни подходи, даващи възможност на Java кодът да взаимодейства с код, написан на други езици. Обща характеристика на тези алтернативни решения е, че Java приложението и нативния код съществуват в различни процеси (в някои случаи и на различни машини). В някои случаи, обаче, целта е Java приложението да комуникира с нативен код, който е част от същия процес. Именно в тези случаи използването на JNI интерфейса е необходимият подход. Разгледани са някои примери за такива ситуации.

2.1.3.2 Обвиване на наследен C/C++ код

На Фигура 2.4 е показано схематично обвиването на наследен C/C++ код. Основен момент тук е създаването на слепващия JNI код, който се реализира на C/C++.



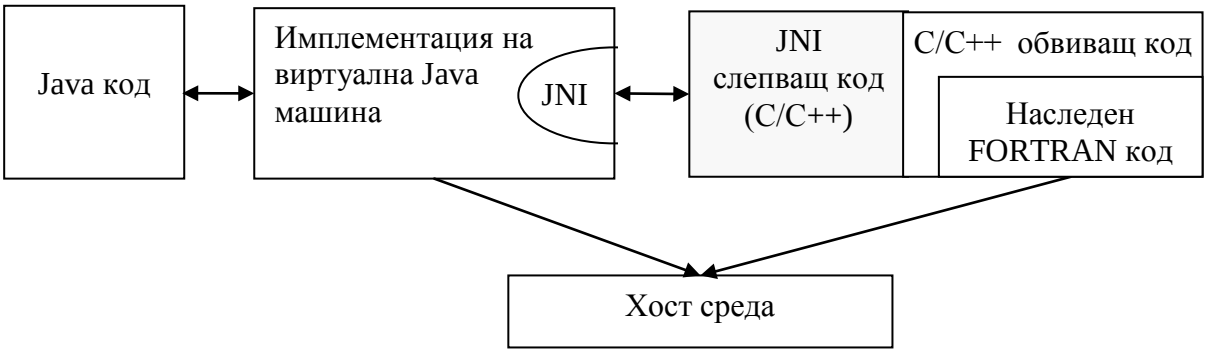
Фигура 2.4 Обвиване на наследен C/C++ код

Процедурата на обвиване на C/C++ код, която е приложена в настоящата дисертация и отпечатана в [23], включва следните основни стъпки:

1) Създаване на Java клас, съдържащ декларации на нативните методи.
2) Генериране на C заглавен файл, който да се включи в междинния C/C++ код, имплементиращ Java методите, декларирани като `native` в т.1).
3) Имплементиране на нативния метод чрез C/C++. Междинният (слепващ) C/C++ код реализира декларираните в създадения заглавен файл функции.
4) Компилиране на нативната имплементация до нативни библиотеки за динамично зареждане (.dll за Windows, или .so за Linux).

2.1.3.3 Обвиване на наследен FORTRAN код

Като допълнение към процедурата за обвиване на C/C++ наследени кодове, описана в предишния параграф, тук се явява съвместното компилиране на C/C++ и FORTRAN кода в динамична библиотека.



Фигура 2.5 Обвиване на наследен FORTRAN код

На Фигура 2.5 е показано схематично обвиването на наследен FORTRAN код. Както се вижда освен създаването на слепващия JNI код, тук се създава и допълнителен C/C++ код, обвиващ наследеното приложение. Това добавя още един слой, в който трябва да се съблюдават принципите на многоезичното програмиране, описани в следващия параграф.

Преходите FORTRAN-C/C++ и обратно не изискват наличието на допълнителен интерфейс, подобен на JNI. Взаимодействието FORTRAN-C/C++ е двупосочно и се реализира директно, поддържа се на ниво компилатори, като има известни разлики при различните компилатори и версии на FORTRAN.

Една C/C++ функция извиква функция или процедура на FORTRAN, така както извиква всяка друга C функция, като се спазват конвенциите на извикване и именуване и параметрите се предават само по адрес, както е описано в следващия параграф.

Наличието на прехода FORTRAN-C/C++ може да окаже влияние върху бързодействието на цялостната система при наличието на фактори, които варират за различните версии на компилаторите и езиците и трябва да се проучват за всеки конкретен случай.

2.1.3.4 Принципи на многоезичното програмиране и JNI

При създаването на Java обвивки на наследени кодове, както по принцип при прилагането на многоезично програмиране при интегриране на C/C++ и FORTRAN код, възникна необходимостта от съблюдаване на следните ключови моменти:

- конвенции на извикване (calling conventions);
- конвенции на именуване (naming conventions);
- изпращане на параметри по стойност или по адрес;
- работа с типовете данни в различните езици.

Разглеждането е направено както за случая на взаимодействие Java-C/C++, така и на FORTRAN-C.

2.1.3.5 Техники за обвиване на нативни функции – „съответствие едно-към едно" и „споделени стъбове". Сравнение

Техниките „съответствие едно-към-едно" (one-to-one mapping) и „споделени стъбове" (shared stubs) са два подхода за създаване на обвиващи Java класове за нативни библиотеки.

Подходът „съответствие едно-към-едно" изисква да се създаде по една стъб функция за всяка нативна функция, която искаме да обвием. На всяка функция, декларирана като `native` в Java класа, съответства една нативна стъб функция, която от своя страна съответства на една функция от наследения код. Тук стъб функцията изпълнява две функции:

- адаптира конвенцията на извикване на нативната функцията към тази, която се очаква от виртуалната Java машина
- извършва преобразуване между Java и нативните типовете данни, така както това е дефинирано в JNI спецификацията (виж параграф 2.1.3.1).

Създаването на една стъб функция за всяка нативна функция, която обвиваме, може да окаже много трудоемко, когато много голям брой нативни функции трябва да бъдат обвити. В този случай може да се приложи техниката „споделен стъб".

Споделеният стъб е нативен метод, който препраща към други нативни функции. Той отговаря за конвертирането на типовете на аргументите в зависимост от това какво изпраща извикващата функция и какво се очаква от нативната функция.

След разглеждане на предимствата и недостатъците на двата метода от гледна точка на баланс между производителност, преместваемост и кратък срок на реализация, в настоящата дисертация е избрана техниката „съответствие едно-към-едно" за изграждане на Java обвивки на наследени физични кодове с цел създаване на среда за симулация на лазери с метални пари.

2.1.3.6 Създаване на Java peer класове за обвиване на нативни структури от данни

В процеса на работа възникна необходимост да се обвиват и нативни структури от данни. Класове, които директно съответстват на нативни структури от данни, се наричат *peer класове.* Обща характеристика на peer класовете е, че съдържат поле, което сочи към нативната структурата от данни.

Основната идея е да се дефинира Java peer клас, който съответства на C++ клас. Всяка инстанция на Java peer класа съответства на инстанция на дадения C++ клас, управлявайки жизнения цикъл на C++ обекта. В този случай е необходимо всяко извикване на Java метод, деклариран като `native`, да може да локализира подходящия C++ обект. На практика това означава да се съхранява указател към C++ обекта в Java обекта. Тъй като езикът Java не поддържа концепцията за указател, то Java peer класът ще съдържа

целочислено поле, сочещо към C++ обекта - това е 32- или 64-битово поле в зависимост от платформата.

Основен аспект при работа с Java peer класовете е **освобождаването на заетата от нативната структура памет.** Peer класовете се дефинират в Java кода и следователно инстанциите им ще бъдат автоматично унищожени от оптимизатора на памет (garbage collector), когато не се използват повече. В същото време е необходимо да се гарантира, че паметта, заета от съответстващите нативни обекти, също ще бъде освободена. В противен случай ще се получат утечки в паметта.

Като евентуално решение е разгледано включването на `finalize` метод в интерфейса на peer класа, като са посочени недостатъците и алтернативите на този подход.

Разисквано е, че утечки в паметта могат да се получат и ако не се извърши освобождаване на заетата от нативните структури памет по всички пътища на изпълнение. Това включва възможните изключения, които могат да възникнат по време на процеса на използване на инстанция на peer клас.

2.1.3.7 Влияние на JNI обвивката върху бързодействието

В изследване на Sun [32], сравняващо бързодействието на един и същ сложен алгоритъм, реализиран чрез Java и чрез C, се показва, че C версията, извиквана от Java през JNI слепващ код, може да е многократно по-бавна от чистата Java реализация. Като основна причина тук се изтъква, че при извикване на JNI код виртуалната Java машина трябва да достъпва фрагменти от код извън нейната нормална оперативна среда. Когато C код прави връщане към Java средата, се извършва подобен преход. Оценка на загубите от тези преходи е дадена в [32], като са сравнени няколко различни техники за копиране на масив от 1000 елемента от примитивен тип. Операцията копиране е реализирана чрез различни нативни и ненативни методи. За по-добро сравнение на различните техники, всяко копиране е извършено 10000 пъти. За сравнение е използвано и извикване на нативна функция, която не прави нищо.

Освен изследването на Sun, подобно разглеждане и изводи за ефективно интегриране на Java и нативен код са направени и в [20]. Изследването е направено за осем различни имплементации на виртуална Java машина както за Linux, така и за Windows. За разлика от изследването на Sun, тук не е правено сравнение с ненативни извиквания. Основните изводи, представени в [20], са:

- Копирането на масиви и символни низове е по-неефективно от директния достъп на JNI до Java масиви и низове;
- За нативни методи с малко количество изчисления загубите от преходите Java-нативен код надхвърля спечелената производителност, постигната чрез реализацията на пресмятанията с машинно зависим код.
- Интензивното извършване на обратни извиквания от нативния код към Java кода трябва да се избягва, тъй като при някои реализации на виртуални машини, то е неефективно.

От разгледаното в настоящия параграф може да се направи заключението, че JNI обвивката би могла да повлияе върху общата производителност на симулацията чрез количеството на преходите Java – нативен код и обратно. При извършения реинженеринг чрез обвиване, показан в параграфи 2.2.2 и 2.2.4, броят на тези преходи е сведен до възможния минимум като преходите са „едрозърнести”, т.е. инициира се наследената функционалност, представляваща основна стъпка на симулацията, свързана с интензивни изчисления, след което се връщат резултатите в Java слоя.

2.1.4 Обвиване на приложение като цяло

Същността на предложения метод се изразява в стартирането на приложението в отделен процес, който е външен за виртуалната Java машина, като аргументи се изпращат през стандартния вход и резултати се получават чрез стандартния изход.

Процесът на обвиване в този случай включва използването основно на два Java класа - `java.lang.Runtime` и `java.lang.Process`.

Като недостатък на метода може да се отбележи факта, че изглежда много лесно осъществим, но на практика е свързан с множество скрити проблеми. Например в JDK Javadoc документацията е специфицирано, че тъй като някои нативни платформи осигуряват буфер с ограничен размер за потоците на стандартния вход и изход, то опит за запис или прочитане наведнъж на входен/изходен поток от процес-дете, може да предизвика блокиране на под-процеса. Това налага буфериране на горните операции при прихващане от родителя на изхода от процес-дете.

Друг недостатък на метода е необходимостта предварително да е известно името и местоположението на извикваното приложение, както и факта, че не всяка шел команда може да бъде подадена като вход на `Runtime.exec` - например невъзможно е по този начин да се ползва пренасочване на входа/изхода и др.

В резултат може да се направи извода, че техниката с използване на метода `Runtime.exec` е с много по-ограничени възможности от тези на метода с прилагане на Java Native Interface.

2.2 Приложение за симулация на лазери с метални пари

2.2.1 Проектиране на обвиващия код

Настоящият параграф представя някои подходи, приложени при проектирането на обвиващия код като се прави отнасяне към базови образци за проектиране. Представени са и създадените UML диаграми. Резултатите са отпечатани в [27].

Разгледани са следните случаи:

- Единствен клас обвива цялото наследено нативно приложение;
- Създаване на библиотека от обвиващи класове;
- Приложение на образците за проектиране Adapter и Proxy (“GoF”);
- Приложение на образеца WrapperFacade за обвиване на необектен код.

2.2.2 Обвиване на солвър за пресмятане на разпределението на потенциала и интензитета на електричното поле

Настоящият параграф описва процеса на изграждане на Java обвивка на солвър, написан на FORTRAN77 под Windows. Разглежданият солвър използваме за пресмятане на разпределението на потенциала и интензитета на електричното поле като част от процеса на симулация на лазери с метални пари. Тук бяха приложени методиката, описана в 2.1.3.3, и принципите на многоезичното програмиране, описани в 2.1.3.4. Резултатите са отпечатани в [23] и частично в [24].

Солвърът е с относително несложен интерфейс на взаимодействие – приема като параметри масиви от входни данни и връща таблица от числа, която за простота на изследването се записва в текстов файл.

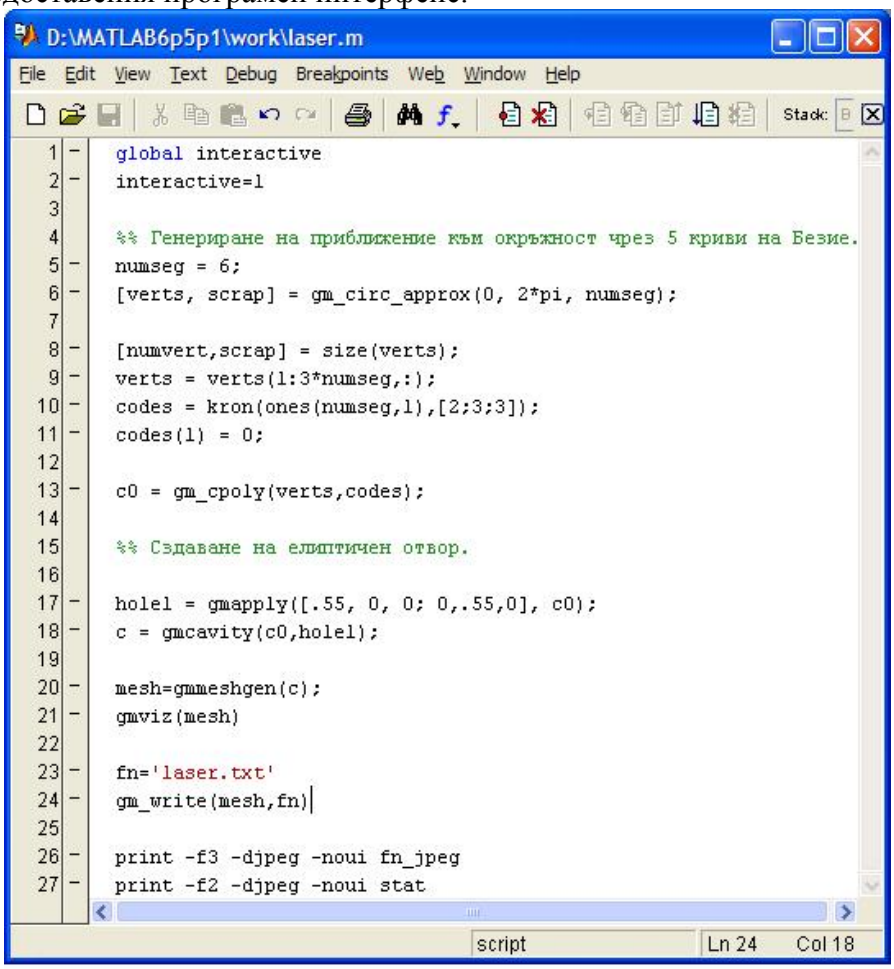
Показана е следната последователност от стъпки: получаване на манипулатор на динамичната библиотека; ако манипулаторът е валиден, се прави опит за получаване на адрес на функция – FORTRAN функцията POISSONSUB; ако полученият адрес на функция е валиден, то се прави извикване на функцията; накрая се извършва освобождаване на библиотеката за динамично свързване.

Крайната цел на извършеното обвиване на солвъра за пресмятане на разпределението на интензитета на електричното поле е полученият обвиващ Java клас да бъде преобразуван в уеб услуга, която да бъде включена в BPEL процес за симулация, както е показано е Глава 3.

2.2.3 Обвиване на цяло приложение – генератор на мрежи по метода на крайните елементи

Тук се описва обвиването на едно цяло приложение чрез стартирането му като външен за виртуалната машина процес по метода, описан в параграф 2.1.4. Резултатите са отпечатани в [22].

Като пример беше избрано приложението QMG 2.0 [30]. Пакетът QMG генерира мрежи по метода на крайните елементи в дву- и триизмерни пространства. Пакетът предоставя програмен интерфейс, състоящ се от множество процедури, даващи възможност да се създават описания на различни геометрии. Част от функциите се написани на Matlab, други на C++ или Tcl/Tk като функциите, написани на C++, могат да се извикват директно от Matlab. За целите на настоящото изследване бяха създадени тестови скриптове, ползващи предоставения програмен интерфейс.



Фигура 2.17 Matlab скрипт за описание на геометрията на лазерното устройство и генериране на мрежа чрез пакета QMG 2.0

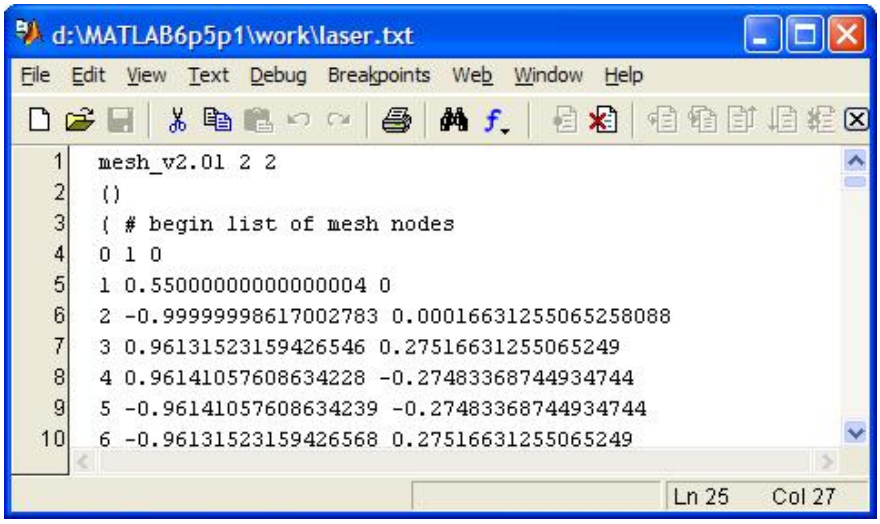
Един от създадените скриптове е представен на Фигура 2.17. Функциите `gm_circ_approx`, `gm_cpoly`, `gmapply`, `gmmeshgen`, `gm_write` са част от езика на QMG. Скриптът първо генерира окръжност чрез приближение с помощта на пет криви на Безие, след което се очертава и вътрешен отвор. След това се стартира генериране на мрежата чрез `gmmeshgen` и запис на генерираната мрежа във файл чрез `gm_write`. Описаната геометрия отговаря на газоразрядната конфигурация на He-Cd лазер с външни надлъжни електроди, която се състои от две тръби: външна, изработена от кварц и вътрешна – от Al_2O_3. За простота описанието на външните електроди е пропуснато.

Както е показано в параграф 2.1.4, методът се базира на използването основно на два Java класа - `java.lang.Runtime` и `java.lang.Process`. Символният низ, който се подава като входен параметър на извикването `Runtime.exec` включва името на

приложението – matlab, и необходимите му аргументи – създадения QMG скрипт laser.m. Файлът с генерираната мрежа е показан на Фигура 2.18.

Обвиващият Java клас, съдържащ извикването `Runtime.exec`, лесно беше преобразуван в услуга, приемаща като параметър файл с B-rep описанието на геометрията на лазерното устройство. Уеб услугата беше създадена с помощта на Apache Axis Toolkit и разположена на Apache Tomcat сървлет контейнер.

Поради ограничеността на метода в сравнение с метода, базиран на Java Native Interface, представеното в този параграф не намери приложение в по-нататъшната работа по дисертацията и беше разгледано с цел пълнота на изследването.



Фигура 2.18 Резултати от генерирането на мрежа за зададената геометрия

2.2.4 Обвиване на базовата функционалност на PLASIMO – софтуерна система за симулиране на нискотемпературна плазма

Във връзка със създаването на среда за симулация на лазери с метални пари бяха осъществени контакти с групата по Елементарни процеси в газовия разряд към Факултета по приложна физика при Технологичния университет в Айндховен, Холандия, където повече от дванадесет години се разработва симулационния софтуер PLASIMO (PLAsma SImulation MOdel) [29]. PLASIMO е рамка за моделиране на източници на нискотемпературна плазма с възможности да се извършва както пълна, така и частична симулация. PLASIMO е написан на C++ под Linux с прилагане на всички парадигми на обектно-ориентирания дизайн, модулна организация, използване на шаблони на класове и функции, използване на стандартната библиотека с шаблони STL, и др . [3] PLASIMO предлага висока степен на възможност за конфигуриране на рамката от потребителя.

В резултат на осъщественото научно сътрудничество върху PLASIMO бяха приложени идеите и методите на работа, описани в предходните параграфи. Резултатите са публикувани в [26] и частично в [2].

Тук изцяло се прилага методът на обвиване с използване на Java Native Interface, разгледан подробно в 2.1.3. В резултат е създадена библиотека от Java обвиващи класове и съответния JNI „слепващ” код – библиотека от стъб функции, като са приложени техниките „съответствие едно-към-едно” (one-to-one mapping) и създаване на peer класове, описани в параграфите 2.1.3.5 и 2.1.3.6.

Представено е създаването на обвивки по метода „съответствие едно-към-едно” на функции, принадлежащи на пространството от имена `plRuntime` в PLASIMO. Пространството от имена `plRuntime` съдържа функции за конфигуриране на средата. В случая става обвиване на глобални PLASIMO функции за конфигуриране на средата, но прилагането на техниката „едно-към-едно” не зависи от това дали се обвиват глобални или член-функции на даден клас. Единственото изискване е за всяка функция, декларирана като нативна в обвиващия Java клас, да бъде създадена една JNI стъб функция.

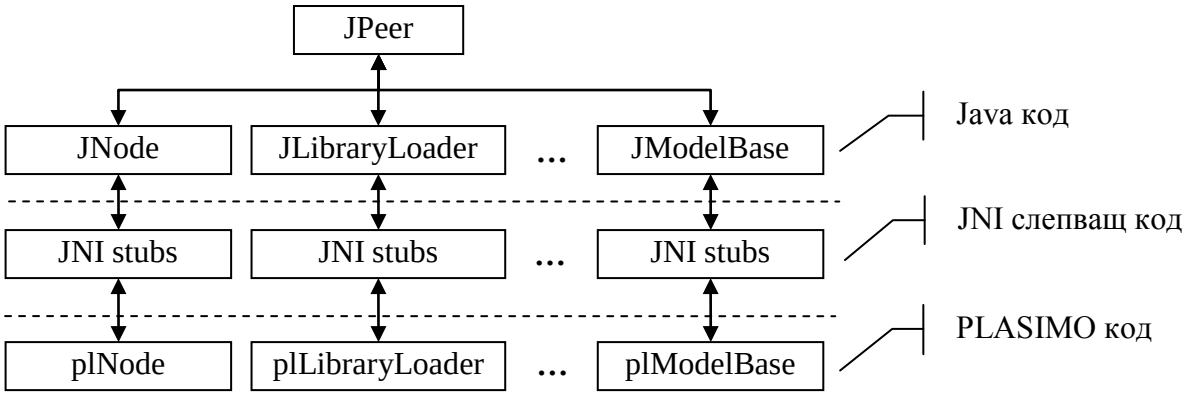
Чрез показаните реализации са разисквани и такива особености на интерфейса JNI като използване на JNI функциите, предоставящи възможности за отражение на типовете като GetObjectClass, GetSuperclass, GetMethodID, които са разгледани в параграф 2.1.3.1.

Друга важна особеност, която е представена, е начинът, по който се третират изключенията, възникнали в PLASIMO кода. Подходът, който е избран при обработка на изключения, възникващи в нативни PLASIMO функции, се изразява в следното: когато JNI кодът установи наличие на изключение, генерирано от PLASIMO кода, незабавно генерира Java изключение – в случая инстанция на класа JException, който беше създаден специално за справяне с тази ситуация. Това става чрез функцията `JNI_ThrowException`, разработена като помощна функция при обвиването на PLASIMO класовете.

В процеса на работа възникна необходимост да се използват C++ класове от Java програма, така че да имаме достъп до инстанциите на тези класове, докато приложението работи. Както беше показано в 2.1.3.6 обвиването на нативни структури от данни се извършва с помощта на peer класове, играещи ролята на прокси класове към C++ класовете. В процеса на обвиване на приложението PLASIMO беше създадена библиотека от Java peer класове, които съответстват на базови класове в PLASIMO. Част от тях са показани на Фигура 2.25.

Поради нарастващия брой peer класове, беше създаден абстрактен Java peer клас, задаващ базовата функционалност на всички peer класове в PLASIMO и съдържащ 64-битово поле, сочещо към съответната инстанция на дадения C++ клас.

Java peer класовете, които бяха създадени, имат същите методи като C++ класовете, които обвиват, като имплементацията на тези методи е да извикват съответстващите им C++ методи. Не е необходимо да има пълно съответствие между интерфейсите на PLASIMO C++ класа и съответния му Java клас - бяха обвивани само методи, които към дадения момент са били необходими. В течение на работата интерфейсът на един peer клас може да бъде допълван, ако такава необходимост възникне.



Фигура 2.25 Java peer класове, съответстващи на PLASIMO C++ класове

Накрая е представена UML диаграма на компоненти, показваща зависимостите между създаденото с тестови цели Java приложение JPlasimo, ползващо библиотеката с обвиващи Java класове, и множеството библиотеки за динамично зареждане. Част от тези компоненти са наследени PLASIMO библиотеки, които се използват без каквото и да е изменяне или повторно компилиране. Останалата част от компонентите са компилираните библиотеки с JNI слепващия код.

Глава 3. Приложение на архитектура, ориентирана към услуги, за научен софтуер

3.1. SOA – основни принципи, технологии, стандарти, реализация

Архитектурата, ориентирана към използването на услуги (Service Oriented Architecture, SOA), е относително нова концепция за изграждане на приложения, базирана на разработката, публикуването и консумирането на автономни софтуерни компоненти,

наречени услуги. SOA е стратегия и подход в информационните технологии, при който приложенията се възползват или зависят от услуги, предоставени в мрежа, например уеб. Реализирането на SOA може да включва изграждането на приложения, които ползват услуги, преобразуването на приложение в услуга, която други приложения да ползват, или и двете.

По-прецизно определение на термина „архитектура, ориентирана към услуги" е дадено в [28], където първо се дефинират термините архитектура и услуга както следва:

- **Архитектура** е формално описание на система, дефиниращо нейната цел, функции, външно видими свойства и интерфейси. Това включва също описание на вътрешните компоненти на системата и техните връзки заедно с принципите, определящи дизайна, операциите и по-нататъшното развитие на системата.
- **Услугата** е софтуерен компонент, който може да бъде достъпен през мрежа, за да предостави функционалност на клиента на услугата.
- Терминът **архитектура, ориентирана към услуги**, се отнася към изграждането на надеждни разпределени системи, които предоставят функционалност под формата на услуги, като се набляга на слабата свързаност на взаимодействащите софтуерни компоненти.

Услугата предоставя специфична функционалност, която може да е единична дискретна функция, или може да изпълнява множество от свързани бизнес функции. Във втория случай за услугата се казва, че е с **висока степен на гранулярност** (coarse grained). Множество услуги могат да бъдат използвани съвместно по координиран начин, формирайки съставна услуга, която да отговоря на по-сложни бизнес изисквания.

Съществени **предимства на SOA** са, че прилагането на този архитектурен подход води до повторна използваемост, гъвкавост, мащабируемост, взаимодействие.

3.1.1 Реализация на SOA чрез уеб услуги

SOA и уеб услугите са две различни неща, но уеб услугите са се превърнали в предпочитания, базиран на стандарти, начин за реализация на SOA. Самата SOA не дефинира точен стандарт за комуникация между услугите. Достатъчно е просто да се стъпи на някой от множеството такива – принципно могат да бъдат използвани RPC, DCOM, ORB и така нататък. Въпреки това, XML базираният стандарт за описание на уеб услуги (езикът WSDL) се е наложил като стандарт в тази архитектура.

3.1.2 Технология на уеб услугите

Приложението на уеб услугите зависи от това стандартите за уеб услуги да бъдат широко приети. За да е възможно това, тези стандарти и технологиите, които ги имплементират, трябва да отговарят на следните критерии:

- Уеб услугата трябва да може да приема заявки от всеки клиент, независимо от платформата, на която този клиент е имплементиран.
- Клиентът трябва да може да намира и използва всяка уеб услуга независимо от детайлите на имплементацията на тази услуга или от платформата, на която тя се изпълнява.

Стандартите за уеб услуги покриват такива области като:

- Използване на общ език за маркиране при комуникация - *eXtencible Markup Language (XML).*
- Използване на общ формат на съобщенията при обмен на информация - *Simple Object Access Protocol (SOAP)*].
- Използване на общ формат при описание на една услуга - *Web Services Description Language (WSDL).*
- Използване на общи средства за търсене на услуги - *Universal Description, Discovery, and Integration (UDDI).*

В допълнение към тези основни стандарти по-сложните уеб услуги, които имплементират широк спектър от бизнес процеси, се нуждаят от стандарти за: сигурност, транзакции, управление на последователността на процесите, обмен на съобщения.

Реализирането на уеб услуги зависи от наличните технологии, имплементиращи стандартите за уеб услуги. Големи индустриални производители като IBM, Oracle, Microsoft и SAP предоставят цялостни платформи за изграждане, разработка, изпълнение и управление на големи многослойни приложения, уеб услуги и SOA решение, сред които водещи са платформите Microsoft .NET и Java EE.

За целите на изграждането на научно приложение, базирано на използването на услуги, се прие платформата J2EE като основни причини за това са създаването на Java обвивки на наследен математически и физичен софтуер, както е показано в Глава втора и изискването да се използват свободни технологии и библиотеки. По тази причина са разгледани основните технологии за уеб услуги, интегрирани в J2EE.

3.1.3 Архитектура на уеб услугите. Начин на действие

Вътрешната структура на уеб услугата може да се раздели на два слоя: комуникационен слой и бизнес слой. Двата слоя работят заедно, за да позволят на приложенията да си взаимодействат. Разгледани са отговорностите на всеки един от слоевете и взаимодействието между тях.

Обикновено стандартите и технологиите за уеб услуги обхващат два основни типа шаблони за взаимодействия между приложения:

- за дистанционно извикване на процедури (Remote Procedure Call, RPC);
- документно ориентирани (document-oriented),

които също са разисквани в настоящия параграф.

3.1.4. Мотивация за използване на SOA, базирана на уеб услуги, за изграждане на научен софтуер в областта на физиката.

Мотивацията да бъде приложена SOA архитектура, базирана на услуги при изграждането на физичен софтуер и в частност при симулация на процесите, протичащи в лазерите с метални пари, е разгледана в няколко направления:

- Повторно използване на сложен математически и физичен код;
- Повишени изисквания към виртуалната физична лаборатория - разноплатформено разпределено приложение с достъп до най-новите технологии;
- Постигане на повторна използваемост, гъвкавост, мащабируемост и взаимодействие.

3.1.5. Езикът BPEL – основни характеристики, BPEL процеси

Може би най-важната характеристика на SOA, допринесла за нарастващата популярност на тази архитектура, е възможността относително лесно да се дефинират бизнес процеси като композиция на уеб услуги.

Езикът BPEL е един от семейството езици за композиция на уеб услуги и описание на бизнес процеси като замества предшествениците WSFL (създаден от IBM) и XLANG (създаден от Microsoft.). В момента BPEL е най-популярният език в тази област и е получил поддръжката на голямата част от софтуерната индустрия. Първата версия на BPEL се появява през август 2002 г, а през април 2007 г. е одобрена и публикувана като стандарт и версия BPEL 2.0.

BPEL е XML-базиран език, който се наложи като стандарт за композиция на множество синхронни и асинхронни уеб услуги в нови съставни уеб услуги, наречени *бизнес процеси* и изпълнявани от BPEL сървър.

3.1.5.1. Композиция на уеб услуги

Композицията на уеб услуги означава създаване на приложения от уеб услуги в качеството им на съществуващи и повторно използваеми изграждащи блокове.

Съществуват два различни подхода за композиция на уеб услуги, които определят как се контролира изпълнението на процеса: оркестрация и хореография.

От гледна точка на изпълнението на един процес, оркестрацията е по-гъвкавия механизъм и по тази причина е получил по-голяма популярност.

Езикът BPEL поддържа оркестрацията като дефинираният бизнес процес играе ролята на централен координатор. Крайният резултат от композицията на уеб услуги е отново уеб услуга, т.е. един бизнес процес се описва с езика BPEL, изпълнява се на BPEL сървър, но се предоставя на външния свят за ползване чрез WSDL интерфейса си както всяка друга уеб услуга.

3.1.5.2. Общи характеристики и изисквания към бизнес процесите

Основните характеристики на езика BPEL могат да се разглеждат само в контекста на основните характеристики на един бизнес процес изобщо:

- Потоци (workflows);
- Сътрудничество (collaboration);
- Транзакции (transactions);
- Изключения (exceptions);
- Събития (events);
- Асинхронни взаимодействия (asynchronous interactions);
- Съхранение на информация за състояние (process states);
- Запис на реализираните взаимодействия (audit traits);
- Споразумения (agreements);
- Сигурност на съобщенията (message security and reliability);

3.1.5.3. BPEL процес – основни характеристики и компоненти

Езикът BPEL е създаден специално за описание на бизнес процеси и по тази причина предоставя поддръжка на общите характеристики на един бизнес процес, които бяха разгледани по-горе. Някои от ключовите характеристики на BPEL са свързани с поддръжка на: транзакции с компенсация, области, обработка на изключения, корелация, манипулатори на събития, контейнери, паралелност и недетерминизъм, процедури.

Езикът BPEL поддържа два различни начина за описание на бизнес процесите:

- Изпълним процес – позволява да се специфицират точните детайли на бизнес процесите.
- Абстрактни бизнес протоколи – позволяват специфициране само на публичния обмен на съобщения между различните участници.

Основните компоненти на един BPEL процес са дейности, партньорски връзки и променливи, които също са разгледани в настоящия параграф.

3.2 Приложение за симулация на лазери с метални пари – създаване на BPEL процес за симулация

3.2.1 Приложение на езика BPEL за научни потоци

Спектърът на това, което може да бъде наречено научен работен поток (scientific workflow), е много широк и включва потоци за научни открития, потоци, които автоматизират ръчни процедури или правят повторен инженеринг на дадени инструменти или данни, както и потоци с интензивни изчисления. В настоящия параграф са представени редица от общите изисквания към един научен поток като: композиция и повторна употреба на услуги, мащабируемост, откъснато изпълнение, надеждност и отказоустойчивост, взаимодействие с потребителя, мониторинг, „умно” повторно стартиране, произход на данните и др.

Като цяло речникът на езика BPEL е приспособен към изискванията на бизнес процесите, към които най-общо има различни изисквания в сравнение с научните потоци. В повечето източници е дискутирано, че бизнес процесите се фокусират върху шаблони и събития, свързани с потока на управление, като често потокът данните има второстепенно

значение. При системите за научни процеси, от друга страна, съществува тенденцията те да имат модел на изпълнение, който е много по-ориентиран към потока на данните.

BPEL спецификацията е разгледана в контекста на изброените изисквания към един научен процес, в резултат на което в настоящия параграф е направено заключението, че BPEL е напълно приложим за оркестрация на научни услуги. В допълнение BPEL може да служи като стандартно представяне на един научен поток и по този начин да способства за по-голяма степен на възпроизводство. Резултатите са отпечатани в [25].

3.2.2 Изграждане на BPEL процес за симулация на разпределението на потенциала и интензитета на електричното поле

3.2.2.1 Постановка на задачата

Един от най-важните параметри на разряда е интензитета на електричното поле E. Неговата стойност влияе директно на електрическия ток, разпределението на мощността, температурния профил на газовия разряд, енергията на електроните и др.

С цел изследване на различни физични процеси трябва да бъде извършена симулация на разпределенията на скаларния потенциал и интензитета E на електричното поле. За получаване на скаларния потенциал в напречното сечение на обемния разряд, е необходимо да се реши двумерното квазистационарно уравнение на Поасон

$$(1) \qquad \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -\frac{e}{\varepsilon_0}(n_i(x,y) - n_e(x,y))$$

при следните смесени гранични условия

$$(2) \qquad \varphi_L = 0, \qquad \varphi_R = U_a \ ,$$

$$(3) \qquad \varphi_\Gamma = 0 \, ,$$

$$(4) \qquad \varepsilon_i E_{in} = \varepsilon_j E_{jn} \, ,$$

където ε_0 е диелектрична константа, $n_i(x,y)$ и $n_e(x,y)$ са съответно концентрациите на йони и електрони. Граничните условия (2) показват, че към левия електрод е приложено напрежение, равно на нула, а при десния електрод напрежението е U_a. Граничното условие (3) се определя от факта, че газоразрядната тръба най-често се поставя в метален корпус, който от изисквания за безопасност е занулен и има потенциал нула. Условието (4) отразява критерия за преход на интензитета на полето на границата на два разнородни материала с диелектрични константи ε_i и ε_j.

За намиране на интензитета на електричното поле е необходимо да се използва израза $\overline{E} = -grad\, U$.

Симулацията се извършва посредством вариране на работните параметри на лазера.

3.2.2.2 BPEL процес за симулация

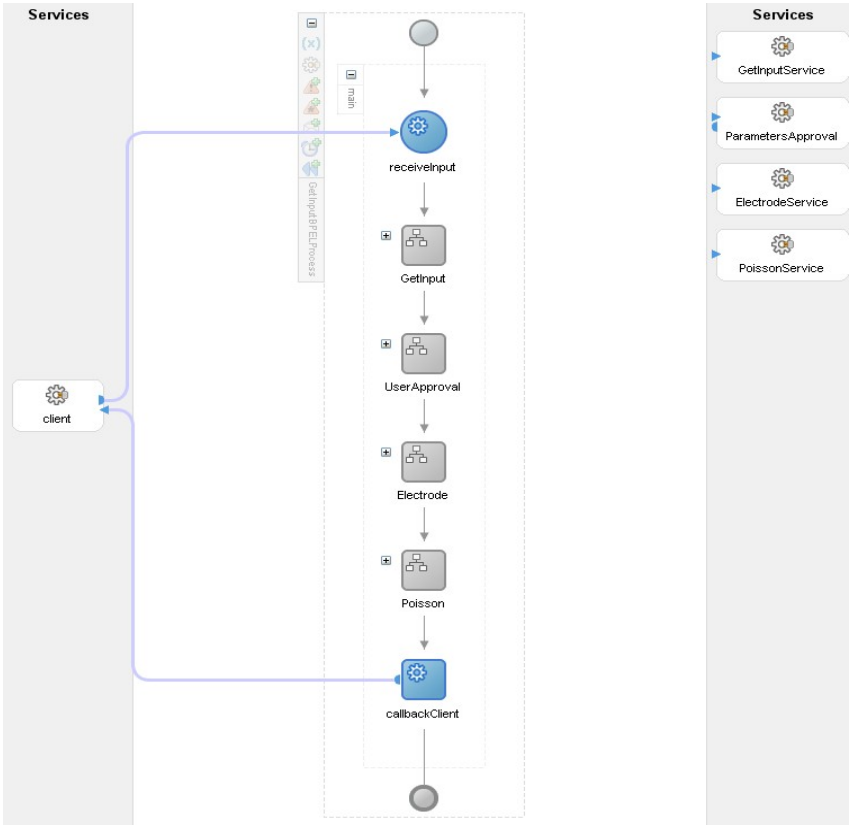
В този параграф е представено изграждането на прототип на BPEL процес за симулация на разпределенията на потенциала и интензитета на електричното поле в съответствие с описаната постановка на задачата. Резултатите са отпечатани в [21].

Представеното в настоящия параграф е надграждане на практическите резултати, представени в параграф 2.2.2.

Графично представяне на BPEL процеса е дадено на Фигура 3.6. Уеб услугите, които бяха създадени и композирани в BPEL процес, са: *GetInputService*, *ParametersApproval*, *ElectrodeService* и *PoissonService*.

Описана е последователността на работния поток: получаване на входящо съобщение от клиента, което стартира изпълнението на процеса на симулация; извличане на избрания тип на лазер като символен низ от входния XML файл; синхронно извикване на уеб услугата *GetParameters*, която връща съответното множество от параметри за това конкретно лазерно устройство. Показани са генерираната SOAP заявка за стартиране на

BPEL процеса, както и генерираният SOAP отговор на заявката за извличане на множеството параметри за заявения тип лазерно устройство.



Фигура 3.6 Графично представяне на изградения BPEL процес на симулация

Множеството от параметри представлява група от свързани параметри (физични константи, променливи) с добавени мета данни, описващи различни аспекти на параметрите, като помощна информация, валиден диапазон (напр. минимална, максимална стойност и др.), стойност по подразбиране, дали параметърът е задължителен и т.н.

Параметрите се съхраняват в XML файлове. Разработена е и съответната XML схема за описание на типовете на параметрите и за валидиране на XML документите с параметри. Следователно, всеки от тези XML документи е инстанция на XML схема документ.

На Фигура 3.9 е представена част от XML схемата, използвана за дефиниране на множеството параметри. Дефинирани са два основни типа:

- `parameterset` - представя множеството параметри;
- `parameter` – абстрактен тип, който се наследява от различните типове.

Дефинирани са също така и допълнителни абстрактни типове, които наследяват `parameter`, като `quantity` (за числови параметри за количество), `array` (за параметри от тип масив) и други. Чрез наследяване на базовите типове са дефинирани разнообразни типове като int, intEnum, double, doubleEnum, string, stringEnum, intArray, doubleArray и т.н., като за всеки от тях е добавена подходящата мета информация.

Дефинирането на множество от параметри чрез XML схема направи възможно да се работи както с различни множества от параметри, съответстващи на различни типове лазери, така и с различни инстанции на множеството параметри за даден лазерен тип.

След получаването на множеството параметри, процесът на симулация стартира уеб услугата *ParametersApproval* за човешка намеса (human task). Целта на тази уеб услуга е спиране на процеса на симулация, докато параметрите със техните зададени стойности не бъдат одобрени. Тук потребителят може да промени някои от предварително дефинираните стойности на физични константи, геометрични параметри и т.н.

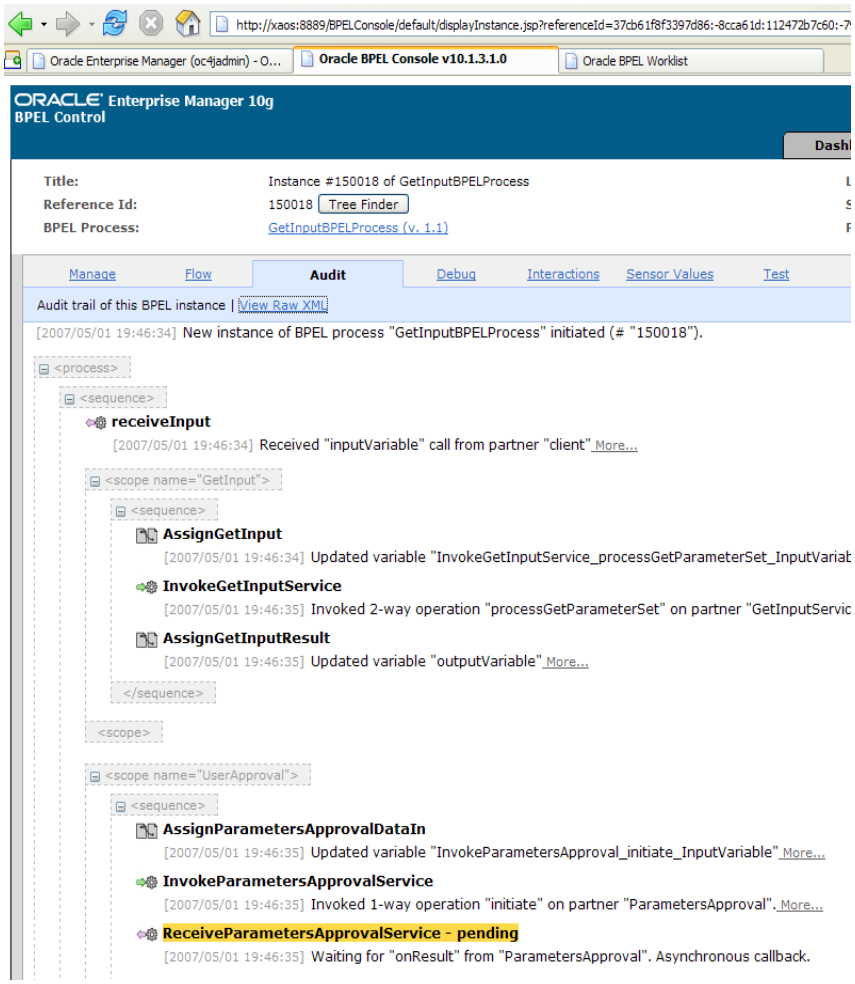
```
  <complexType name="parameter" abstract="true">
    <sequence>
      <element name="name" type="string"/>
      <element name="required" type="boolean"/>
      <element name="prompt" type="string"/>
      <element name="help" type="string"/>
    </sequence>
  </complexType>
   <!--
 ********************************************************************************
 The abstarct type for numeric quantity parameters:
    units: (optional) string containing the unit of measure of the quantity
 ********************************************************************************
  -->
  <complexType name="quantity" abstract="true">
    <complexContent>
      <extension base="sdoc:parameter">
        <sequence>
          <element name="units" type="string" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!--
 ********************************************************************************
Used to define int parameters. Inherits all the elements of param and adds
the following:
    default: the value of the parameter to be used if unspecified
    stringdefault: a non-numeric string inicating a "special" default(e.g."all")
    min: (optional) minimum value for the parameter
    max: (optional) maximum value for the parameter
    validValues: (optional) enumeration of valid values for the parameter
 ********************************************************************************
 -->
  <complexType name="int" >
    <complexContent>
      <extension base="sdoc:quantity">
        <sequence>
          <element name="default" type="int" minOccurs="0"/>
          <element name="min" type="int" minOccurs="0"/>
          <element name="max" type="int" minOccurs="0"/>
          <element name="validValues" type="sdoc:intEnum" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

Фигура 3.9 Отрязък от създадената XML схема за дефиниране на типовете на параметрите на симулацията

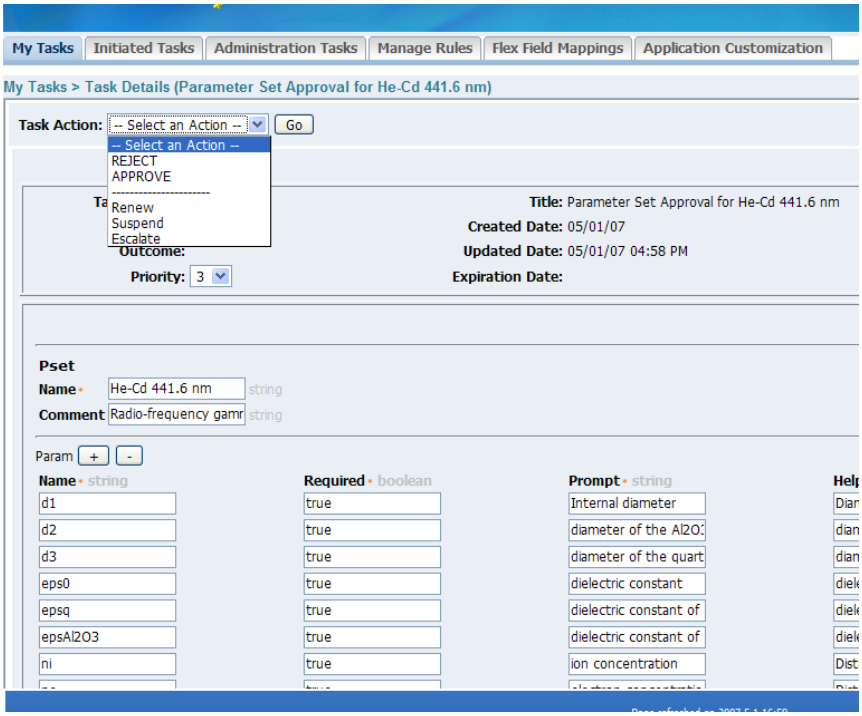
Изпълнение на инстанция на разработения BPEL процес за симулация е представена на Фигура 3.10. BPEL процесът е инициирал задачата за човешка намеса и е в състояние на изчакване тази задача да завърши, получавайки обратно обновените данни (множеството параметри) при завършване на задачата – както се вижда от Фигура 3.10 дейността *receiveParametersApprovalService* е висяща в очакване на отговор.

На Фигура 3.11 е представен уеб интерфейсът на приложението Oracle BPEL Worklist Application, което е част от BPEL сървъра Oracle BPEL Process Manager. Услугата *ParametersApproval* очаква одобрение или промяна стойностите на параметрите, както се вижда от Фигура 3.11.

Услугата *ParametersApproval* връща като резултат множеството параметри с техните окончателни стойности, след което следва синхронно извикване на уеб услугата *ElectrodeService*. Това е наследен FORTRAN компонент, обвит чрез JNI и предоставен като уеб услуга. Този компонент генерира подходящата мрежа за дискретизация и класифицира точките в различните под-области: външни електроди, лазерна тръба и т.н. Основни входни параметри за услугата *ElectrodeService* са подадените напрежения на електродите, които след това се използват за получаване на граничните условия.



Фигура 3.10 Инстанция на BPEL процеса на симулация в състояние на изчакване на отговор от задачата за човешка намеса



Фигура 3.11 Уеб интерфейс към Oracle Worklist Application в процес на изпълнение на задачата за човешка намеса – услугата *ParametersApproval*

Последната услуга от работния поток, която бива инициирана от BPEL процеса, е услугата *PoissonService*. Това също е наследен FORTRAN модул, който е обвит чрез Java Native Interface, както е описано в Глава 2, и предоставен като уеб услуга за композиране в разработения процес на симулация. Модулът се използва за пресмятане на разпределението на потенциала и интензитета на електричното поле чрез решаването на двумерното квзистационарно уравнение на Поасон. Услугата генерира като резултат двумерни таблици от данни, които се записват в текстови файлове.

3.3 Приложение при софтуерната система за симулиране на нискотемпературна плазма PLASIMO

Основните задачи, които бяха поставени при изграждането на прототип на уеб-базиран вариант на PLASIMO (WebPlasimo), се развиха в две посоки:

1) Създаване на уеб – интерфейс към съществуващата рамка за симулация на нискотемпературна плазма PLASIMO.
2) Публикуване на функционалност на PLASIMO под формата на уеб услуги за ползване от други научни екипи, включително и създаване на тестови клиенти на публикуваните уеб услуги.

И двете задачи предполагат създаване на Java обвивки на базови PLASIMO модули с цел повторно използване и достъп до най-съвременни уеб-технологии, реализацията на което е разгледано подробно в Глава 2, параграф 2.2.4.

3.3.1 Използвани технологии при създаването на прототипа WebPlasimo. Основни характеристики на WebPlasimo

При създаването на изграждания прототип на WebPlasimo са приложени основно следните технологии:

- Apache Struts 2
- Dojo
- Apache Axis 2

Тези технологии са разгледани, като е дискутирано и отражението им върху основните характеристики на WebPlasimo.

Както сървърната част на прототипа WebPlasimo, така и създадените тестови уеб услуги са разположени на сървлет контейнера Apache Tomcat .

3.3.2 Компоненти и функционалност на прототипа WebPlasimo

Представени са основните компоненти, изграждащи прототипа WebPlasimo:

- уеб-базиран клиент с богат потребителски интерфейс, създаден чрез технологията Dojo;
- сървърна част, създадена чрез технологията Apache Struts 2;
- Java обвивки на наследения C++ PLASIMO код;
- Java обвивки на наследения PLASIMO код, публикувани като уеб услуги.

Разгледана е детайлно функционалността на изградения уеб-базиран прототип WebPlasimo. Разгледани са разликите при варианта с извикване на уеб услуги.

За целите на разработения прототип са създадени **прототипи на инструменти**, които са представени в следващите параграфи:

3.3.3 Генератор на XML файлове

Създаденият прототип на инструмент за генериране на първоначален XML файл на базата на зададена XML схема има за цел да улесни създаването на конфигурационните файлове, съдържащи необходимите за една симулация входни физични константи, геометрични параметри, набор от библиотеки за динамично свързване и т.н.

Инструментът се базира на реализацията на Apache на спецификацията XML Schema API, която е публикувана като препоръка на консорциума W3C от март 2004 год. Спецификацията е имплементирана в Apache Xerces2 Java Parser като библиотека от класове и интерфейси с отворен код.

Инструментът има следните характеристики:

- Възможност за обработване на схеми с дефинирани повече от един коренов елемент – за тази цел при подаването на конкретния XML Schema файл се изисква и задаване на името на корена;
- Обработване на сложни типове;
- Обработване на прости типове;
- Генериране на елементи;
- Генериране на атрибути;

Крайният XML документ, който инструментът създава, е валидна XML инстанция на зададената схема като съдържа всички задължителни елементи, включително вложените им под-елементи, задължителните атрибути, евентуално стойностите по подразбиране, фиксираните стойности, съблюдава кратността на елементите и подредбата на елементите. Това дава възможност да се съкрати значително времето за създаване на входния за симулацията файл, което е от изключително практическо значение, тъй като по принцип процесът на конфигуриране на симулацията е продължителен и трудоемък.

3.3.4 Уеб – базиран XML редактор

Предназначението му е да предостави лесен и удобен потребителски интерфейс за редактиране на XML документи. За целите на Уеб-базирания XML редактор са създадени два допълнителни спомагателни инструмента:

- **Инструмент за генериране на графичен-потребителски интерфейс –** целта на инструмента е да създаде допълнително ниво на абстракция между XML DOM дървото на обработвания XML файл и реалния JavaScript код, който генерира интерфейса за редактиране на елемент. Допълнителното ниво на абстракция дава възможност да се автоматизира частично и улесни генерирането на потребителски интерфейс. Предложеният подход се състои в следното:
  - групиране на примитивните XML типове, до по-обобщено множество от фиктивни типове;
  - съпоставяне на всеки от присвоените типове данни на тип на графичен контрол от създаденото за целта множество от абстрактни типове на контроли;

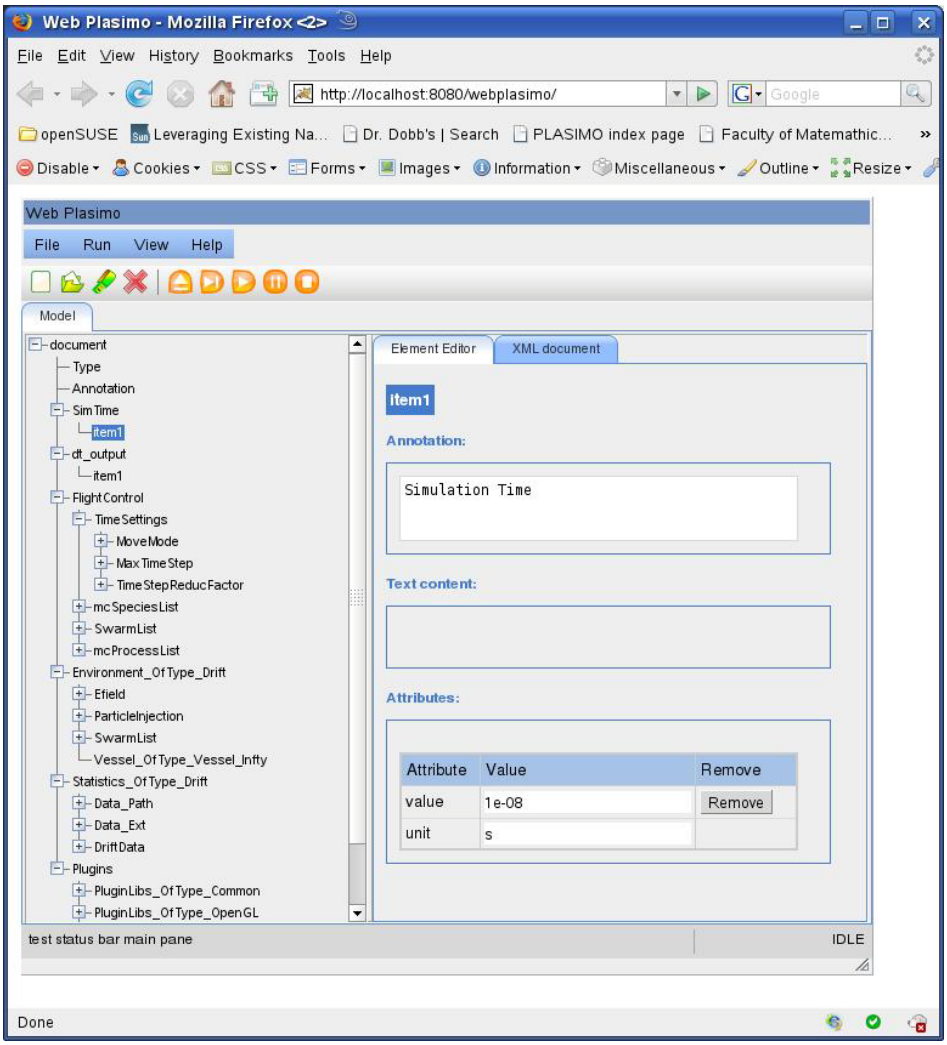
  Предложеният подход предстои да бъде доуточняван в процеса на доразвиване на прототипа WebPlasimo.
- **Повторно използваема библиотека от помощни функции за обработка на XML файлове –** за нуждите на XML редактора е създадена библиотека от помощни функции за обработка на XML документи. Тези функции могат да бъдат използвани повторно и извън контекста на прототипа WebPlasimo.

На Фигура 3.13 е показан прототипът WebPlasimo с общ изглед на инструмента Уеб-базиран XML редактор. Входните параметри за инструмента могат да бъдат следните:

- документът, създаден от инструмента за генериране на първоначални XML файлове. В този случай редакторът се използва за цялостно изграждане на входния за симулацията файл (команда File/New);
- създаден и попълнен отпреди XML файл за конфигуриране на симулацията и XML схемата, на която този документ е инстанция. В този случай редакторът се използва за доуточняване или промяна на някои от входните физични параметри (команда File/Open).

Инструментът се състои от три секции, както това е показано на Фигура 3.13:

Секция Element Editor: предоставя потребителски интерфейс за редактиране на елемент и за визуализиране на анотации (Фигура 3.13). Тук е предоставена следната функционалност:



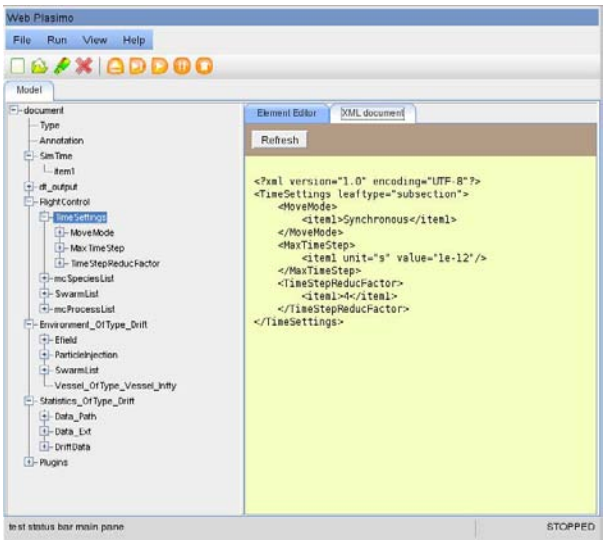
Фигура 3.13 Прототипът WebPlasimo – общ изглед на инструмента Уеб-базиран XML редактор

- възможност за редактиране на текстовото съдържание на даден елемент в съответствие със зададения в схемата тип на съдържание на елемента (Content type - EMPTY, ELEMENT, MIXED, SIMPLE);
- възможност за добавяне и изтриване на атрибут;
- възможност за редактиране на стойност на атрибут;
- визуализиране за на зададените в XML схемата анотации.
- валидация на стойност на елемент/атрибут според типа, зададен в XML схемата;

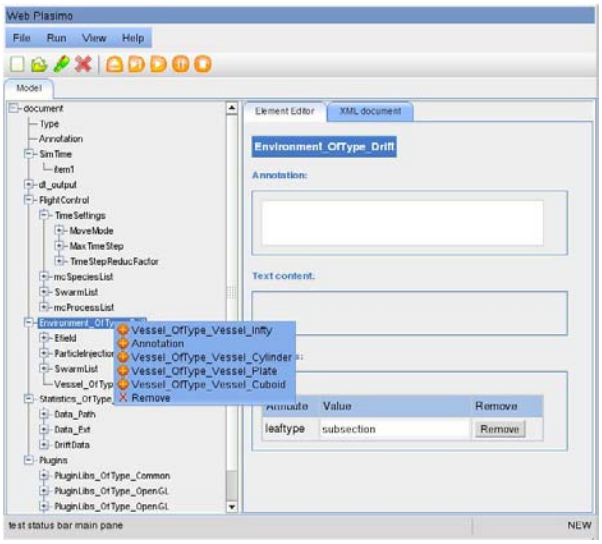
Секция XML document: предоставя визуализиране на XML съдържанието на конфигурационния за симулацията файл. Дадена е възможност за визуализиране както на целия XML документ, така и само на отделен негов възел, който е посочен в Dojo дървото, изобразяващо редактирания XML документ, както това е показано на Фигура 3.15.

Секция Model: изобразява редактирания XML документ чрез Dojo дърво. Предоставя контекстно меню за добавяне и изтриване на елемент (Фигура 3.16) като:

- елемент се добавя заедно в всичките му под-елементи, които не са опционални, стойностите по подразбиране, фиксираните стойности, атрибутите, както това е зададено в съответната XML схема.
- при добавянето на елемент се следи точното местоположение за вмъкване според XML схемата и текущото DOM дърво на документа.



Фигура 3.15 Прототипът WebPlasimo – визуализиране на XML съдържание



Фигура 3.16 Прототипът WebPlasimo – визуализиране на XML документ чрез Dojo дърво и добавяне на елемент

3.3.5 Контролер на изпълнявания модел за симулация

Един базовите класове за модели за симулация в PLASIMO, класът plModelBase, задава интерфейс за управление на изпълнението на модела:

- подготовка на модела;
- стартиране на модела;
- обновяване на изходните данни – при постъпково изпълнение;
- запис на статистически данни за изпълнението на модела - например времетраене, брой итерации и т.н.

За този базов клас е създаден Java обвиващ клас, класът JModelBase, чрез Java Native Interface, както е описано в Глава 2, параграф 2.2.4. Класът JModelBase повтаря интерфейса на plModelBase, но за целите на едно уеб приложение, каквото е прототипа WebPlasimo, се наложи създаването на контролер на изпълнявания модел.

Контролерът на модела е реализиран както в сървърната част на приложението, така и в клиентския код (JavaScript реализация) като целта е максимална синхронизация между двата контролера чрез опресняване на състоянието през определен интервал. Контролерът в сървърната част е реализиран като паралелно приложение с цел да се даде възможност за натрупване в опашка на събитията „натискане на бутон", пристигащи от графичния интерфейс на приложението, както и управление на последователността на състоянията, през които минава изпълнявания модел. Поради естеството на Интернет комуникацията между клиента и сървъра, обаче, е възможно в браузера да бъдат пропуснати някои междинни данни.

В резултат, прототипът WebPlasimo предоставя следните команди (менюто Run или лентата с инструменти на Фигура 3.13):

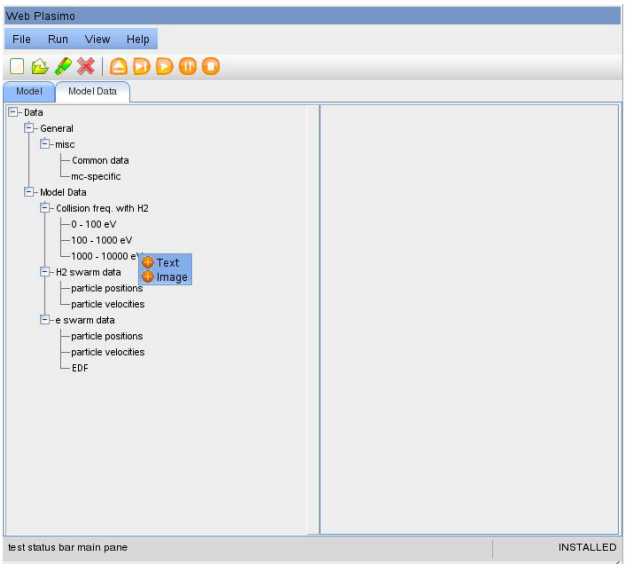
- инсталиране на модел – *Run/Install*
- стартиране на модел - *Run/Start*
- изпълнение на една стъпка - *Run/Step*
- пауза - *Run/Pause*
- спиране на изпълнението на модела - *Run/Stop*

Горната функционалност формира множеството от команди {INSTALL, STEP, RUN, PAUSE, STOP}, които инициират преминаването на контролера през различни състояния, формиращи множеството от възможни състояния {NEW, INSTALLING, INSTALLED, STEPPING, IDLE, RUNNING, PAUSED, STOPPED, ERROR}.

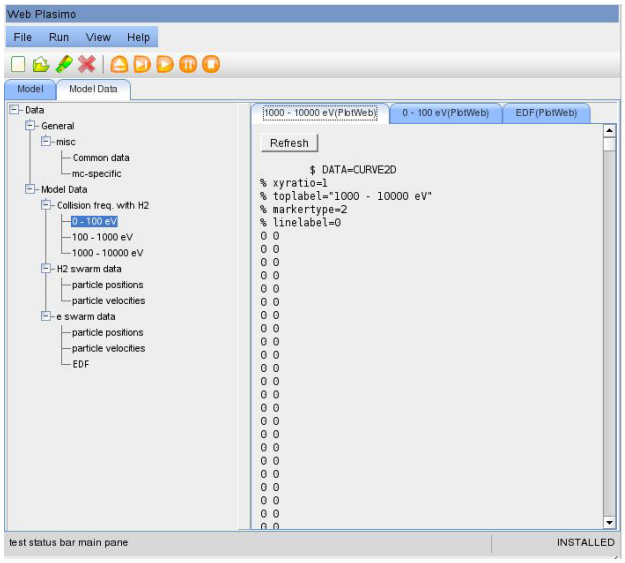
3.3.6 Инструменти за визуализиране в уеб на резултатите от симулацията

Към настоящия момент са изградени два инструмента за изобразяване на данни в уеб – в текстов и в графичен вид. Създадените инструменти се саморегистрират при инстанциирането им за конкретен тип данни чрез рамката за саморегистриране на обекти, която е разработена в PLASIMO.

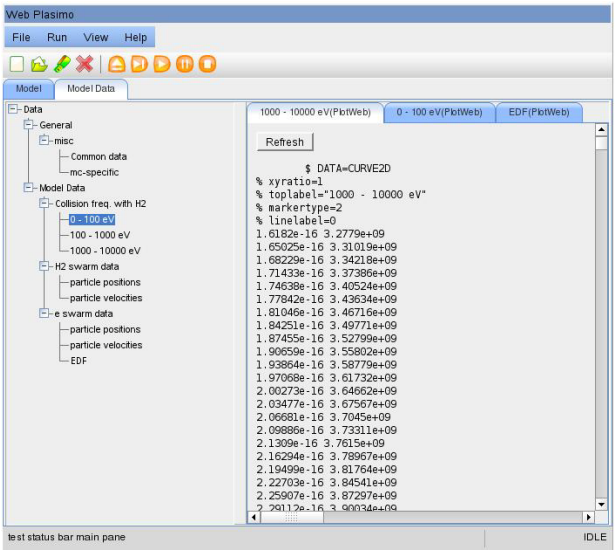
Данните, които участват в симулационен софтуер като PLASIMO, идват от различни източници: xy-графики, таблици, символни низове или обикновени числа. За всеки тип данни е подходящ определен набор от възможни представяния и съответно за тях се регистрират различни инструменти за визуализиране. Някои от инструментите извеждат данните в конзола, други в диалогови прозорци, трети във вид на OpenGL графики и т.н.



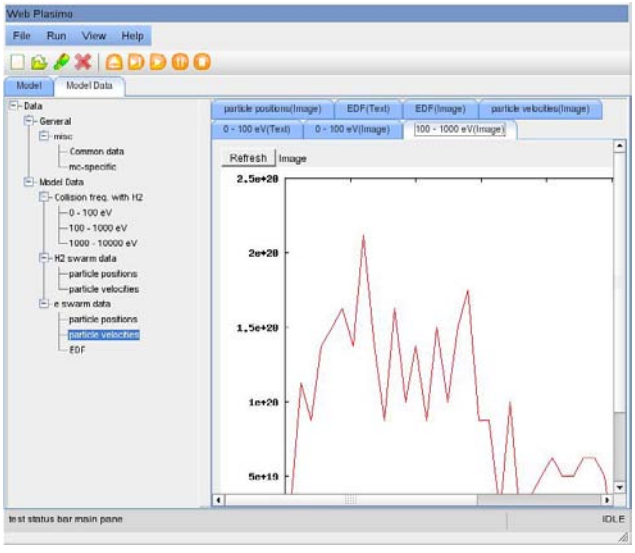
Фигура 3.18 Избор на инструмент за визуализиране за даден тип данни



Фигура 3.19 Подготовка на прототипа за изобразяване на три избрани типа данни



Фигура 3.20 Прототипът след една стъпка от изпълнението на симулацията



Фигура 3.21 Прототипът след една стъпка от изпълнението – графичен вид

В PLASIMO е изградена рамка, която позволява такива множества от данни от произволен тип да бъдат представени на потребителите под формата на списък с унифициран интерфейс. Идеята зад този дизайн е процесът на визуализиране на данни да бъде обобщен до няколко елементарни абстрактни операции, като например (на псевдо - език) *Data::CreateViewByName(name)* и *View::Update*. Такава функционалност е

предоставена чрез абстрактни базови C++ класове. За целите на прототипа WebPlasimo са създадени следните C++ класове като разширение на рамката за визуализиране:

- `plXnYDataWebViewer` – за визуализиране на `double` и `plComplex` данни от вида `plXnYData`, използван за манипулиране на структури от данни, които асоциират единичен масив от *x* стойности с единичен или двумерен масив от *y* стойности.
- `plWebViewerConfig`, `plWebViewerContext`, `plWebViewerSupport` за конфигуриране на инструментите за изобразяване на данни в уеб.

На тези класове са създадени Java обвивки под формата на Java peer класове, както и необходимия междинен „слепващ" JNI код (виж Глава 2, параграф 2.2.4).

При решение да се работи с други типове данни, ще е необходимо създаването на допълнителни класове.

На Фигура 3.18 е показан прототипа WebPlasimo след инсталирането на избрания модел за симулация чрез подаването на подготвения чрез XML редактора входен конфигурационен файл. В резултат е създадена табелата "Model Data", в която в дървовидна структура са показани всички типове на данни, които подлежат на изобразяване за дадения модел.

На Фигура 3.19 е представена подготовката за визуализиране на избрани от потребителя типове данни. В случая са избрани типовете данни *„Collision freq. with H2/0-100 eV", „Collision freq. with H2/100-10000 eV", „e swarm data/EDF"* като за всеки избран тип данни е създадена съответна табела в графичния потребителски интерфейс.

На Фигура 3.20 е показан прототипа WebPlasimo след една стъпка от изпълнението на модела за симулация и след извършено опресняване на данните в активната табела. На Фигура 3.21 е представено съответното извеждане на данните чрез инструмента за визуализиране в уеб на данни в графичен вид. Освен постъпковото изпълнение и проследяване на данните е възможно и непрекъснато изпълнение на модела. При това отново има възможност за опресняване на визуализираните данни, но това става в произволни моменти от симулацията.

3.3.7 Уеб услуги

Различните варианти на приложение на уеб услугите, както и на свързаните с тях технологии, върху софтуера за симулация на нискотемпературна плазма PLASIMO, са отпечатани в [2]. Там са дискутирани основно два аспекта на приложение:

1) осигуряване на точки за достъп, базирани на уеб услуги, до индивидуални изграждащи блокове или до PLASIMO като цяло.
2) обработка на физични данни (физични константи, напречни сечения и т.н.), така че на крайния потребител да се предоставят освен „сурови" физични данни, но и пресметнати на базата на други такива.

По отношение на първия аспект са създадени и тествани две уеб услуги:

- **ConvertNodeWS** – предоставя две операции за конвертиране на входни файлове:
  - за преобразуване на входен PLASIMO файл за конфигуриране на симулацията в XML формат.
  - за преобразуване на XML файл за конфигуриране на симулацията във входен PLASIMO формат.
- **SpawnAllViewersWS** – предоставя единствена операция, която след приключването на симулацията връща списък от крайните резултати за всички възможни типове данни, за които е регистриран избрания инструмент за визуализиране. Резултатите се предоставят в текстов вид като таблици от числа. няма възможност за постъпково изпълнение или пауза на модела на симулацията, тъй като създадената уеб услуга не поддържа състояние.

Услугите са реализирани като POJO класове чрез технологията Apache Axis2. За всяка услугите, отново чрез Axis2, е създаден и тестов конзолен клиент. Освен това,

услугите могат да бъдат извиквани и от прототипа WebPlasimo като вариант на уеб-базиран клиент на уеб услуга.

Създадените уеб услуги са базирани на реализираните чрез Java Native Interface обвивки на аналогичната функционалност в PLASIMO, както това е показано в Глава 2. На практика клиентът на PLASIMO кода не комуникира директно с нативния код, а с уеб услугата, обвиваща нативните модули.

Перспективи

Бъдещото развитие на софтуерната система за симулация на лазери с метални пари може да се разгледа в следните аспекти:

- Продължаващо изследване на методите за обработка и споделяне на физични данни:
  - На крайния потребител да се предоставят освен „сурови" физични данни, но и пресметнати на базата на други такива;
  - Изследване на възможностите сървърите за физични данни да предоставят вторично пресметнатите физични данни под формата на уеб услуги. Идеи в тази насока вече бяха представени и дискутирани при участие в конференцията по Атомни и молекулярни данни и техните приложения, която се проведе в Пекин, Китай от 28 - 31 октомври 2008 г.
  - Доусъвършенстване на създадената XML схема, задаваща типовете на физичните константи и параметри, участващи в симулацията на лазери с метални пари чрез разработения BPEL процес.
- Разработване на асинхронни уеб услуги, поддържащи съхранение на състоянието чрез възможностите за поддръжка на сесии на наличните рамки, имплементиращи SOAP спецификацията. Целта е постигане на възможност за предоставяне на пълно функционална симулация като уеб услуга, позволяваща включително и постъпково изпълнение.
- Разработване на по-голямо количество уеб услуги, предоставящи частични резултати, подобни на вече създадените в прототипа Web Plasimo. Тези уеб услуги могат да се използват и извън контекста на симулация на лазери с метални пари за други физични цели. Например, предвижда се публикуването като уеб услуги на Болцман солвър и на код за пресмятане на Абелеви инверсии, които са част от симулационния софтуер PLASIMO, разработван в Техническия университет в Айндховен, Холандия.
- Преобразуване на разработения инструмент *Уеб–базиран XML редактор* до конзола за тестване на уеб услуги, която би била съществено улеснение при разработване на архитектури, ориентирани към използването на услуги.

Апробация

Резултати, получени в изследването, са използвани в следните национални и университетски проекти:

- Научен проект № 03М28 към звено „Научна и приложна дейност" на ПУ на тема: „Моделиране на лазери с метални пари и взаимодействието им с веществото", 2003-2004.
- Научен проект № 05М47 към звено „Научна и приложна дейност" на ПУ на тема: „Числено изследване на характеристиките на високочестотна плазма", 2005-2006.
- Национален научен проект на тема „Разработка на софтуерни продукти за компютърно симулиране на физичните процеси в газовия разряд с приложение в малки и средни предприятия" към Фонд „Научни Изследвания" на МОН, конкурс "Стимулиране на научните изследвания в държавните висши училища", договор ВУ - МИ -205/2006, (3 год.).

- Научен проект № 07М07 към звено „Научна и приложна дейност” на ПУ на тема: „Математическо и информационно моделиране на импусно-периодични и газови лазери”, договор 07М07, 2007-2008.
- Научен проект ИС-М-4 към звено „Научна и приложна дейност” на ПУ на тема: „Междуфакултетен разпределен център за електронно обучение”, 2008-2009.

Част от резултатите, получени в дисертационния труд, са докладвани на следните международни и национални конференции:

- IX International Conference on Laser & Laser Information Technologies & V International Symposium on Laser Technologies & Lasers ILLA/LTL 2006, Smolyan, Bulgaria, October 4-7, 2006.
- III International Bulgarian-Turkish Conference “Computer Science”, Istanbul, Turkey, October 12-15, 2006.
- V International Conference on Information Research and Applications i.Tech, Varna, Bulgaria, June 26-30, 2007.
- 3rd Balkan Conference in Informatics BCI’07, Sofia, Bulgaria, 27-29 September, 2007.
- VI International Conference on Information Research and Applications i.Tech, Varna, Bulgaria, June 23 - July 03, 2008.
- 6th International Conference on Atomic and Molecular Data and Their Applications (ICAMDATA), Beijing, China, October 28-31, 2008.

Част от идеите и концепциите, представени в дисертацията, са приложени върху софтуера за симулация на нискотемпературна плазма, разработван от групата по Елементарни процеси в газовия разряд към Факултета по приложна физика в Техническия университет в Айндховен, Холандия. Внедряването на резултатите е на ниво прототип и е описано в Глава 2 и Глава 3, като са направени и съответните изводи. Резултатите са отпечатани в двете съвместни публикации с колегите от Холандия и Софийския университет.

Изградените инструменти и прототипи са използвани в процеса на обучение по дисциплина „*Въведение в създаването на уеб услуги*”, която се провежда във ФМИ към ПУ (доц. д-р Антон Илиев, гл.ас. Анна Малинова).

Авторска справка

Основните приноси на дисертационния труд са:

1. Реализирани са процедури за интегриране на съществуващ физичен софтуер, като са анализирани и описани приложимите налични технологии и стандарти, съобразно характеристиките на наследения симулационен софтуер.
2. Проектиран е обвиващия код, като е анализирана приложимостта на някои добре известни обектно ориентирани практики и образци за проектиране.
3. Създадени са обвивки на различни наследени модули с цел повторно използване и интегриране в разработваната софтуерна система за симулация на лазери с метални пари.
4. Предложена е и е реализирана архитектура, ориентирана към използването на услуги, позволяваща съвместна работа на нови и наследени (собствени и чужди) модули, обвити като уеб услуги.
5. Направен е анализ на BPEL спецификацията от гледна точка на използването на езика BPEL за изграждане на научен процес за симулация.
6. Реализиран е BPEL процес за решаване на основната моделна задача за компютърна симулация на разпределенията на потенциала и интензитета на електричното поле, които са основа за по-нататъшни пресмятания на параметрите на газовия разряд в лазерното устройство.

7. Реализиран е прототипът WebPlasimo – уеб-базиран вариант на рамката за симулация на нискотемпературна плазма PLASIMO. В резултат се решава по-общата задача за симулация на плазма, частен случай на която е симулацията на процесите, протичащи в лазерите с метални пари.
8. Проектирани са и са създадени инструментални програмни средства, обслужващи различни етапи на компютърната симулация, на базата на най-съвременни уеб технологии и рамки за разработка на приложения в съответствие с избраната архитектура.
9. Проектиран е и е реализиран удобен потребителски интерфейс за въвеждане на данните на симулацията със и без допълнителна обработка, както и за представяне на резултатите от симулацията в текстов и графичен вид.
10. Разработени са уеб услуги за основни етапи на симулацията, като са анализирани перспективите на развитие в тази посока.

Изказвам сърдечна благодарност на научния си ръководител доц. д-р Снежана Гочева-Илиева за съветите и подкрепата и на колегите от групата по Елементарни процеси в газовия разряд към Факултета по приложна физика в Техническия университет в Айндховен, Холандия за възможността да приложа концепциите и идеите от този труд върху техния код.

Публикации по дисертационния труд

1. Malinova A. A., S. G. Gocheva-Ilieva, I. P., Iliev, *Wrapping Legacy Codes for Numerical Simulation Applications*, Proceedings of the III International Bulgarian-Turkish Conference Computer science, Istanbul, Turkey, October 12-15, 2006, Part II, pp. 202-207, 2007.

2. Malinova A. A., S. G. Gocheva-Ilieva, I. P. Iliev, *Web Services-based simulation of metal vapour lasers*, Proceedings of ILLA ‘2006 - IX International Conference on Laser and Laser-information Technologies: Fundamental Problems and Applications and LTL ‘2006 – V Intern. Symp. Laser Technologies and Lasers, Smolyan, Bulgaria, October 4-7, 2006, pp. 315-323, April 2007.

3. Malinova A. A., S. G. Gocheva-Ilieva, *Application of the Business Process Execution Language for building scientific processes for simulation of metal vapor lasers*, Proceedings of the 3rd Balkan Conference in Informatics, Sofia, Bulgaria, 27-29 September, 2007, Volume 2, pp.75-86, 2007.

4. Malinova A. A., S. G. Gocheva-Ilieva, *Using the Business Process Execution Language for managing scientific processes*, International Journal “Information Technologies and Knowledge” (IJITK), vol. 2, N3, p. 257-261, 2008.

5. Malinova A., V. Yordanov, J. van Dijk, *Leveraging existing plasma simulation codes*, International Book Series "Information Science & Computing", Number 5, pp.136-142, Supplement to the International Journal "Information Technologies & Knowledge", Volume 2/2008.

6. Dijk J., A. Malinova, V. Yordanov, J. van der Mulen, *New Interfaces for the Plasimo Framework*, 6th International Conference on Atomic and Molecular Data and Their Applications (ICAMDATA), Beijing, China, October 28-31, 2008. Conference Proceedings of American Institute of Physics - http://proceedings.aip.org/resource/2/apcpcs/1125/1/176_1?isAuthorized=no).

7. Malinova A., *Design approaches to wrapping native legacy codes*, Scientific works, Plovdiv University, vol. 36, Book 3, 2008 (под печат).

Литература

[1] Boisvert R. F., P. Tak, P. Tang, *The Architecture of Scientific Software*: IFIP Tc2/WG2.5 Working Conference on the Architecture of Scientific Software, October 2-4, 2000, Ottawa, Canada, Springer, 2001.

[2] Dijk J. van, A. Malinova, V. Yordanov, J. van der Mullen, *New Interfaces for the Plasimo Framework*, Proceedings of the 6th International Conference on Atomic and Molecular Data and Their Applications (ICAMDATA), Beijing, China, October 28-31, 2008. Conference Proceedings of American Institute of Physics - http://proceedings.aip.org/resource/2/apcpcs/1125/1/176_1?isAuthorized=no).

[3] Dijk J. van, *Modeling of Plasma Light Sources: an object oriented approach*, PhD thesis, Eindhoven University of Technology, The Netherlands, 2001, http://plasimo.phys.tue.nl/publications/vandijk2001.pdf

[4] Elliot J. Chikofsky and James H. Cross II, *Reverse engineering and design recovery: A taxonomy*, In Robert S. Arnold, Software Reengineering, pages 54–58. IEEE Computer Society Press, 1992.

[5] Gocheva-Ilieva S.G., I. P. Iliev, *Mathematical modeling of the electric field in copper bromide laser*, Proc. of ICNAAM 2007, vol. CP936, American Institute of Physics (AIP), pp. 527-530, 2007.

[6] Iliev I. P., S. G. Gocheva-Ilieva, A. A. Malinova, N. V. Sabotinov, *Three criteria for studying the breakdown in radio-frequency discharge in nitrogen*, Proceedings of ILLA '2006 - IX Intern. Conf. Laser and Laser-information Technologies, Smolyan, Bulgaria, October 4-7, 2006, pp. 324-331, April 2007.

[7] Iliev I. P., S. G. Gocheva-Ilieva, A. A. Malinova, *Simulation of radio-frequency weak-current nitric discharge*, Proceedings of the IV International Symposium on Laser Technologies and Lasers - 8.10.-11.10. 2005, LTL2005, Plovdiv, pp. 250-255, 2006.

[8] Iliev I.P. , S. G. Gocheva-Ilieva, *On the application of the multidimensional statistical techniques for exploring copper bromide vapor laser*, CP1067, American Institute of Physics, 2008, 475-482.

[9] Iliev I.P., S. G. Gocheva-Ilieva and N. V. Sabotinov, *Classification analysis of the variables in CuBr laser*, Quantum Electron., vol. 39, N2, 143-146, 2009.

[10] Iliev I.P., S. G. Gocheva-Ilieva, D. N. Astadjov, N. P. Denev and N. V. Sabotinov, *Statistical analysis of the CuBr laser efficiency improvement*, Optics and Laser Technology, vol. 40, issue 4, pp. 641-646, 2008.

[11] Iliev I.P., S. G. Gocheva-Ilieva, D. N. Astadjov, N. P. Denev and N. V. Sabotinov, *Statistical approach in planning experiments with a copper bromide vapor laser*, Quantum Electron., vol. 38, N 5, 436-440, 2008.

[12] Iliev I.P., S. G. Gocheva-Ilieva, K.A. Temelkov, N.K. Vuchkov and N. V. Sabotinov, *Analytical model of the temperature in UV CU+ lasers*, CP1067, American Institute of Physics, 2008, 114-121.

[13] Iliev I.P., S. G. Gocheva-Ilieva, N. V. Sabotinov, *Analytic study of the temperature profile in a copper bromide laser*, Quantum Electron., vol. 38, N 4, 2008, 338-342.

[14] Iliev I.P., S. G. Gocheva-Ilieva, N. V. Sabotinov, *Modelling of radio-frequency breakdown in argon*, J. of Optoelectronics and Advanced Materials, vol. 11, 10, pp. 1392-1395, 2009.

[15] Iliev I.P., S. G. Gocheva-Ilieva, *Statistical techniques for examining copper bromide laser parameters*, Proc. of ICNAAM 2007, American Institute of Physics (AIP), vol. CP936, pp. 267-270, 2007.

[16] Iliev I.P., S.G. Gocheva-Ilieva, N. V. Sabotinov, *Prognosis of the Copper Bromide Laser Generation through Statistical Methods*, XVII Int. Symp. Gas Flow and Chemical Lasers & High Power Lasers 2008, Lisbon, 15 - 19 September 2008, Proc. SPIE, ed. by R.Vidal et al., vol.7131, SPIE, Bellingham, WA, pp. 71311J1-J8, 2009.

[17] Iliev I.P., S.G.Gocheva-Ilieva, N.V.Sabotinov, *Improved model of the gas temperature in copper bromide laser*, Quantum Electron., vol. 39, N 5, pp. 425-430, 2009.

[18] Java Language Specification, http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html

[19] Kurziniec D., V. Sunderam, *Efficient cooperation between Java and Native Codes – JNI*

Performance Benchmark, Proceedings of the 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001), 2001.

[20] Malinova A. A., S. G. Gocheva-Ilieva, *Application of the Business Process Execution Language for building scientific processes for simulation of metal vapor lasers*, Proceedings of the 3rd Balkan Conference in Informatics, Sofia, Bulgaria, 27-29 September, 2007, Volume 2, pp.75-86.

[21] Malinova A. A., S. G. Gocheva-Ilieva, I. P., Iliev, *Web services – based simulation of metal vapor lasers*, Proceedings of the IX International Conference on Laser & Laser Information Technologies & V International Symposium on Laser Technologies & Laser ILLA/LTL '2006, Smolyan, Bulgaria, October 4-7, 2006, pp. 315-321.

[22] Malinova A. A., S. G. Gocheva-Ilieva, I. P., Iliev, *Wrapping legacy codes for Numerical simulation applications*, Proceedings of the III International Bulgarian-Turkish Conference Computer science, Istanbul, Turkey, October 12-15, 2006, Part II, pp. 202-207, 2007.

[23] Malinova A. A., S. G. Gocheva-Ilieva, *Using the Business Process Execution Language for managing scientific processes*, International Journal "Information technologies and Knowledge", Vol. 2/2008, pp. 257-261.

[24] Malinova A., V. Yordanov, J. van Dijk, *Leveraging existing plasma simulation codes*, International Book Series "Information Science & Computing", Number 5, pp.136-142, Supplement to the International Journal "Information Technologies & Knowledge", Volume 2/2008.

[25] Malinova A., *Design Approaches to Wrapping Native Legacy Codes*, Scientific works, Plovdiv University, vol. 36, Book 3, 2008 (под печат).

[26] OGSA Glossary Terms v 1.5: http://www.ogf.org/documents/GFD.81.pdf

[27] PLASIMO simulation software, http://plasimo.phys.tue.nl

[28] QMG 2.0 Mesh generation software, http://www.cs.cornell.edu/home/vavasis/qmg2.0/qmg2_0_home.html

[29] Stoyanova-Doycheva A., *A Software Refactoring Process and the Supported Tools*, Fourth Int. Conf.Informatics and Information Technology, Bitola, Macedonia, 11-14 Dec 2003.

[30] Wilson S., J. Kesselman , *Java Platform Performance: Strategies and Tactics*, Prentice Hall, 2001, http://java.sun.com/docs/books/performance/1st_edition/html/JPTitle.fm.html