СУ “Св. Климент Охридски”
Факултет по Математика и Информатика
Катедра “Информационни технологии”

Дипломна работа

Тема:
“Управление на курсовите задачи в системата за дистанционно обучение ARCADE (Architecture for Reusable Courseware, Authoring and Delivery)”

Дипломант : Диана Иванова Мутафчиева
Специалност: Електронен Бизнес
Факултетен номер: M21235

Научен ръководител: доц. д-р Боян Бончев

София, 2005

СЪДЪРЖАНИЕ

1. Въведение

Дистанционното обучение е нова форма, която все повече навлиза в нашето ежедневие като начин на обучение. В съвременния динамичен и напрегнат живот този метод на обучаване става все по-привлекателен. Това е процес, който предоставя информация и възможност за обучение във време, място и форма, удобна за участниците в процеса. При дистанционното обучение участващите в процеса – обучаващи и обучавани участват по различно време и от различно място. Този начин на обучение предоставя възможност за по-гъвкаво и по-рационално разпределение на времето. Учебните материали се съхраняват в електронен вид. Съхраняването и предаването на информацията в този вид са възможни благодарение на съвременните информационни технологии и глобалната мрежа Интернет. Информацията в електронен вид може да бъде достъпна за много хора едновременно. Дистанционното обучение предоставя още възможност за обучение с индивидуално темпо както и многократно използване на даден курс. Наред с предимствата този метод на обучение има разбира се и недостатъци. Сред тях са недостатъчната комуникация между участниците в процеса на обучението, асинхронният начин на обучение – знанията се предават и приемат от участниците по различно време, липсва учител до ученика. Обучаващите от своя страна имат необходимост от по-строг самоконтрол.

1.1. Цел

В рамките на проекта ARCADE (Architecture for Reusable Courseware Authoring and Delivery), разработван към катедра “Информационни технологии” на Факултет по математика и информатика на Софийски университет “Св. Климент Охридски”, е реализирана съвременна платформа за дистанционно обучение. Основните функции на системата включват управление на курсове, учебна програма, тестове и задания, комуникация между потребителите.

Целта на конкретната дипломна работа е усъвършенстване и разработка на допълнителни възможности към съществуващия модул за задания в ARCADE. Това са:

- ✓ Реализация на разпределяне на заданията “отдолу нагоре” – възможност студентите сами да избират заданията си за самостоятелна работа съгласно зададени от инструктора ограничения – срокове, максимален брой студенти и др.
- ✓ Създаване и управление на календарни събития в системата ARCADE, описващи процеса на дистанционно обучение, като например задаване на задачи, тестове, и др.
- ✓ Възможност студентите сами да предлагат теми за заданията за самостоятелна работа в ARCADE, които да бъдат обсъждани и одобрявани или отхвърляни от инструктора.

Тези възможности ще бъдат проектирани посредством унифицирания процес за разработка на софтуер и езика UML. Като програмен език за реализация ще се използва езикът Java.

1.2. Ключови потребители

Потребители на платформата ARCADE са студентите и инструкторите в една учебна организация. Такава организация в момента е Софийският университет “Св. Климент Охридски”. Различните потребители изпълняват различни функции, като правата им са строго определени и ограничени Най-многочислена е групата на студентите. Основните потребители на платформата ARCADE са системен администратор, курсов администратор, инструктор и студент. Има и пети потребител, гост, който има достъп до функциите, достъпни за всички потребители – лични данни, дискусии, новини, поща, чат и др. Управлението на курсовете (регистриране, модифициране, изтриване) се извършва от курсовия администратор. От друга страна инструкторът допълва курсовете с подходящи материали, определя датите за записване и провеждане на курса и го представя на студентите. Последните разглеждат материалите, свързани с конкретен

курс, следят важните дати в развитието на курса, явяват се на тестове, предават задания и биват оценявани. За разглежданите възможности, които ще бъдат реализирани в текущата разработка, основна роля играят потребителите Инструктор и Студент.

1.3. Полза от реализацията

Предимството от разработката на тези допълнителни възможности в системата ARCADE е по-голямата свобода за студентите при тяхното оценяване за дадена дисциплина, възможността сами да разработват задания, с което да обогатяват съответните курсове, които посещават.

1.4. Структура на дипломната работа

Дипломната работа се състои от 7 раздела – увод, три глави за изложение, заключение, списък на използваната литература и публикации.

Глава 1: настоящето кратко представяне на проекта

Глава 2: общи характеристики на системите за дистанционно обучение, преглед на някои други платформи, преглед на проекта ARCADE

Глава 3: използвани технологии

Глава 4: детайлно описание на целия процес на проектиране и разработка на описаните възможности, обект на разглеждане на дипломната работа – потребителски изисквания, анализ, дизайн, имплементация, тестване

Глава 5: обобщаване накратко цялостното съдържание на дипломната работа и нейния принос

Глава 6: използвана литература, тук може да бъде намерена допълнителна информация за предметната област, за конкретните технологични решения и др.

Глава 7: публикации по дипломната работа, съдържа информация за направената публикация по работата

2. Платформи за дистанционно обучение. Проект ARCADE

Все повече университети и корпорации разработват и внедряват в експлоатация прототипи на платформи за дистанционно обучение. Също така расте и броят на потребителите им. В момента на пазара съществуват много платформи за online курсове – WebCT, TeleTop, WebAssign, BlackBoard и др. Всички те си приличат по структура и модули, като осигуряват сходна функционалност. Всяка една система за дистанционно обучение трябва да предлага следните базови възможности:

- ✓ Управление на потребителите – системата трябва да поддържа различни видове потребители и съответни нива на достъп; различните потребители изпълняват различни функции по време на работа на системата; едни се грижат за правилната работа на цялата система, други подготвят процеса на обучение, трети го реализират; най-многобройна обикновено е групата на обучаваните
- ✓ Създаване на Web базирани курсове – системата трябва да предлага набор от средства и инструменти за изграждане на online курсове; всеки създаден курс трябва да може да се показва под различна форма на различните обучавани, в зависимост от тяхното ниво, участниците трябва да имат възможност да четат различна информация от курса
- ✓ Управление на курсовете – системата трябва да дава възможност за четене на курсове от различни участници, тематично групиране на курсовете, определяне на зависимости на един курс от друг
- ✓ Проследяване на процеса на обучение – разпределяне на курсовете във времето, създаване на график, определяне на срокове за преминаване през определен курс и т.н.
- ✓ Тестване – възможност за оценяване на придобитите знания от страна на участниците; трябва да се предлагат възможности за

създаване, провеждане и оценяване на тестове и задания, свързани с курсовете

- ✓ Комуникация – също много важен модул за платформите за дистанционно обучение; системата трябва да предоставя възможност за комуникация между отделните потребители; това може да става чрез изпращане на съобщения, участие в дискусии, обмяна на данни и др; комуникацията може да бъде два вида в зависимост от начина на провеждането й във времето – синхронна и асинхронна.

2.1. Платформа за дистанционно обучение WebCT

WebCT е комерсиална система за управление на курсове [3]. Тя предоставя богат набор от инструментални средства. Използва се в множество университети в целия свят. С WebCT може да се направи цял курс или само избрани функции като календар, бюлетин и форум за дискусии, чат, електронна поща, търсене, речник, инструменти за оценяване - например такива за създаване и управление на задания, както и за самотестване. WebCT позволява проследяване на дейността на студентите, управление на оценките и осигурява място на студентите, където те да създават собствени web страници. Накратко WebCT е лесно персонализируем web сайт за дистанционни курсове. Когато инструкторът поиска WebCT акаунт, той или тя получава достъп до сайта на WebCT с набор от образователни инструменти, които улесняват ученето, комуникацията и сътрудничеството между инструктора и неговите или нейните студенти. Предоставен е и набор от инструменти за администриране, които да подпомагат инструкторите в управлението на техните курсове. Инструментите на WebCT са вградени в сайта, което означава че студентите и инструкторите имат достъп до функциите на WebCT чрез свързване към Интернет, използвайки правилно конфигуриран браузър и избиране и активиране, за да се използват инструментите. Никакъв софтуер с изключение на браузър

не е нужен, за да се проектира или ползва един WebCT курс. Платформата предоставя следните възможности:

- ✓ съхраняване на оценки и разглеждане на статистики за успеха на студентите
- ✓ студентите имат възможност да разглеждат своите оценки
- ✓ създаване на връзки към допълнителни материали, свързани с курса
- ✓ участниците в образователния процес могат да общуват помежду си чрез съобщения, поща или чат
- ✓ създаване и предлагане на студентите задания и тестове с автоматично оценяване
- ✓ поддръжка на виртуални кутии, където студентите могат да представят резултатите от своите задачи по електронен път

Заданията в системата WebCT представляват курсови проекти, които се раздават от преподавателя и се приемат по електронен път. WebCT поддържа следната функционалност, свързана със заданията. Инструкторът има възможност да опише заданието (курсовия проект), както и неговата максимална оценка. Установяват се дата и час за представяне на заданието от студентите. Могат да бъдат прикрепени файлове, съдържащи изображения и таблици към заданието. Инструкторът може да оцени и коментира предаденото задание. Заданията в системата WebCT имат следните атрибути: заглавие, инструкции, максимална оценка, начална дата за представяне на заданието, крайна дата за предаване на заданието, информация, показваща дали студентът ще получи известие след получаване на заданието, информация, показваща дали инструкторът ще получи известие, след като студентът предаде резултата от заданието, информация, показваща дали оценката ще бъде видима или не за студента. При оценяването на заданията от инструктора той вижда датата на предаване на заданието, името на студента, максималната възможна оценка, файловете с резултата от заданието. След преглеждане на резултата инструкторът нанася оценката и предоставя пояснителен

коментар за решението си. По всяко време той има възможност да нулира резултата на студента или направо да изтрие цялото задание.

2.2. Платформа за дистанционно обучение WebAssign

WebAssign представлява online система за управление на тестове и задания за домашна работа [4]. Тя позволява да се създават задания за домашна работа, тестове от готова за използване база данни от въпроси от учебници или да бъдат създавани и оформяни авторски материали от преподавателя. Преподавателят има пълен контрол над домашната работа, която студентът получава, включително датата на представяне, съдържанието, вида на обратната връзка и формата на въпросите. Той може да назначава, събира, оценява и записва заданията мигновено. Тестовете се генерират със случайно избрани номера на въпросите, за да се създаде уникален набор за всеки обучаем. Системата позволява проследяване на представянето на студентите, управление на оценките и индивидуално отлагане на крайната дата на предаване, както и максималния брой предавания за всяко задание поотделно. WebAssign може да работи с повечето операционни системи и браузъри, съществуващи на пазара в момента. Заданията за домашна работа в системата WebAssign имат следните атрибути:

- ✓ име – показва се на студентите и трябва да бъде уникално и информативно
- ✓ описание – появява се при студента, в обобщения списък на заданията и в самото задание, добре е описанието да включва кратко резюме на заданието и условията, които се поставят на студентите, които го изпълняват
- ✓ инструкции – тук могат да бъдат дадени различни помощни материали, инструкции, връзки към източници на информация
- ✓ въпроси – тук преподавателят може да избере какви въпроси да включи в заданието, от кои учебници

- ✓ настройки за администрация на заданието – период, през който студентите могат да предават резултатите от заданията си, брой позволени предавания на заданието, тип, категория на заданието (домашна работа, тест, изпит), вид метод на случайност за формиране на въпросите, кой от предадените варианти, ако са повече от един, ще бъде оценен от инструктора, автор на заданието – най-често текущият потребител, който ползва системата или оригиналният автор на заданието

Демонстрационна версия на платформата може да бъде видяна на адреса http://www.webassign.net/info/demo.html.

2.3. Платформа за дистанционно обучение Moodle

Платформата Moodle представлява софтуерен пакет с отворен код за създаване на интернет базирани курсове и web сайтове [5]. Съкращението произлиза от наименованието Модулна обектно ориентирана динамична среда за обучение – Modular Object-Oriented Dynamic Learning Environment. Системата непрекъснато се развива. Предоставя възможност за управление на сайта, управление на потребители, курсове, задания и тестове, модули за чат и форум. Модулът за задания предоставя следните възможности:

- ✓ заданията могат да имат крайна дата и максимална оценка
- ✓ студентите могат да качват своите задания на сървъра, файловете могат да бъдат във всякакъв формат
- ✓ задания могат да се предават и след крайната дата, като дните на закъснение се виждат от преподавателя
- ✓ по преценка на преподавателя заданията могат да бъдат предавани втори път за втора по-добра оценка
- ✓ към оценката преподавателят може да добави коментар, след оценяването на някой студент, системата автоматично му изпраща email с известие за оценката

Атрибутите на заданията в Moodle са следните:

- ✓ име на заданието – произволен низ от символи
- ✓ описание на заданието – има възможност за всякакво форматиране на текста – различни шрифтове и стилове, използване на изображения и т.н.
- ✓ тип на заданието – системата предоставя възможност заданията да бъдат предавани online чрез качване на файлове или offline чрез предаване на хартиени копия; при online предаване на заданието се специфицира максимален размер на предавания файл
- ✓ начална и крайна дата на заданието
- ✓ информация, показваща дали заданието може да бъде предадено втори път за по-висока оценка
- ✓ информация, показваща дали заданието може да бъде предавано след крайната му дата
- ✓ максимална оценка – предоставена е възможност за различни типове на оценяване

Демонстрационна версия на системата може да бъде видяна на адреса http://moodle.org/login/index.php, като потребителско име и парола се използва *teacher*.

2.4. Проект ARCADE

ARCADE (Architecture for Reusable Courseware Authoring and Delivery) – архитектура за създаване и многократно използване на курсове – представлява проект, разработван от екип от изследователи и студенти към катедра Информационни технологии, Факултет по Математика и Информатика на СУ “Св. Климент Охридски”. Разработката е базирана на езика UML (унифициран език за моделиране – Unified Modelling Language) и унифицирания процес за разработване на софтуер – The Unified Software Development Process [1]. Характеристики на системата са:

- ✓ Поддържане на различни типове потребители и строго съблюдаване на техните права на достъп. Това са студенти, инструктори, системни администратори и т.н.

- ✓ Предоставяне на удобни средства за създаване и публикуване на курсове.
- ✓ Възможност за създаване на тестове и задания към курсовете и автоматизирано тестване, също и оценяване на обучаващите се.
- ✓ Средства за комуникация между участващите в процеса на обучението – вътрешна поща, система за новини, дискусии, чат и др.
- ✓ Отвореност и лесна разширяемост на системата.
- ✓ Преносимост на системата на различни платформи (Windows или UNIX базирани компютърни мрежи).
- ✓ Използване на платформата с различни браузъри, налични на пазара.

Системата поддържа пет различни типа потребители – Студент, Инструктор, Автор на курсове, Курсов Администратор и Системен Администратор.

Студент. Това е всеки човек, който използва ресурсите за обучение, за да придобие знания или умения. Всеки студент може да участва в повече от един регистриран курс и може да достъпва материалите на курса. Студентите са организирани в групи от студенти. Студентите имат възможност да:

- ✓ Влизат в системата след въвеждане на потребителско име и парола
- ✓ Променят своите лични данни
- ✓ Търсят и преглеждат регистрираните курсове и да показват информация за тях
- ✓ Записват курсове, обявени за публично достъпни, и да получат известие за регистрация
- ✓ Разглеждат материалите за всички курсове, за които те са се записали
- ✓ Получават информация и да участват в оценяването на своето представяне
- ✓ Предоставят и изтеглят материали от пространството, свързано с курса

- ✓ Комуникират с други участници с помощта на средствата за комуникация, зададени за съответния курс

Инструктор. Подпомага Студентите по време на процеса на обучение. Негова задача е да създава инстанции на курс (персонализиране на даден курс според специфичните нужди на обучение на студентите с определен профил) от даден курс и да определя и приспособява средата на курса. Инструкторите имат възможност да:

- ✓ Дефинират стратегията за движение за курса чрез използване на инструмент, предоставен от системата
- ✓ Избират шаблон за визуалното представяне на инстанцията на курса
- ✓ Описват средствата за комуникация, налични за тази инстанция на курс
- ✓ Определят програмата на инстанцията на курса
- ✓ Задават процедурите за оценяване – създаване и добавяне на тестове, определяне на задания, които не са включени в първоначалното описание на курса, преглед на резултатите от заданията и оформяне на окончателната оценка за студентите
- ✓ Разменят съобщения с други участници на курса
- ✓ Експортират материалите, предадени от студентите във формат, който позволява тяхното използване извън системата

Автор на курсове. Този потребител е отговорен за създаване и модифициране на курсове според предварително дефиниран модел. Авторите на курсове имат възможност да:

- ✓ Създават и модифицират запис с мета данни за курс според съществуващи стандарти
- ✓ Описват структурата на курс
- ✓ Предоставят материали за курс, като качват ресурси
- ✓ Експортират курса във формат HTML, за да може да бъде използван извън системата

Авторът на курсове всъщност представлява частен случай на потребителя Инструктор. Всеки инструктор може да бъде и автор на курс.

Курсов Администратор. Този тип потребител управлява курсовете, инстанциите на курсове и програмите. Курсът съдържа основните материали за обучение за инстанциите, които са базирани на него. Няколко различни инстанции на курс могат да са базирани на един и същи курс. Програмата е връзка между инстанциите на курсове, групите студенти и инструкторите. Курсовите администратори имат възможност да:

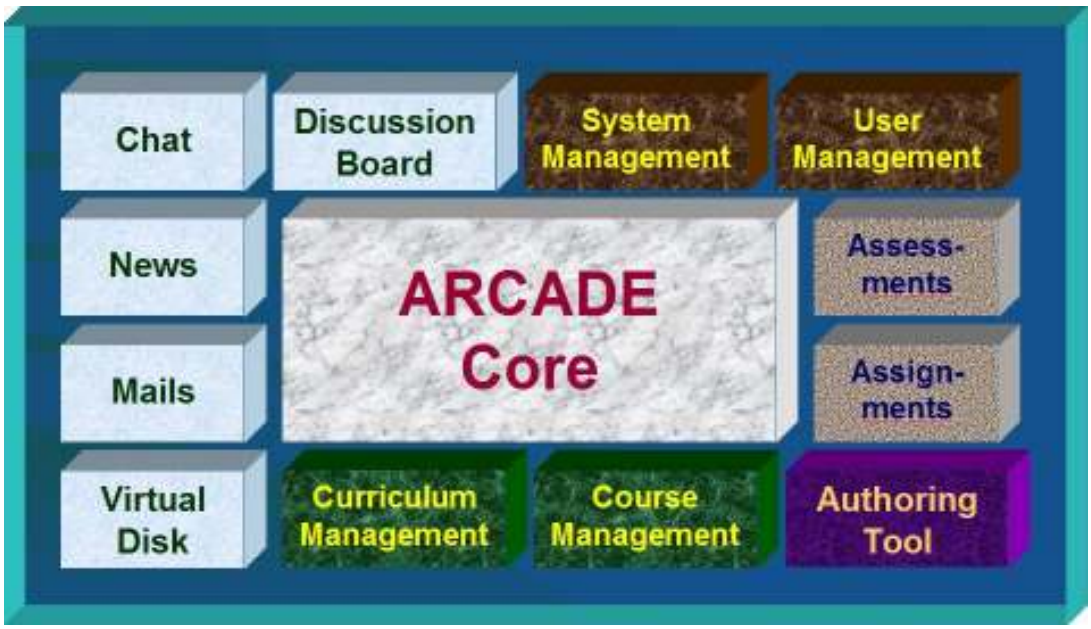
- ✓ Регистрират и изтриват курсове и инстанции на курсове в и от системата
- ✓ Определят дали дадена инстанция на курс е отворена или ограничена, т.е. за нея се изисква заплащане
- ✓ Записват и отписват студенти и групи от студенти за и от инстанции на курсове
- ✓ Създават и управляват програми
- ✓ Управляват групите студенти и да определят инструктор за всяка група
- ✓ Изпращат съобщения до други потребители на системата чрез използване на вътрешната поща

Системен Администратор. Има пълен контрол над системата, управлява системните ресурси като потребителски акаунти и групи от потребители, назначава права и определя настройките за системата, управлява и следи за сигурността на системата. Системните администратори имат възможност да:

- ✓ Добавят и модифицират лични данни, както и да изтриват потребители
- ✓ Създават, редактират и изтриват групи потребители – един потребител може да участва в повече от една група
- ✓ Назначават права на потребители и групи от потребители за всеки обект в системата (курс, инстанция на курс, тест, чат, дискусии и т.н.)
- ✓ Управляват файловете със записани събития, определят кои събития да се записват, да преглеждат и изтриват тези файлове

- ✓ Изпращат съобщения до други потребители на системата, използвайки вътрешната поща

Архитектурата на ARCADE съдържа пет модула. На фигура 2.4.1 могат да бъдат видяни отделните модули.



Фигура 2.4.1. Архитектура на системата ARCADE

- ✓ Управление на системата и потребителите – включва по-голямата част от функциите на системния администратор: управление на потребителите и групите от потребители, назначаване на потребителски права за различните обекти в системата, управление на историята на системните събития и т.н.
- ✓ Управление на курсовете и учебните програми – съдържа различни функции на инструктора и курсовия администратор: управление на курсовете и курсовите инстанции, записване на студентите в групи, създаване на учебни програми
- ✓ Инструмент за създаване на курсове – съдържа функционалността за създаване на курсове от авторите на курсове; предоставя възможност за преизползване на съдържанието на курсовете, лесна персонализация и интернационализация, лесно създаване на съдържание на курс чрез използване на шаблони

- ✓ Тестове и задания – този пакет съдържа всички функции на инструктора за създаване и разпределяне на тестове, курсови проекти и други задания на студентите, както и оценяване на тези задания и оформяне на окончателната оценка за курса; тук са включени също и функциите на студентите за представяне и изпълнение на тестове, предаване на задания и следене на личните постижения
- ✓ Комуникация – този модул съдържаща функционалността, предоставяща различни начини за комуникация между потребителите на системата: e-mail, чат, табло за дискусии, новини и виртуално пространство

Първата версия на системата е пусната реално в употреба във Факултета по Математика и Информатика на СУ "Св. Климент Охридски" за учебната 2002/2003 година. От 2003/2004 година до момента работи версия 2 на системата. ARCADE се ползва от магистърските програми на Факултета по Математика и Информатика и следдипломните квалификации. В бакалавърските програми отделни курсове също се водят през ARCADE. Изцяло през системата се водят магистърските програми Разпределени системи и мобилни технологии и Софтуерно инженерство. Сред някои от курсовете, които се преподават през ARCADE, са Управление на проекти, XML, Софтуерни технологии, Електронен бизнес, Обектно-ориентиран процес на разработване на софтуер, Обектно-ориентиран анализ и дизайн на софтуер, Middleware, Сървърно програмиране на Java, Мобилни технологии, Средства за бизнес моделиране и др. ARCADE е тествана и с курсове извън рамките на информационните технологии, като например курсове в областта на медицината. Съвсем скоро няколко курса взеха участие и спечелиха награди в Националния конкурс за създаване на електронно учебно съдържание – Проектиране на човеко-машинен интерфейс (II-ра награда), Бизнес с Интернет (II-ра награда), Визуално програмиране с Delphi (III-та награда), Изкуствен интелект и Web бази от данни (поощрителни награди).

3. Използвани технологии

Почти всички системи за дистанционно обучение използват и предоставят услугите си посредством глобалната мрежа Интернет. По този начин те осигуряват достъп до съхраняваната от тях информация и предлагат разнообразна функционалност на потребителите си. ARCADE също използва Интернет. Тя позволява отдалечено композиране, съхраняване и четене на курсове, дава възможност за запазване на файлове (например общи материали, предоставени към даден курс), притежава инструменти за тестове и тренинг на участниците, за комуникация между потребителите, предоставя възможност за управление на потребителите и техните права и т.н.

За изграждането на Web базирана система се използва трислойна архитектура. Отделните компоненти на такава архитектура са Web браузър, Web сървър и бизнес логика, База данни. Потребителят работи със системата посредством Web браузър от своя компютър, който е свързан с Интернет. На друг компютър, също свързан с Интернет и притежаващ уникален идентификатор (Интернет адрес), се намира Web сървърът. Последният приема заявките от потребителя (клиент на приложението), формира резултатите като съвкупност от HTML страници и ги връща обратно на клиента. На същата машина или на отделна може да се намира бизнес логиката на приложението. Тук се извършват всички действия, изискани от потребителя – необходимите изчисления, взаимодействието с базата от данни и др. Тук е и мястото, на което се прави проверка за правата на потребителя. При липса на съответни такива, достъпът до изисканата информация не се разрешава. В бизнес логиката се крие цялата функционалност на приложението. Третият слой е базата от данни. В нея се съхранява цялата необходима информация за работа на приложението.

3.1. Web браузър

Широко разпространени на пазара са Internet Explorer, Netscape Communicator, Opera и др. ARCADE има изискване да работи с Internet

Explorer (версия 4.0 или по-нова) и с Netscape Communicator (версия 4.x). Това са най-широко използваните програми за достъп до Интернет. Те са масово разпространени и с това изискване се покриват почти 100% от потребителите на Интернет.

3.2. Web сървър, бизнес логика

Известни са няколко различни технологии за разработка на Интернет приложения. Изборът на подход за ARCADE е обусловен от конкретните изисквания към системата. Бизнес логиката на едно Web приложение се състои от съвкупност от програмни модули, скриптове и др., които извършват съответните операции. Разпространени са няколко платформи за изграждане на бизнес логика на едно приложение. Предпочитана за последните няколко години е платформата, базирана на езика Java – обектно ориентиран език за програмиране, позволяващ изграждането на приложения, които могат да се изпълняват на различни операционни системи и хардуер. За изграждане бизнес логиката на ARCADE е избран Java. Следват някои от основните причини за избора.

Java непрекъснато се усъвършенства и разширява. От 1998 година това се прави от общност, в която участват реално всички големи фирми в индустрията с изключение на Microsoft. С течение на времето Java прераства в група от специализирани платформи: Java 2 Platform, Standard Edition (J2SE) – платформа за разработване на основни библиотеки и десктоп приложения, Java 2 Platform, Enterprise Edition (J2EE) – платформа за разработване на големи компонентно базирани и многослойни приложения, Java 2 Platform, Micro Edition (J2ME) – платформа за разработване на приложения за устройства с ограничени възможности като мобилни телефони, телевизори и т.н. В екипа, поддържащ и разширяващ J2SE/J2EE, участват фирми като Apache Software Foundation, Apple, BEA, Borland, Google, Hewlett-Packard, IBM, Intel, Oracle, SAP и разбира се Sun Microsystems. В екипа за J2ME са всички водещи фирми в мобилния сектор - Motorola, Nokia, NTT DoCoMo, Samsung, Siemens, Sony-Ericsson, Symbian,

Vodafone, IBM, Intel и Sun Microsystems. Това непрекъснато усъвършенстване и разширяване осигурява нови програмни интерфейси за всички стандарти в индустрията, а участието на големите фирми, всяка от които е одобрила тези нови интерфейси, гарантира, че те ще се придържат към тях. За приложенията, базирани на Java, това означава голяма гама от готови библиотеки, така че до голяма степен се спестява разработването и поддържането на такива, както и ползването на библиотеки, разработени от трети фирми. А също така и пълна съвместимост и поддръжка от продуктите на всички тези големи фирми, което осигурява свободен избор на хардуер, операционна система и други приложения, като например application server, от които може да се нуждае – с други думи свободна конкуренция, базирана на стандартите между големите фирми, от която крайният потребител, в случая нашето приложение, само печели.

ARCADE е лесно преносима и независима по отношение на платформата (Windows, Unix) система. Едно от силните места на Java е компилирането на програмния код до т.нар. байт код, който се стартира на Java виртуална машина и не зависи от конкретната операционна система, нито от хардуера на конкретния компютър. Това е и основната причина Java да се наложи като стандарт за Интернет програмиране.

ARCADE е отворена и лесно разширяема система, позволяваща бързо и нетрудоемко добавяне на нови възможности поради динамично изменящите се пазарни изисквания. Тези нужди идеално се вписват в обектно-ориентирания подход на програмиране, на който се базира Java. Полиморфизмът, наследяването и капсулирането в комбинация, осигуряват много по-ефективни, сигурни и разширяеми програми, в сравнение със структурния подход. Добре проектирана йерархия от класове става основа за многократно използване на код, спестяване на време за разработване и усилия за тестване.

Изборът на Java като език за програмиране предопределя и начина за генериране на набора от HTML страници от Web сървъра, а именно JSP (Java Server Pages) технологията. Комбинирането на статични HTML

страници с фрагменти от Java код се оказва много сполучливо за изграждане на цялостното приложение.

JSP технологията дава възможност на разработчиците да създават бързо и лесно да поддържат динамични Web страници с много информация. Основната идея на JSP страниците е да бъде отделен външният вид на страниците от бизнес логиката, която генерира съдържанието на страницата. Благодарение на това разделение разработчиците могат да работят над бизнес логиката, докато дизайнерите могат да изграждат външния вид на страниците без да е нужно да знаят Java или да разбират от програмиране. Това е възможно благодарение на интегрирането на JavaServer Pages Standard Tag Library (JSTL) в JSP, така че информацията може да бъде вмъквана в дизайна на страницата благодарение на набор от тагове, които приличат на HTML/XML тагове, предоставящи възможността за итерация, условия, интернационализация и т.н. Също така се предоставя възможността да се създават и собствени тагове, което представлява стандартен начин за разширяване на възможностите на JSP и JSTL, като по този начин се създават и нови библиотеки от преизползваеми тагове. Макар възможността за вмъкване на части от код да се поддържа за съвместимост, използването й не е препоръчително.

Останалите възможности накратко са:

- ✓ Perl – език за скриптове, замислен като език от високо ниво с наченки на изкуствен интелект и лесна работа с файлове. Притежава слабо развит обектен модел, работи предимно на Unix базирани платформи. Неприложим за големи приложения.
- ✓ PHP – технология, намерила добро развитие за последните години. Не е много подходяща за големи приложения, липсва богат език за програмиране от ранга на Java. Разработвана е предимно за изграждане на малки Интернет сайтове, откъдето идва и името (Personal Home Page).
- ✓ ASP – разработка на Microsoft, много добра, лесно интегрираща се с всички останали продукти на корпорацията (Access, Internet

Information Server, SQL Server). За съжаление работи само с продукти на Microsoft и само на Windows операционна система.

За JSP сървър е избран софтуерът на Apache, Jakarta-Tomcat. Този продукт е малък по обем, лесен за инсталиране и администриране. Apache Tomcat е Web сървър, който поддържа JSP спецификацията и включва много други функции, което го прави много добра среда за разработка и изпълнение на web приложения и web услуги.

3.3. База данни

Съхранението на данните е следващият ключов момент в изграждането на кое да е приложение. На пазара съществуват няколко основни разработки на сървъри за бази от данни. В момента се предлагат почти единствено сървъри на релационни бази от данни (RDBMS – Relational Database Management Systems). Релационният модел се наложи като стандарт и на практика останалите модели (мрежови, йерархичен) вече не се използват.

За база данни в ARCADE първоначално е избрана Oracle 8i. Това е един мощен сървър за бази от данни с много допълнителни възможности, част от които се използват с пълна сила. Необходимостта от преносимост на системата обуславя съхраняването на абсолютно всяка информация директно в базата от данни (вкл. файлове). Така се постига пълна независимост от операционната система – файловете се съхраняват в базата от данни, а не в операционната система. Други причини за избора на Oracle са нуждата от голяма банка за съхраняване на информация и управлението ѝ. Обемът на данните изисква индексиране, рестрикции и др. Oracle дава възможност за автоматично наблюдение на цялостността и коректността на данните (Data Integrity). Фирмата се е погрижила и за широк набор от драйверни програми за интегриране с Java (JDBC). Налична е и много документация, която улеснява процеса на разработка на продукта.

Към момента ARCADE използва версия 8.1.7. В последствие е създадена възможност ARCADE да използва и MySQL за база от данни. Текущата разработка също използва MySQL. Това е един интересен “мини” сървър,

спечелил доверието на половината от Интернет приложенията. Разпространява се безплатно и е много лесен за инсталиране и администриране. MySQL е релационна система за управление на база данни с отворен код. Има голямо количество документация, която е достъпна безплатно, в Интернет също може да се намерят много материали относно MySQL. Фирмата, която разработва MySQL, предлага и поддръжка срещу заплащане. MySQL си е спечелил името на най-бързата СУБД с отворен код, но бързината е за сметка на функционалността, която предлага. MySQL не предлага тригери, вложени заявки и views. Въпреки това във версия 5.0, която още не е обявена за стабилна, ще бъде добавена поддръжка на част от тази функционалност.

4. Проектиране

При създаване на софтуерно приложение разработчиците започват със събиране на потребителските изисквания. Потребителските изисквания се извличат с помощта на така наречените случаи на употреба в модела на случаите на употреба. Следващите стъпки са анализ и дизайн на системата с цел реализиране случаите на употреба. Най-напред се създава модел на анализа, след това дизайн модел и модел на внедряването. Следва имплементация на системата в имплементационен модел, който включва кода, компонентите на системата. Накрая се създава тест модел, с помощта на който се проверява, че системата предоставя възможността, описана в случаите на употреба. Всички модели са свързани помежду си.

4.1. Потребителски изисквания

Събирането на изискванията има две основни цели. Едната е да се намерят случаите на употреба, които са стойностни за потребителите. Другата цел е тези случаи на употреба да бъдат представени по начин, разбираем от бъдещите потребители и клиента. За тази цел при описанието на потребителските изисквания се използва езикът на клиента и потребителите.

Системата има много видове потребители, като всеки вид потребител се представя като актьор. Актьорите използват системата, като взаимодействат със случаите на употреба. Случаят на употреба представлява последователност от действия, които системата извършва, за да предложи някакъв резултат на актьора. Моделът на случаите на употреба се изгражда от всички актьори и случаи на употреба на системата. Този модел се използва, за да се постигне съгласие с потребителите и клиента за това какво системата трябва да прави. На него може да се гледа като на спецификация на всички възможни начини на използване на системата.

По време на процеса на събиране на изискванията се идентифицират потребителските и нуждите на клиента като изисквания. Функционалните

изисквания се представят чрез случаи на употреба в модел на случаи на употреба. Останалите изисквания се представят или към някой конкретен случай на употреба, който засягат, или могат да се записват в отделен списък.

Актьорите на една система не е задължително да бъдат единствено хора. Актьори могат да бъдат други системи, които взаимодействат със системата. Всеки актьор има определени роли, когато взаимодейства със системата. Физически един потребител може да изпълнява роли на няколко актьора. Както и няколко физически лица могат да изпълняват ролите на един актьор. Например в системата ARCADE има много потребители, които изпълняват ролите на актьора студент, както и много потребители, които изпълняват ролите на актьора инструктор. Реален е също случаят, при който един инструктор може да бъде и студент. Можем да намерим и специфицираме всички актьори, като определим какви потребители ще използват системата и кои други системи ще взаимодействат с нея. Всяка категория потребители или взаимодействащи системи се представят като актьори на разглежданата система.

При определяне на потребителските изисквания се включват следните стъпки:

- ✓ Списък на кандидат изискванията
- ✓ Разбиране контекста на системата
- ✓ Събиране на функционалните изисквания
- ✓ Събиране на нефункционалните изисквания

Списъкът на кандидат изискванията съдържа всички идеи, възникнали през живота на системата от клиенти, потребители, аналитици и разработчици, които в даден момент могат да се превърнат в реални изисквания. Този списък нараства с добавянето на нови идеи, а намалява, когато идеите бъдат избрани за изисквания на системата и преобразувани в други артифакти, като например случаи на употреба. Списъкът с идеи се използва единствено за планиране на работата.

За извличането на верните изисквания е необходимо разработчиците да познават добре контекста, в който работи (ще работи) системата. Това се постига по два основни начина: чрез създаване на домейн модел или чрез създаване на бизнес модел. Домейн моделът описва важните понятия на контекста като домейн обекти и ги свързва един с друг. Домейн обектите представляват “нещата”, които съществуват или събитията, които се случват в средата, в която работи системата. Домейн моделът се описва с UML диаграми, най-вече диаграми на класовете. Тези диаграми илюстрират на потребителите, клиентите, критиците и другите разработчици домейн класовете и как те са свързани един с друг чрез асоциации. Бизнес моделът представлява надмножество на домейн модела. Целта на бизнес моделирането е описанието на процесите. Бизнес моделът специфицира кои бизнес процеси ще бъдат поддържани от системата. Той описва също действащите лица, техните отговорности и действията, които те могат да извършват със системата.

Начинът, по който се идентифицират изискванията, се базира на случаите на употреба. Всеки случай на употреба представя начин за използване на системата. За всеки потребител има няколко различни случаи на употреба, всеки от които представя начина, по който този потребител използва системата.

Нефункционалните изисквания задават системните свойства като изпълнение, зависимост от платформата, поддръжка, разширяемост, сигурност.

4.1.1. Модел на случаите на употреба

Най-важният документ (артифакт) от процеса на събиране на изискванията представлява моделът на случаите на употреба. Моделът на случаите на употреба съдържа актьорите, случаите на употреба, както и връзките между тях. Моделът на случаите на употреба се представя с помощта на UML диаграми, които показват актьорите и случаите на употреба от различни гледни точки и с различни цели. Ако моделът на случаите на употреба е голям, е добре да се въведат пакети в модела.

4.1.2. Актьори

Моделът на случаите на употреба описва какво прави системата за всеки тип потребител. Всеки тип потребител се представя като един или повече актьора. Всяка външна система, с която взаимодейства системата, също се представя като един или няколко актьора. Така актьорите представят страните извън системата, които взаимодействат със системата. За да бъдат открити всички актьори на системата, е необходимо да се идентифицират всички потенциални потребители, след което тези потребители трябва да бъдат категоризирани. С идентифицирането на всички актьори на системата е идентифицирана външната среда на системата. За всеки идентифициран актьор се прави кратко описание.

Модулът за управление на курсовите задачи в системата ARCADE взаимодейства с тип потребител, който използва системата, за да създава и разпределя задания, да създава и управлява календарни събития, както и да одобрява или отхвърля предложения за задания. Този тип потребител се представя от актьора Инструктор. Той взаимодейства със системата ARCADE, като ръководи един или няколко курса.

Другият тип потребител, който взаимодейства с модула за управление на курсовите задачи в системата ARCADE, е актьорът Студент. Този потребител взаимодейства с модула, като прави избор на задание, както и създава предложения за задания. Този актьор взаимодейства със системата ARCADE, като участва в един или няколко курса и достъпва материалите по съответните курсове, които посещава.

В системата ARCADE има още два актьора, системен администратор и курсов администратор. Системният администратор поддържа системата ARCADE, управлява достъпа на различните потребители. Курсовият администратор управлява и поддържа курсовете в системата, създава, редактира и премахва курсове. Той може също да регистрира, редактира и премахва студенти и инструктори от даден курс.

Фигура 4.1.2.1 изобразява отделните актьори в системата.

Фигура 4.1.2.1. Актьори в системата ARCADE

Един актьор има роля за всеки случай на употреба, с който той взаимодейства. Всеки път, когато определен потребител (човек или друга система) взаимодейства със системата, съответната инстанция на актьора извършва съответната роля. Инстанцията на актьор представлява определен потребител, взаимодействащ със системата.

4.1.3. Случаи на употреба

Всеки начин, по който един актьор използва системата, се представя като случай на употреба. Случаят на употреба специфицира поредица от действия, включително и алтернативи на поредицата, които системата може да извършва, взаимодействайки с актьорите. Случаят на употреба съдържа операции и атрибути. Описанието на случая на употреба може да включва диаграми на състоянията (statechart), диаграми на активностите (activity), диаграми на кооперирането (collaborations) и диаграми на последователностите (sequence). Инстанция на случай на употреба представлява изпълнението на случая на употреба. Инстанциите на случаи на употреба не взаимодействат с други инстанции на случаи на употреба. Единствените взаимодействия в модела на случаите на употреба са между инстанции на актьори и инстанции на случаи на употреба. За всеки случай на употреба в текстов формат се описва потокът на събитията за този случай. Потокът на събитията специфицира какви действия извършва системата, когато се изпълнява съответният случай на употреба. Потокът на събития специфира също как системата взаимодейства с актьорите, когато се изпълнява случаят на употреба. В текстов формат се описват и всички специални, нефункционални изисквания, които се отнасят за случая на употреба. Тези изисквания са необходими, за да бъдат обработени в по-

следващите етапи от разработването на системата като анализа, дизайна или имплементацията. Както вече споменах всеки случай на употреба се описва детайлно с поток на събитията. Потокът на събитията съдържа информация как започва случаят на употреба, как завършва и как взаимодейства с актьорите. Случаят на употреба дефинира състоянията, в които могат да влязат инстанциите на случая на употреба, както и всички възможни пътища между тези състояния.

4.1.3.1. Случай на употреба **Записване за задание**:

Предусловие: Студентът е влязъл в системата ARCADE в модула, свързан със заданията за самостоятелна работа.

Поток на събитията

Основен път

1. Студентът започва случая на употреба, като влиза в подмодула за избор на задание.
2. Студентът избира курс, за който се отнася заданието.
3. Студентът избира календарно събитие, за което се отнася заданието.
4. Студентът избира съответно задание, за което желае да се запише.
5. Студентът избира да се запише за избраното задание.
6. Ако са изпълнени условията за записване за задание, системата записва избраното задание за този студент.
7. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент студентът може да се откаже от записването за задание и да се върне назад.
- ✓ За стъпка 2, ако студентът не е записан за нито един курс, той не може да се запише за задание и трябва да се върне назад.
- ✓ За стъпка 2, ако за избрания курс няма зададени календарни събития, студентът не може да се запише за задание и трябва да се върне назад.
- ✓ За стъпка 6, ако не са изпълнени всички условия, необходими при записване за задание, системата няма да запише студента за

избраното задание. Системата издава съобщение за грешка. Условията за записване за задание са:

- Текущата дата трябва да е преди крайната дата за записване;
- Броят на записалите се до момента студенти за съответното задание трябва да е по-малък от определения брой студенти за него;
- Студентът не трябва да е записан за друго задание от същото календарно събитие.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата запише студента за заданието, когато студентът избере да се върне назад и не се запише за задание или когато някое от условията за записване за задание не е изпълнено и системата не записва студента за това задание.

4.1.3.2. Случай на употреба **Отписване от задание**:

Предусловие: Студентът е влязъл в системата ARCADE в модула, свързан със заданията за самостоятелна работа.

Поток на събитията

Основен път

1. Студентът започва случая на употреба, като влиза в подмодула на разпределените му задания.
2. Студентът избира курса, за който се отнася заданието, от което ще се отписва.
3. Студентът избира съответното задание, от което желае да се отпише.
4. Студентът избира да се отпише от избраното задание.
5. Ако са изпълнени условията за отписване от задание, системата отписва студента от това задание.
6. Инстанцията на случая на употреба завършва.

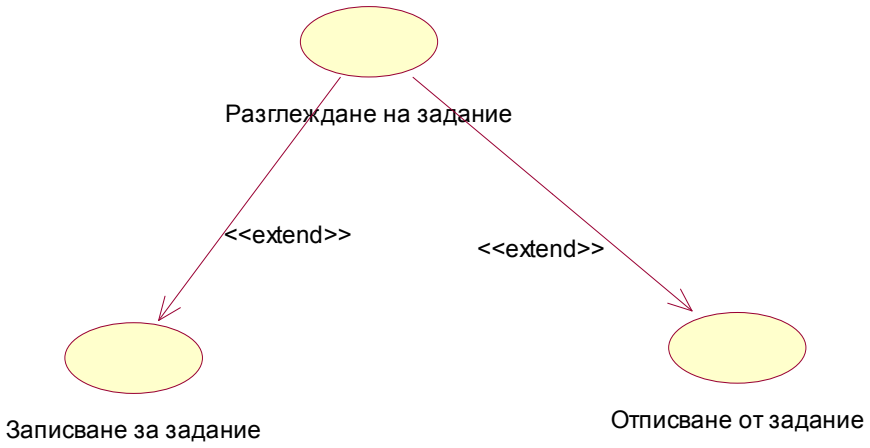
Алтернативни пътища

- ✓ Във всеки един момент студентът може да се откаже от отписването от задание и да се върне назад.

- ✓ За стъпка 2, ако студентът не е записан за нито един курс, той не може да се отпише от задание и трябва да се върне назад.
- ✓ За стъпка 3, ако студентът няма разпределено задание за избрания курс, той не може да се отпише от задание и трябва да се върне назад.
- ✓ За стъпка 5, ако не са изпълнени всички условия, необходими при отписване от задание, системата няма да отпише студента от избраното задание. Системата издава съобщение за грешка. Условията за отписване от задание са:
  - ➢ Заданието трябва да бъде от тип за избор от студента. Задания, разпределени от инструктора (с тип на разпределение отгоре-надолу) не могат да бъдат отписвани от студентите);
  - ➢ Текущата дата трябва да е преди крайната дата за записване за това задание.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата отпише студента от заданието, когато студентът избере да се върне назад и не се отпише от заданието или когато някое от условията за отписване от задание не е изпълнено и системата не отписва студента от това задание.

Случаите на употреба Записване/Отписване от задание се разширяват от досега съществуващия случай на употреба Разглеждане на задание. Тези случаи на употреба са изобразени на фигура 4.1.3.1.



Фигура 4.1.3.1. Случаи на употреба Разглеждане на задание, Записване и Отписване от задание

4.1.3.3. Случай на употреба **Управление на календарни събития**:

Предусловие: Инструкторът е влязъл в системата ARCADE в модула, свързан със заданията.

Поток на събитията

Основен път

1. Случаят на употреба започва с влизането на инструктора в подмодула за управление на календарни събития.
2. Инструкторът избира курс, за който се отнасят календарните събития.
3. Инструкторът прави избор от възможни действия, свързани с календарните събития (създаване, редактиране или изтриване).
4. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент инструкторът може да се върне назад.
- ✓ За стъпка 2, ако инструкторът не е регистриран като инструктор на нито един курс, той не може да работи с календарни събития и трябва да се върне назад.

Крайно условие: Инстанцията на случая на употреба завършва, когато инструкторът избере да създаде, редактира или изтрие календарно събитие (стартира случаи на употреба Създаване/Редактиране/Изтриване на календарно събитие) или когато избере да се върне назад.

4.1.3.4. Случай на употреба **Създаване на календарно събитие**:

Предусловие: Инструкторът е избрал курса, за който ще създава календарно събитие.

Поток на събитията

Основен път

1. Случаят на употреба започва с избора на инструктора да създаде ново календарно събитие за избран от него курс.
2. Инструкторът описва характеристиките на новото календарно събитие като име, тип, дати и т.н.
3. Инструкторът избира да запише така описаното календарно събитие.
4. В случай че всички условия за създаване на календарно събитие са изпълнени, системата записва новото календарно събитие.
5. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент инструкторът може да избере да не създава календарно събитие и да се върне назад.
- ✓ За стъпка 4, ако не са изпълнени условията, необходими за запис на календарно събитие, системата не създава календарно събитие. Системата издава съобщение за грешка. Условията за запис на календарно събитие са:
  - Името на календарното събитие трябва да бъде въведено.
  - Началната и крайната дата за календарното събитие трябва да бъдат въведени.
  - Крайната дата не може да е преди началната дата за събитието.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата създаде календарно събитие, когато инструкторът е избрал да се върне назад без да създава календарно събитие или когато, поради неизпълнени условия, системата не създава календарно събитие.

4.1.3.5. Случай на употреба **Редактиране на календарно събитие**:

Предусловие: Инструкторът е избрал курса, за който ще редактира календарно събитие, както и самото календарно събитие, което желае да редактира.

Поток на събитията

Основен път

1. Случаят на употреба започва с избора на инструктора да редактира избраното от него календарно събитие.
2. Инструкторът редактира характеристиките на календарното събитие като име, тип, дати и т.н.
3. Инструкторът избира да запише така редактираното календарно събитие.
4. В случай че всички условия за запис на календарно събитие са изпълнени, системата записва редактираното календарно събитие.
5. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент инструкторът може да избере да не редактира календарното събитие и да се върне назад.
- ✓ За стъпка 4, ако не са изпълнени условията, необходими за запис на календарно събитие, системата не записва редактираното календарно събитие. Системата издава съобщение за грешка. Условията за запис на календарно събитие са:
  - Името на календарното събитие трябва да бъде въведено.
  - Началната и крайната дата за календарното събитие трябва да бъдат въведени.
  - Крайната дата не може да е преди началната дата за събитието.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата запише редактираното календарно събитие, когато инструкторът е избрал да се върне назад без да редактира календарно събитие или когато, поради неизпълнени условия, системата не записва редактираното календарно събитие.

4.1.3.6. Случай на употреба **Изтриване на календарно събитие**:

Предусловие: Инструкторът е избрал курса, за който ще изтрива календарно събитие, както и самото календарно събитие, което желае да изтрие.

Поток на събитията

Основен път

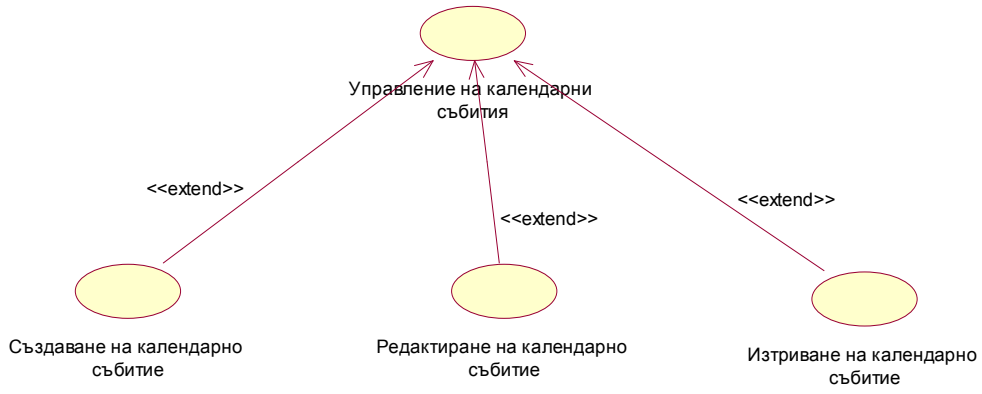
1. Случаят на употреба започва с избора на инструктора да изтрие избраното от него календарно събитие.
2. Системата изтрива календарното събитие.
3. Инстанцията на случая на употреба завършва.

Алтернативен път

- ✓ Във всеки един момент инструкторът може да избере да не изтрива календарното събитие и да се върне назад.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата изтрие календарното събитие или когато инструкторът е избрал да се върне назад без да изтрие календарното събитие.

Случаите на употреба Създаване/Редактиране/Изтриване на календарно събитие разширяват случая на употреба Управление на календарни събития. Тези случаи на употреба са изобразени на фигура 4.1.3.2.



Фигура 4.1.3.2. Случаи на употреба Управление на календарни събития, Създаване, Редактиране и Изтриване на календарно събитие

4.1.3.7. Случай на употреба **Определяне на събитие за задание**:

Предусловие: Инструкторът е избрал заданието, за което желае да определи календарно събитие.

Поток на събитията

Основен път

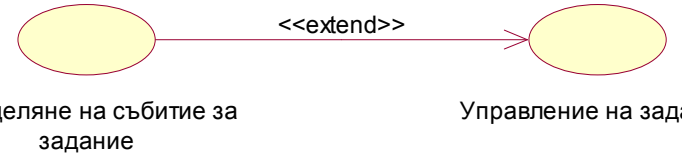
1. Случаят на употреба започва с избора на инструктора да зададе календарно събитие за избраното от него задание.
2. Инструкторът избира конкретното календарно събитие, за което желае да прикрепи избраното задание.
3. Инструкторът избира да зададе избраното календарно събитие за заданието.
4. Системата свързва заданието с избраното календарно събитие.
5. Инстанцията на случая на употреба завършва.

Алтернативен път

- ✓ Във всеки един момент инструкторът може да избере да не определя календарно събитие за заданието и да се върне назад.

Крайно условие: Случаят на употреба завършва, когато системата свърже избраното задание с календарното събитие или когато инструкторът избере да не задава календарно събитие за заданието и се върне назад.

Случаят на употреба Определяне на събитие за задание разширява досега съществуващия случай на употреба Управление на задания. Двата случая на употреба са изобразени на фигура 4.1.3.3.



Фигура 4.1.3.3. Случаи на употреба Определяне на събитие за задание и Управление на задания

4.1.3.8. Случай на употреба **Управление на предложения за задания**:

Предусловие: Инструкторът/Студентът е влязъл в системата ARCADE в модула, свързан с предложения за заданията.

Поток на събитията

Основен път

1. Случаят на употреба започва с влизането на инструктора или студента в подмодула за управление на предложения за задания.
2. Инструкторът или студентът избира курс, за който се отнасят предложенията за задания.
3. Инструкторът или студентът прави избор от възможни действия, свързани с предложенията за задания. Възможните действия за студента са създаване, редактиране и изтриване на предложения. Възможното действие за инструктора е редактиране на предложение.
4. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент инструкторът или студентът може да се върне назад.
- ✓ За стъпка 2, ако инструкторът не е регистриран като инструктор на нито един курс, той не може да работи с предложения за задания и трябва да се върне назад. Съответно ако студентът не е регистриран за нито един курс, той не може да работи с предложения за задания и трябва да се върне назад.

Крайно условие: Инстанцията на случая на употреба завършва, когато инструкторът или студентът избере да извърши някое от допустимите за него действия (стартира случаи на употреба Създаване/Редактиране/Изтриване на предложение за задание) или когато избере да се върне назад.

Допълнителни изисквания: Всеки инструктор може да извършва действия с предложения за задания, създадени само за курсовете, за които той е инструктор. Всеки студент може да извършва действия с предложения за задания, създадени само лично от него. Студентът не трябва да има достъп до предложения, създадени от други студенти.

4.1.3.9. Случай на употреба **Създаване на предложение за задание**:

Предусловие: Студентът е избрал курса, за който ще създава предложение за задание.

Поток на събитията

Основен път

1. Случаят на употреба започва с избора на студента да създаде ново предложение за задание за избран от него курс.
2. Студентът описва характеристиките на новото предложение за задание като име, тип, описание и т.н.
3. Студентът избира да запише така описаното предложение за задание.
4. В случай че всички условия за създаване на предложение за задание са изпълнени, системата записва новото предложение за задание.
5. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент студентът може да избере да не създава предложение за задание и да се върне назад.
- ✓ За стъпка 4, ако не са изпълнени условията, необходими за запис на предложение за задание, системата не създава предложението. Системата издава съобщение за грешка. Условията за запис на предложение за задание са*:*
  - Името на предложението за задание трябва да бъде въведено.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата създаде предложение за задание, когато студентът избере да се върне назад без да създава предложение за задание или когато, поради неизпълнени условия, системата не създава предложение за задание.

4.1.3.10. Случай на употреба **Редактиране на предложение за задание**:

Предусловие: Инструкторът или студентът е избрал курса, за който ще редактира предложение за задание, както и самото предложение, което желае да редактира.

Поток на събитията

Основен път

1. Случаят на употреба започва с избора на инструктора или студента да редактира избраното от него предложение за задание.

2. Инструкторът или студентът редактира характеристиките на предложението за задание като име, тип, описание и т.н.
3. Инструкторът или студентът избира да запише така редактираното предложение за задание.
4. В случай че всички условия за запис на предложение за задание са изпълнени, системата записва редактираното предложение за задание.
5. Инстанцията на случая на употреба завършва.

Алтернативни пътища

- ✓ Във всеки един момент инструкторът или студентът може да избере да не редактира предложението за задание и да се върне назад.
- ✓ За стъпка 4, ако не са изпълнени условията, необходими за запис на предложение за задание, системата не записва редактираното предложение. Системата издава съобщение за грешка. Условията за запис на предложение за задание са*:*
  - ➢ Името на предложението за задание трябва да бъде въведено.
- ✓ На стъпка 3 инструкторът може да избере да одобри предложението за задание. В този случай, ако всички условия за запис на задание са изпълнени, системата записва ново задание и изтрива предложението за задание. Инстанцията на случая на употреба завършва.
- ✓ На стъпка 3 инструкторът може да избере да откаже предложението за задание. В този случай системата изтрива предложението за задание. Инстанцията на случая на употреба завършва.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата запише редактираното предложение за задание, когато инструкторът или студентът избере да се върне назад без да редактира предложението за задание, когато, поради неизпълнени условия, системата не записва редактираното предложение за задание, когато системата запише ново задание и изтрие предложението или когато системата изтрие предложението за задание.

Допълнителни изисквания: Модулът трябва да предоставя възможност за синхронизация при редактиране на предложение от инструктора и студента.

4.1.3.11. Случай на употреба **Изтриване на предложение за задание**:

Предусловие: Студентът е избрал курса, за който ще изтрива предложение за задание, както и самото предложение за задание, което желае да изтрие.

Поток на събитията

Основен път

1. Случаят на употреба започва с избора на студента да изтрие избраното от него предложение за задание.
2. Системата изтрива предложението за задание.
3. Инстанцията на случая на употреба завършва.

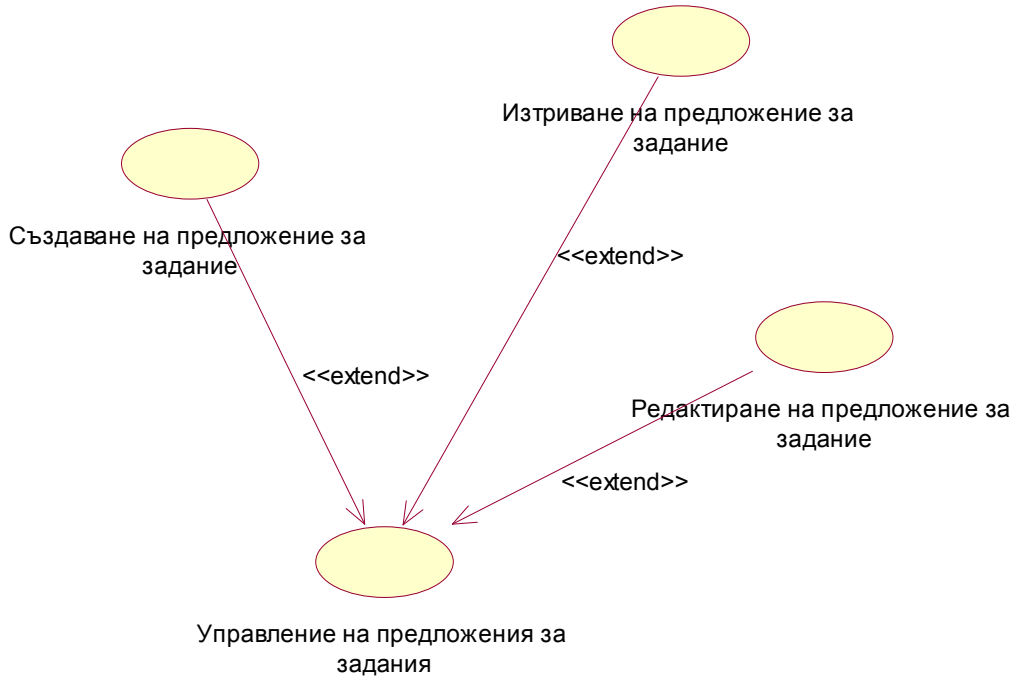
Алтернативен път

- ✓ Във всеки един момент студентът може да избере да не изтрива предложението за задание и да се върне назад.

Крайно условие: Инстанцията на случая на употреба завършва, когато системата изтрие предложението за задание или когато студентът избере да се върне назад без да изтрие предложението за задание.

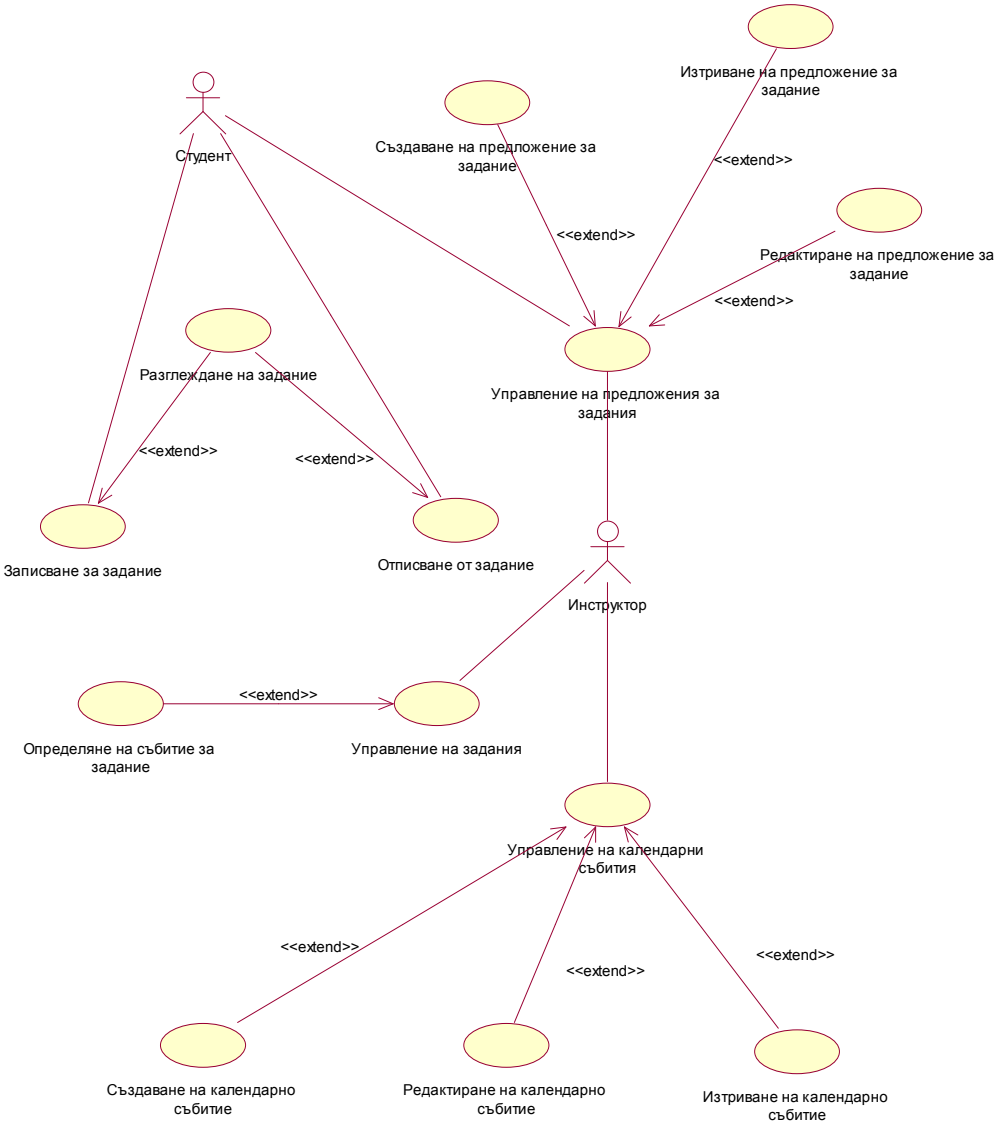
Допълнителни изисквания: Модулът трябва да предоставя възможност за синхронизация при редактиране на предложение от инструктора и изтриването му от студента.

Случаите на употреба Създаване/Редактира/Изтриване на предложение за задание разширяват случая на употреба Управление на предложения за задания. Тези случаи на употреба са изобразени на фигура 4.1.3.4.



Фигура 4.1.3.4. Случаи на употреба Управление на предложения за задания, Създаване, Редактиране и Изтриване на предложение за задание

На фигура 4.1.3.5 е изобразен цялостния модел на случаите на употреба:



Фигура 4.1.3.5. Диаграма на случаите на употреба

4.2. Анализ

При анализа се анализират дотук описаните потребителски изисквания чрез детайлизиране и структуриране.

Случаите на употреба трябва да бъдат независими един от друг колкото е възможно. Това се постига чрез неспускане в подробности относно конкурентност и конфликти между случаите на употреба, когато например

се състезават за едни и същи ресурси, вътрешни за системата. Например случаите на употреба Редактиране и Изтриване на предложение за задание могат да се отнасят за едно и също предложение. В случая когато студент изтрие предложение за задание в момент, когато инструкторът го редактира, ще настъпи конфликт. Нещата, засягащи конкурентност и конфликти могат да останат неразгледани при събирането на изискванията. Друга подробност е, че случаите на употреба трябва да бъдат описани на езика на клиента. Това се постига, най-вече като се използва естествен език за описанието на случаите на употреба.

Основната цел на анализа е да разгледа неразгледаните досега аспекти при събирането на изискванията, да ги анализира по-задълбочено и най-вече на езика на разработчика. Следва кратко сравнение на модела на случаите на употреба с модела на анализа:

1. Моделът на случаите на употреба е описан на езика на клиента. Моделът на анализа е описан на езика на разработчика.
2. Моделът на случаите на употреба предоставя външен поглед на системата. Моделът на анализа предоставя вътрешен поглед на системата.
3. Моделът на случаите на употреба е структуриран със случаи на употреба. Моделът на анализа е структуриран със стереотипни класове и пакети.
4. Моделът на случаите на употреба се използва главно като договор между клиента и разработчиците за това какво системата трябва и не трябва да прави. Моделът на анализа се използва главно от разработчиците за разбиране на това как трябва да бъде оформена системата, т.е. проектирана и имплементирана.
5. Моделът на случаите на употреба може да съдържа повторения, противоречия измежду потребителските изисквания. Моделът на анализа не трябва да съдържа подобни неща.

6. Моделът на случаите на употреба дефинира случаите на употреба, които по-късно се анализират в модела на анализа. Моделът на анализа дефинира реализация на всеки от случаите на употреба.

4.2.1. Модел на анализа

Основен артифакт от процеса на анализа това е моделът на анализа. Моделът съдържа класове на анализа, както и реализация на случаите на употреба от модела на случаите на употреба. Класовете и реализацията могат да бъдат организирани в пакети на анализа. Класовете на анализа представляват абстракции на класовете и възможните подсистеми в дизайна на системата. Случаите на употреба са реализирани от класовете на анализа и техните обекти, като е представено кооперирането между тях.

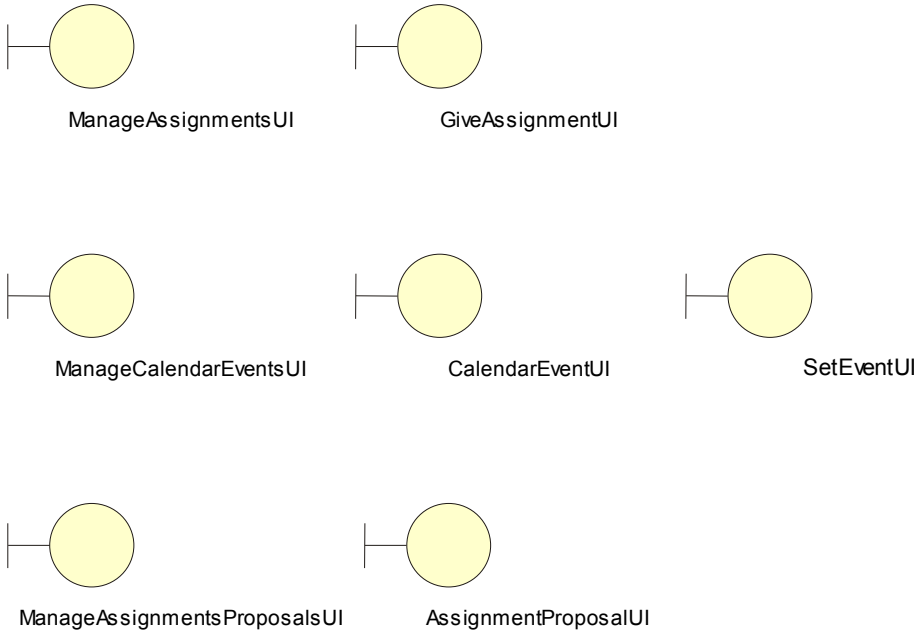
4.2.2. Клас на анализа

Класът на анализа представлява абстракция на един или повече класа и/или подсистеми в дизайна на системата. Класът на анализа се фокусира върху обработката на функционалните изисквания. Нефункционалните, специалните изисквания се отлагат за по-късен етап по време на дизайна и имплементацията. Тук също се използват текстови описания за дефиниране на поведението на класа. Класът на анализа може да дефинира атрибути, но на неформално ниво. Класовете на анализа могат да бъдат 3 вида: boundary, control или entity класове.

4.2.2.1. **Boundary** класове

Boundary класът се използва за моделиране на взаимодействие между системата и нейните актьори (потребители и външни системи). Взаимодействието често включва получаване и предоставяне на информация, както и заявки от и към потребителите и външните системи. Boundary класовете моделират тази част от системата, която зависи от нейните актьори. Те изясняват изискванията, свързани с границите на системата. Boundary класът не е задължен да описва физически как става взаимодействието, това се разглежда по-късно при дизайна и имплементацията. Всеки boundary клас трябва да е свързан поне с един актьор и обратно.

На фигура 4.2.3.1 са изобразени boundary класовете в разработвания подмодул на заданията за самостоятелна работа:

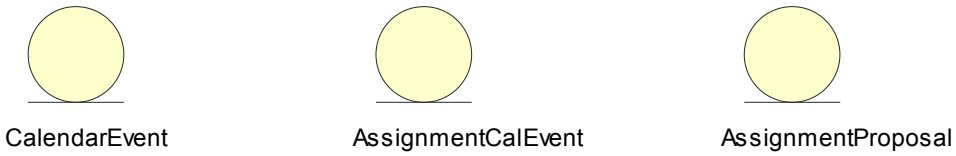


Фигура 4.2.3.1. Boundary класове

4.2.2.2. **Entity** класове

Този вид клас се използва за моделиране на дълготрайна или постоянна информация за някакво явление или концепция, като например индивид, обект от реалния живот или събитие от реалния живот. Entity класът може да има сложно поведение, свързано с информацията, която представя. Примери за entity класове са задание, събитие и т.н.

На фигура 4.2.3.2 са изобразени entity класовете в разработвания подмодул на заданията за самостоятелна работа:



Фигура 4.2.3.2. Entity класове

4.2.2.3. **Control** класове

Control класовете представят съгласуването, последователността, транзакциите и контрола на другите обекти. Често се използват за капсулиране на контрол, свързан с определен случай на употреба. Използват се и за представяне на бизнес логиката на системата, изчисления и решения, които не могат да бъдат свързани с никоя дълготрайна информация, съхранявана от системата. Динамиката на системата се моделира от контролни класове, те обработват основните действия и потоци и разпределят работа на другите обекти (boundary и entity).

4.2.3. Реализация на случаите на употреба

Реализацията на случаите на употреба е част от анализа, която описва как определен случай на употреба се реализира и представя в термините на класовете на анализа и техните обекти. Реализацията на случай на употреба предоставя връзка до определен случай на употреба в модела на случаите на употреба.

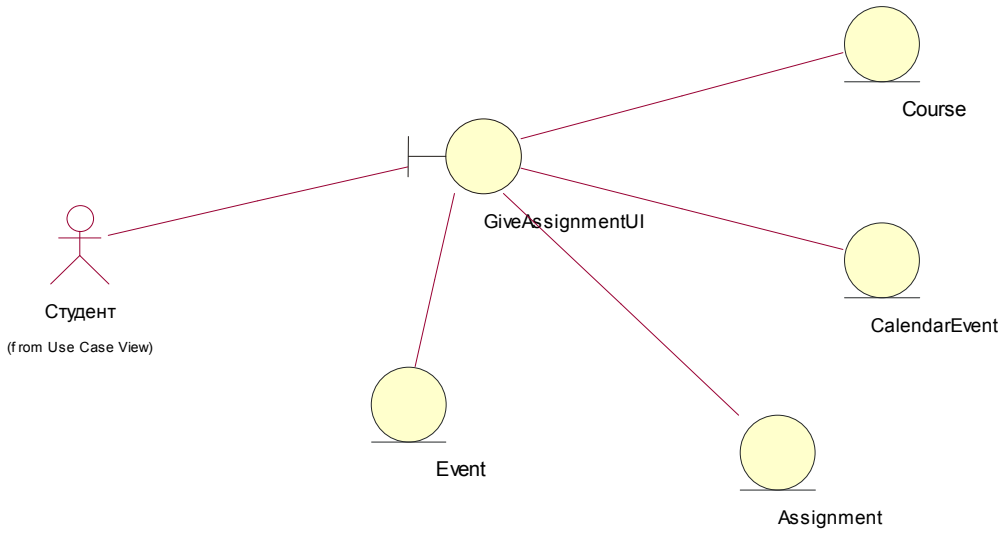
Реализацията на случай на употреба се състои от текстово описание на потока на събитията, диаграми на класовете, които изобразяват участващите класове на анализа и техните взаимоотношения, и диаграми на взаимодействията, които изобразяват реализацията на определен поток или сценарий на случая на употреба в термините на взаимодействията на обектите на анализа. Тук отново фокусът е върху функционалните изисквания. Както при класовете на анализа, обработката на нефункционалните изисквания се отлага за етапите на дизайна и имплементацията.

Последователността от действия при даден случай на употреба започва, когато даден актьор извика случая на употреба чрез изпращане на съобщение до системата в някаква форма. Обект от тип boundary ще получи това съобщение от актьора, след което boundary обектът ще изпрати съобщение до някой друг обект и така замесените обекти взаимодействат, за да реализират случая на употреба. При анализа това се изобразява с диаграми на кооперирането. Тук по-важно е откриването на

изискванията и задълженията на обектите, а не толкова откриването на подробна и хронологическа последователност на взаимодействията, в който случай се използват диаграми на последователностите.

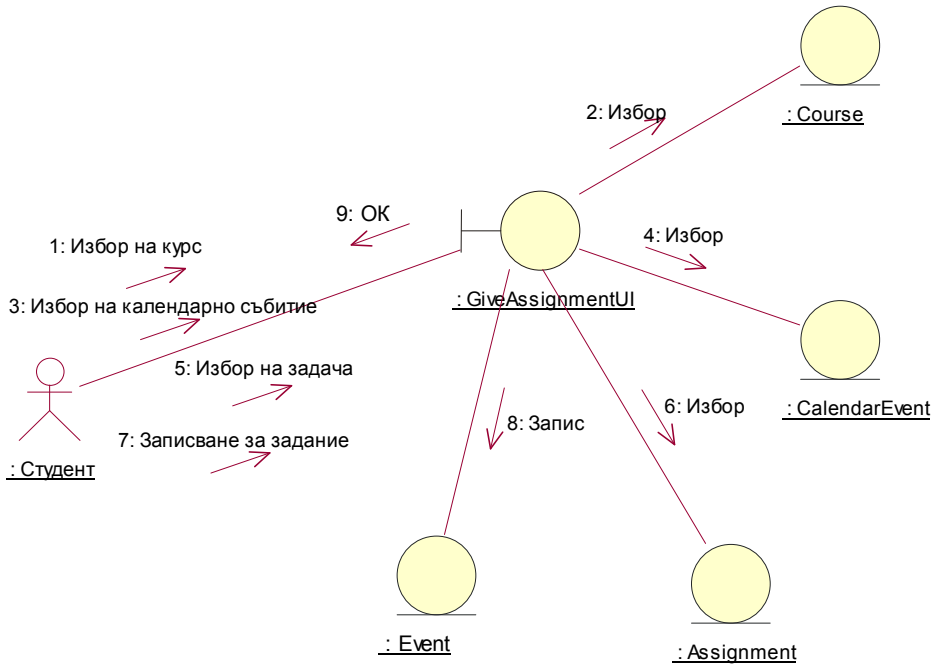
4.2.3.1. Реализация на случай на употреба **Записване за задание**:

Фигура 4.2.3.1.1 изобразява диаграмата на класовете:



Фигура 4.2.3.1.1. Диаграма на класовете за случай на употреба Записване за задание

Фигура 4.2.3.1.2 изобразява диаграмата на кооперирането:

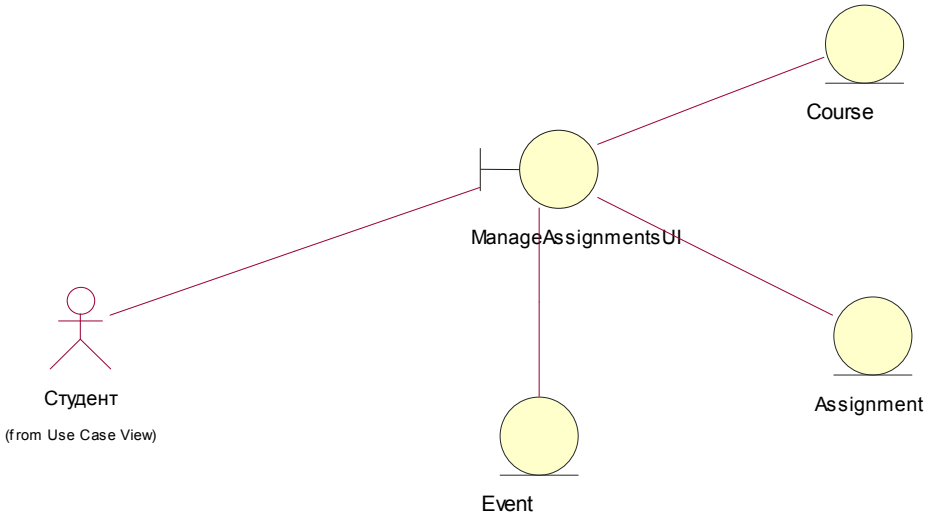


Фигура 4.2.3.1.2. Диаграма на кооперирането за случай на употреба Записване за задание

Поток на събитията: Студентът най-напред избира курса, за който ще се записва за задание, чрез boundary класа GiveAssignmentUI (1, 2). След това студентът избира календарното събитие (3, 4). Следва избор на задачата (5, 6) и накрая записване за заданието (7). Класът GiveAssignmentUI записва студента за избраното от него задание (8).

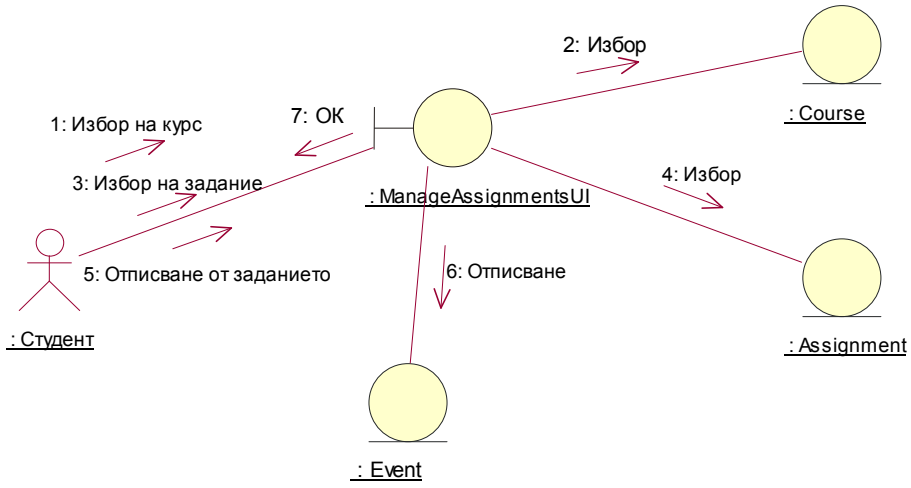
4.2.3.2. Реализация на случай на употреба **Отписване от задание**:

Фигура 4.2.3.2.1 изобразява диаграмата на класовете:



Фигура 4.2.3.2.1. Диаграма на класовете за случай на употреба Отписване от задание

Фигура 4.2.3.2.2 изобразява диаграмата на кооперирането:

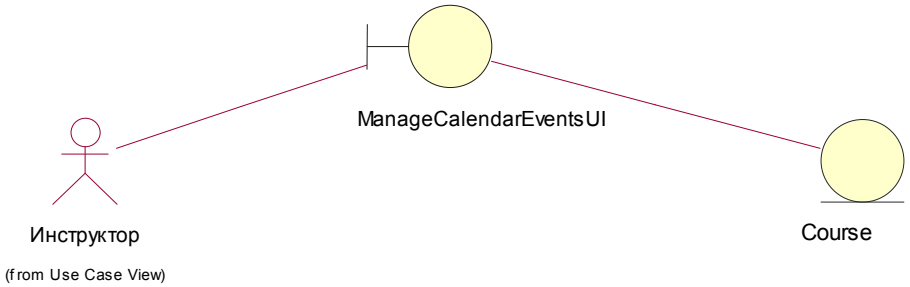


Фигура 4.2.3.2.2. Диаграма на кооперирането за случай на употреба Отписване от задание

Поток на събитията: Студентът избира курса чрез класа ManageAssignmentsUI (1, 2). Следваща стъпка е избор на заданието, от което желае да се отпише (3, 4). Студентът се отписва от заданието (5). ManageAssignmentsUI отписва студента от избраното задание (6).

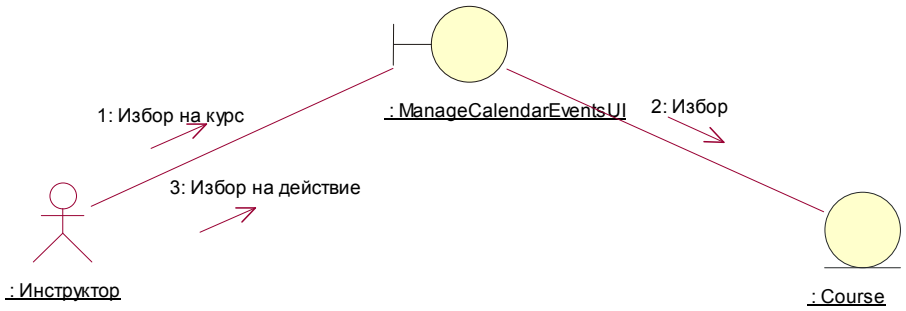
4.2.3.3. Реализация на случай на употреба **Управление на календарни събития**:

Фигура 4.2.3.3.1 изобразява диаграмата на класовете:



Фигура 4.2.3.3.1. Диаграма на класовете за случай на употреба Управление на календарни събития

Фигура 4.2.3.3.2 изобразява диаграмата на кооперирането:

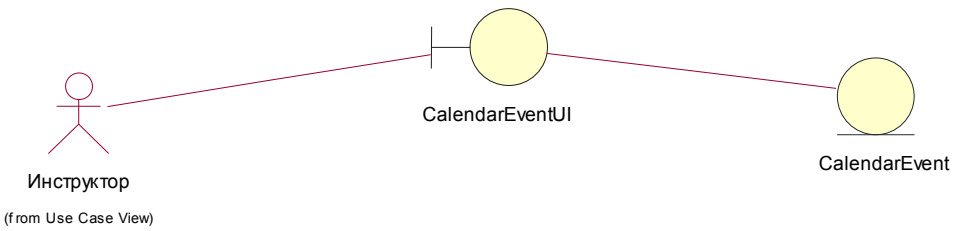


Фигура 4.2.3.3.2. Диаграма на кооперирането за случай на употреба Управление на календарни събития

Поток на събитията: Инструкторът избира курса, за който се отнася календарното събитие, за което желае да извърши нещо (1, 2). След това инструкторът избира действието, свързано с календарното събитие (създаване, редактиране или изтриване) (3).

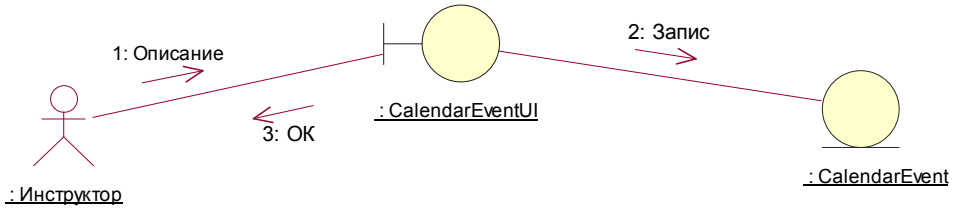
4.2.3.4. Реализация на случай на употреба **Създаване на календарно събитие**:

Фигура 4.2.3.4.1 изобразява диаграмата на класовете:



Фигура 4.2.3.4.1. Диаграма на класовете за случай на употреба Създаване на календарно събитие

Фигура 4.2.3.4.2 изобразява диаграмата на кооперирането:

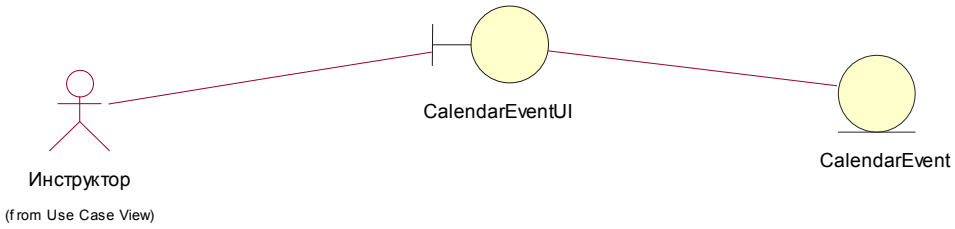


Фигура 4.2.3.4.2. Диаграма на кооперирането за случай на употреба Създаване на календарно събитие

Поток на събитията: Инструкторът описва характеристиките на новото календарно събитие (име, тип, ...) (1). Класът CalendarEventUI записва новото календарно събитие (2).

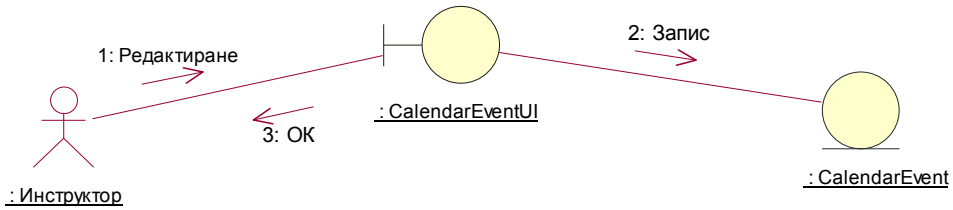
4.2.3.5. Реализация на случай на употреба **Редактиране на календарно събитие**:

Фигура 4.2.3.5.1 изобразява диаграмата на класовете:



Фигура 4.2.3.5.1. Диаграма на класовете за случай на употреба Редактиране на календарно събитие

Фигура 4.2.3.5.2 изобразява диаграмата на кооперирането:

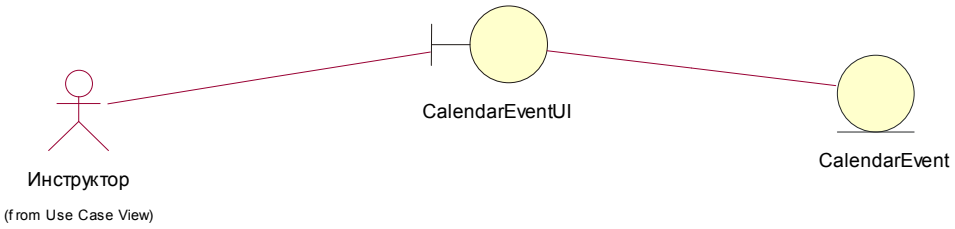


Фигура 4.2.3.5.2. Диаграма на кооперирането за случай на употреба Редактиране на календарно събитие

Поток на събитията: Инструкторът редактира характеристиките на избраното от него календарно събитие (име, тип, ...) (1). Класът CalendarEventUI записва редактираното календарно събитие (2).

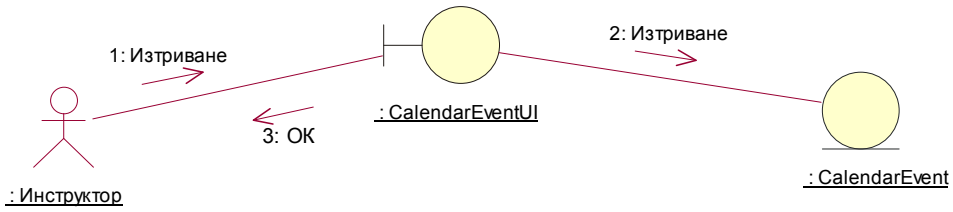
4.2.3.6. Реализация на случай на употреба **Изтриване на календарно събитие**:

Фигура 4.2.3.6.1 изобразява диаграмата на класовете:



Фигура 4.2.3.6.1. Диаграма на класовете за случай на употреба Изтриване на календарно събитие

Фигура 4.2.3.6.2 изобразява диаграмата на кооперирането:

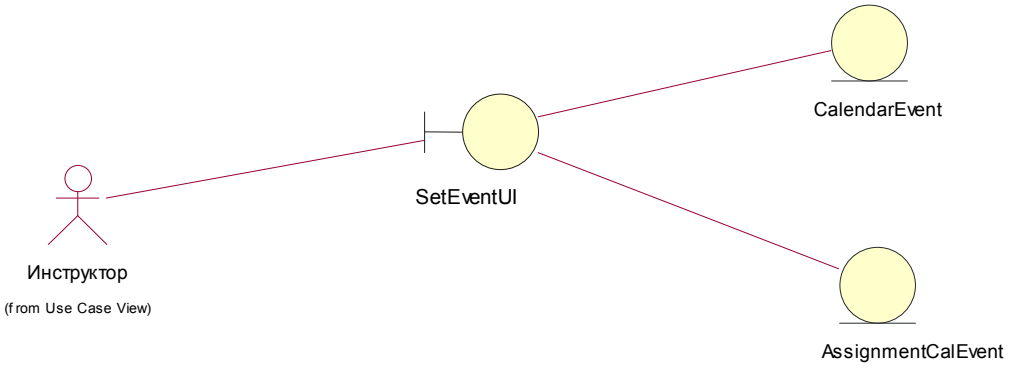


Фигура 4.2.3.6.2. Диаграма на кооперирането за случай на употреба Изтриване на календарно събитие

Поток на събитията: Инструкторът избира да изтрие съответното календарно събитие (1). Класът CalendarEventUI изтрива избраното календарно събитие (2).

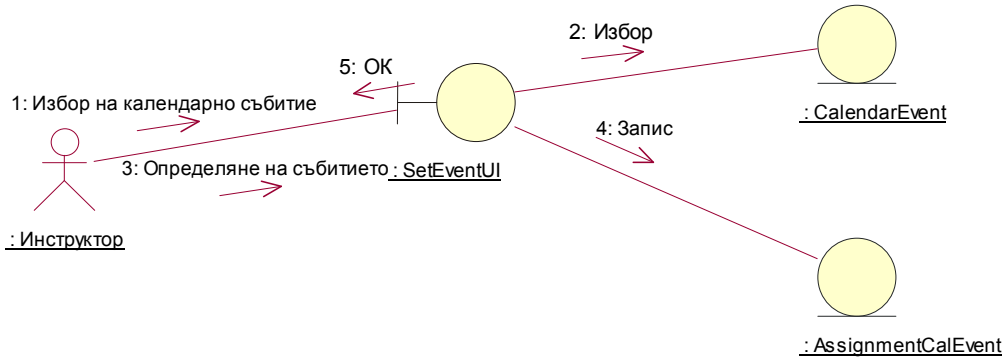
4.2.3.7. Реализация на случай на употреба **Определяне на събитие за задание**:

Фигура 4.2.3.7.1 изобразява диаграмата на класовете:



Фигура 4.2.3.7.1. Диаграма на класовете за случай на употреба Определяне на събитие за задание

Фигура 4.2.3.7.2 изобразява диаграмата на кооперирането:

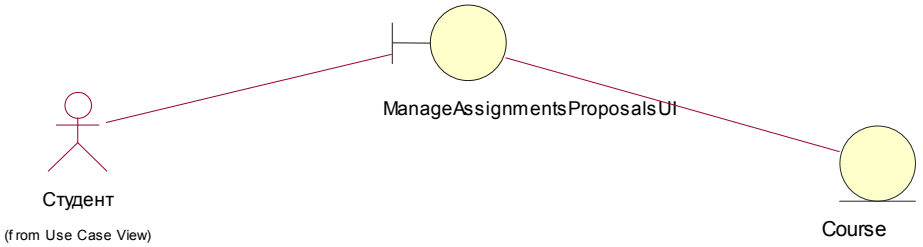


Фигура 4.2.3.7.2. Диаграма на кооперирането за случай на употреба Определяне на събитие за задание

Поток на събитията: Инструкторът избира календарното събитие за съответното задание чрез SetEventUI (1, 2). Инструкторът задава събитието (3). SetEventUI записва събитието за избраното задание (4).

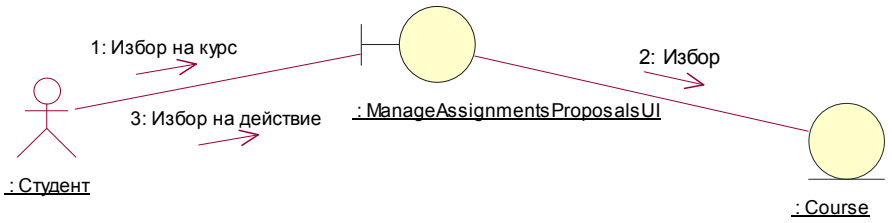
4.2.3.8. Реализация на случай на употреба **Управление на предложения за задания**:

Фигура 4.2.3.8.1 изобразява диаграмата на класовете:



Фигура 4.2.3.8.1. Диаграма на класовете за случай на употреба Управление на предложения за задания

Фигура 4.2.3.8.2 изобразява диаграмата на кооперирането:

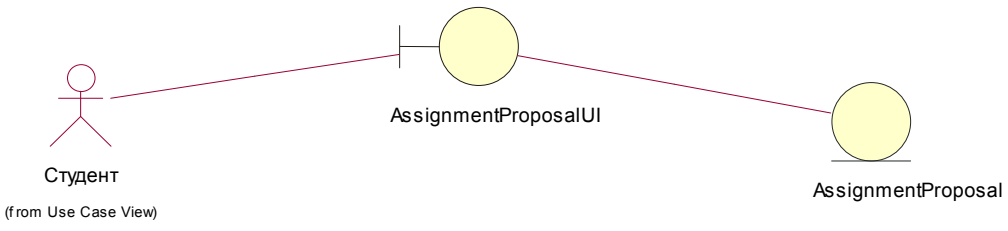


Фигура 4.2.3.8.2. Диаграма на кооперирането за случай на употреба Управление на предложения за задания

Поток на събитията: Студентът избира курса, за който се отнасят предложенията за задания (1, 2) (Тук актьор би могъл да бъде и инструкторът). След това студентът избира действието, свързано с предложението за задание (създаване, редактиране или изтриване за студент и редактиране, одобрение или отказ за инструктор) (3).

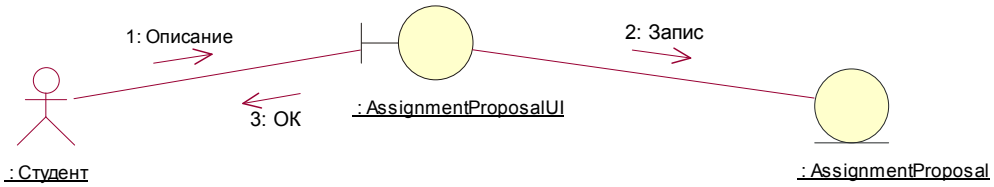
4.2.3.9. Реализация на случай на употреба **Създаване на предложение за задание**:

Фигура 4.2.3.9.1 изобразява диаграмата на класовете:



Фигура 4.2.3.9.1. Диаграма на класовете за случай на употреба Създаване на предложение за задание

Фигура 4.2.3.9.2 изобразява диаграмата на кооперирането:

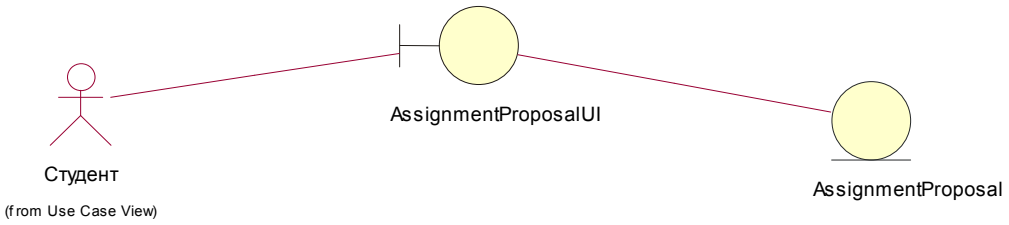


Фигура 4.2.3.9.2. Диаграма на кооперирането за случай на употреба Създаване на предложение за задание

Поток на събитията: Студентът описва характеристиките на новото предложение за задание (име, тип, описание, ...) (1). Класът AssignmentProposalUI записва новото предложение за задание (2).

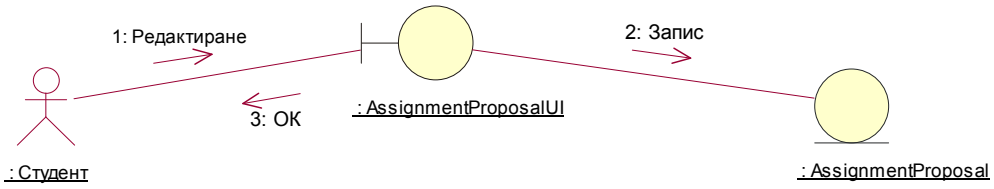
4.2.3.10. Реализация на случай на употреба **Редактиране на предложение за задание**:

Фигура 4.2.3.10.1 изобразява диаграмата на класовете:



Фигура 4.2.3.10.1. Диаграма на класовете за случай на употреба Редактиране на предложение за задание

Фигура 4.2.3.10.2 изобразява диаграмата на кооперирането:

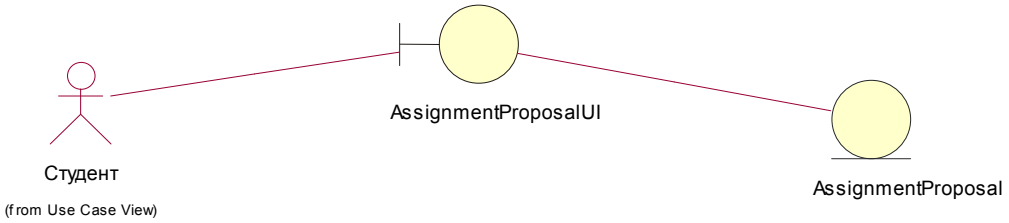


Фигура 4.2.3.10.2. Диаграма на кооперирането за случай на употреба Редактиране на предложение за задание

Поток на събитията: Студентът редактира характеристиките на избраното предложение за задание (име, тип, описание, ...) (1). Класът AssignmentProposalUI записва редактираното предложение за задание (2).

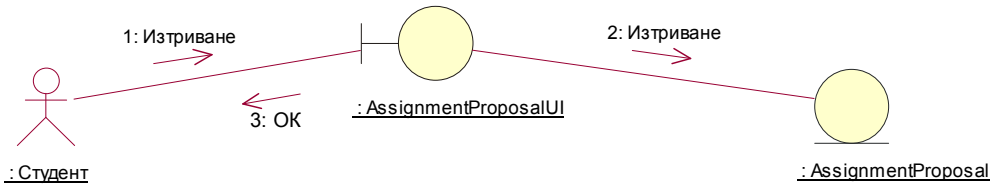
4.2.3.11. Реализация на случай на употреба **Изтриване на предложение за задание**:

Фигура 4.2.3.11.1 изобразява диаграмата на класовете:



Фигура 4.2.3.11.1. Диаграма на класовете за случай на употреба Изтриване на предложение за задание

Фигура 4.2.3.11.2 изобразява диаграмата на кооперирането:



Фигура 4.2.3.11.2. Диаграма на кооперирането за случай на употреба Изтриване на предложение за задание

Поток на събитията: Студентът избира да изтрие съответното предложение за задание (1). Класът AssignmentProposalUI изтрива избраното предложение за задание (2).

4.3. Дизайн

Дизайнът е третата фаза от разработването на системата. Основа за дизайна е моделът на анализа. Дизайнът представлява детайлно описание на системата в съответствие с избрания език за програмиране, технологии за разработка, операционна система, метод на съхраняване на данните (база от данни), потребителски интерфейс и т.н. Резултатът е дизайн модел, който описва системата в термините на подсистеми, интерфейси и класове. Това не са абстрактни класове както при анализа, а конкретни такива, които ще се реализират в следващия етап - имплементацията.

4.3.1. Модел на дизайна

Моделът на дизайна описва физическата реализация на случаите на употреба, като се фокусира върху начина, по който функционалните и

нефункционалните изисквания, както и другите ограничения, свързани със средата на изпълнение, влияят върху разглежданата система. Този модел се използва като основа във фазата на имплементацията. Моделът на дизайна се състои от класове на дизайна, реализация на случаите на употреба и интерфейси. Класовете на дизайна представляват абстракции на компоненти в имплементацията на системата. Случаите на употреба се реализират чрез дизайн класове и техните обекти. Това се представя чрез кооперирane в модела на дизайна и се обозначава като реализация на случаите на употреба в дизайна.

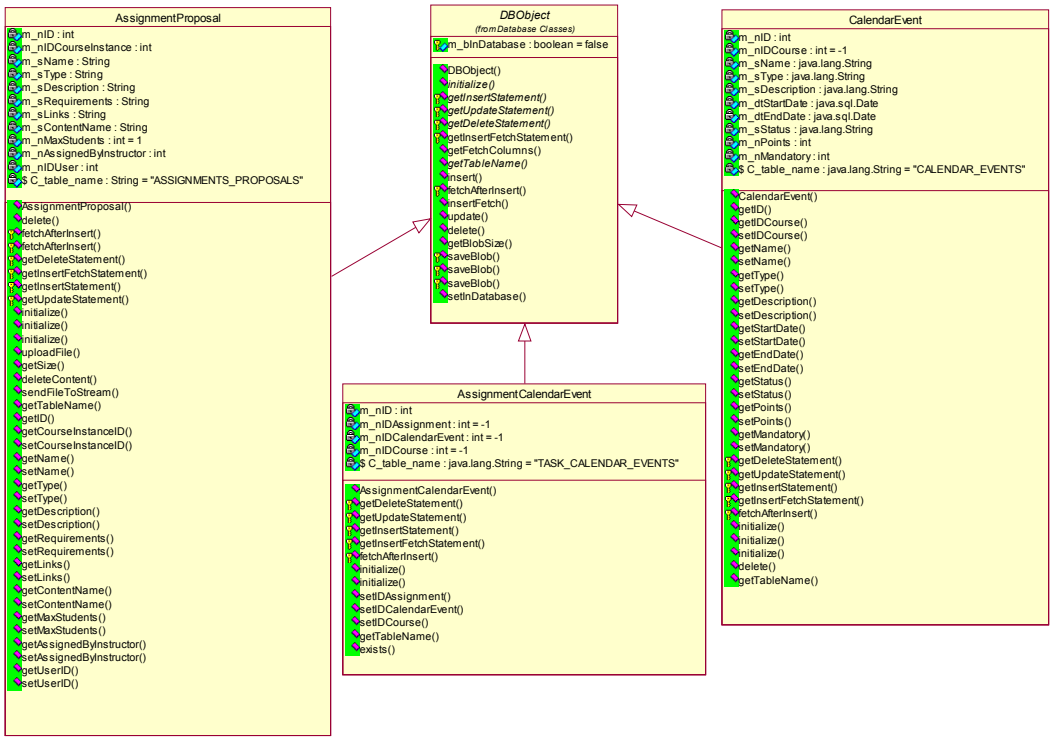
4.3.2. Клас на дизайна

Класът на дизайна представлява абстракция на клас или подобен компонент от имплементацията на системата. Езикът, използван за специфициране на дизайн класа, е същият, който се използва за програмирането му. Операциите, параметрите, атрибутите, типовете и т.н. се специфицират, използвайки синтаксис на избран език за програмиране. За разработвания модул този език е Java. Видимостта на атрибутите и операциите на дизайн класа също се указва (*public, protected, private* в Java). Дизайн класът може да отложи обработката на някои изисквания за имплементацията. Дизайн класът може да реализира интерфейс, ако това е смислена операция в програмния език. В разработвания модул за заданията участват следните класове:

- ✓ AssignmentProposal – капсулира таблицата assignments_proposals, предоставя методи за четене и запис на характеристиките на дадено предложение за задание, наследява класа arcade.database.DBObject
- ✓ CalendarEvent – капсулира таблицата calendar_events, предоставя методи за четене и запис на характеристиките на дадено календарно събитие, наследява класа arcade.database.DBObject
- ✓ AssignmentCalendarEvent – капсулира таблицата task_calendar_events, предоставя методи за четене и запис на характеристиките на такъв вид обект, представлява връзката между едно задание и едно календарно събитие, всяко задание трябва да

има съответно календарно събитие, наследява класа arcade.database.DBObject

На фигура 4.3.2.1 са изобразени класовете, наследници на класа arcade.database.DBObject.

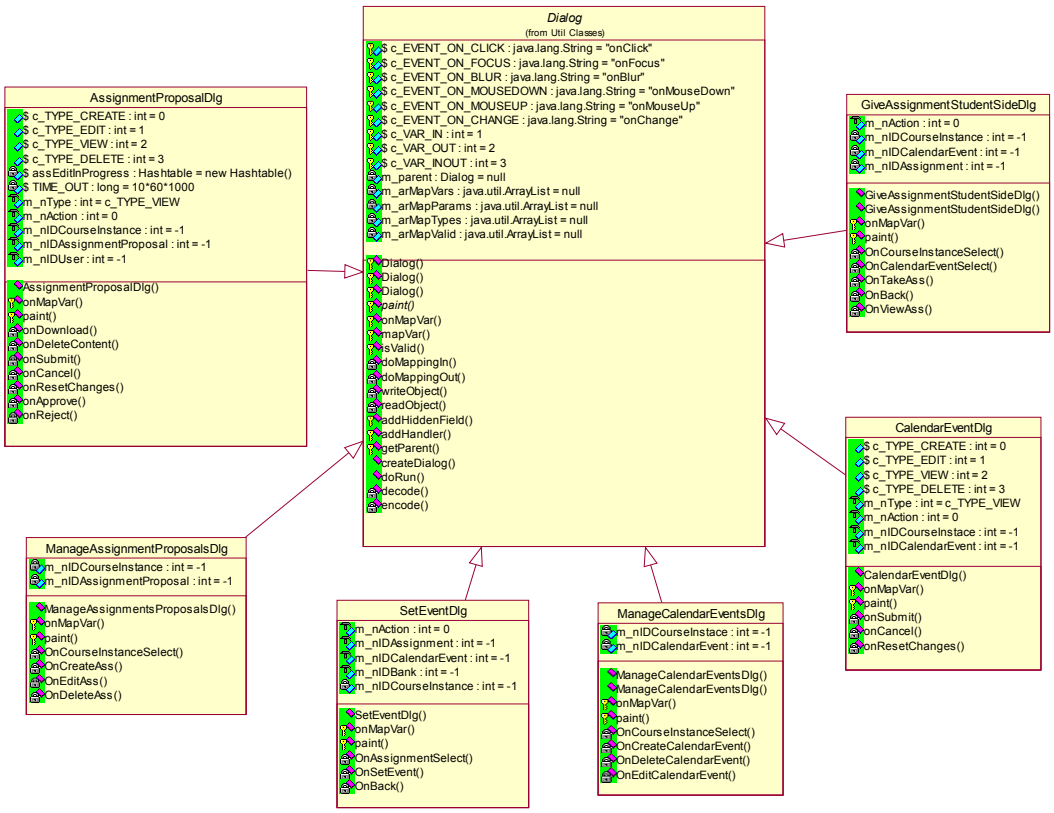


Фигура 4.3.2.1. Диаграма на класовете, наследници на класа arcade.database.DBObject

- ✓ AssignmentProposalDlg – капсулира бизнес логиката за създаване, редактиране и изтриване на предложение за задание, генерира html кода за потребителския интерфейс, наследява класа arcade.util.Dialog
- ✓ ManageAssignmentProposalsDlg – капсулира бизнес логиката за управление на предложения за задания, генерира html кода за потребителския интерфейс, наследява класа arcade.util.Dialog
- ✓ CalendarEventDlg – капсулира бизнес логиката за създаване, редактиране и изтриване на календарно събитие, генерира html кода за потребителския интерфейс, наследява класа arcade.util.Dialog

- ✓ ManageCalendarEventsDlg – капсулира бизнес логиката за управление на календарни събития, генерира html кода за потребителския интерфейс, наследява класа arcade.util.Dialog
- ✓ SetEventDlg – капсулира бизнес логиката за определяне на календарно събитие за дадено задание, генерира html кода за потребителския интерфейс, наследява класа arcade.util.Dialog
- ✓ GiveAssignmentStudentSideDlg – капсулира бизнес логиката за избор на задание от студент, генерира html кода за потребителския интерфейс, наследява класа arcade.util.Dialog

На фигура 4.3.2.2 са изобразени класовете, наследници на класа arcade.util.Dialog.



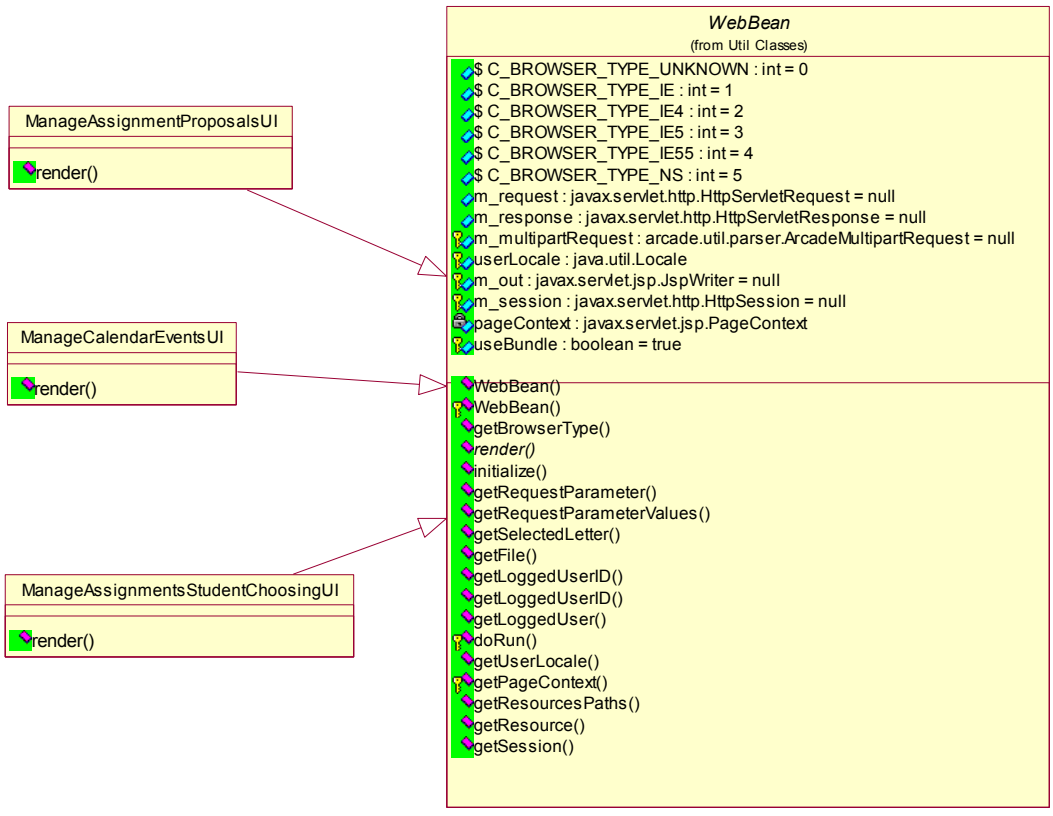
Фигура 4.3.2.2. Диаграма на класовете наследници на класа arcade.util.Dialog

- ✓ ManageAssignmentProposalsUI – имплементира управлението на предложения за задания от инструктора и студента, извиква класа

ManageAssignmentProposalsDlg, който имплементира потребителския интерфейс и бизнес логиката, наследява класа arcade.util.WebBean

- ✓ ManageCalendarEventsUI – имплементира управлението на календарни събития от инструктора, извиква класа ManageCalendarEventsDlg, който имплементира потребителския интерфейс и бизнес логиката, наследява класа arcade.util.WebBean
- ✓ ManageAssignmentsStudentChoosingUI – имплементира управлението на задания от студента, извиква класа GiveAssignmentStudentSideDlg, който имплементира потребителския интерфейс и бизнес логиката, наследява класа arcade.util.WebBean

На фигура 4.3.2.3 са изобразени класовете, наследници на класа arcade.util.WebBean.



Фигура 4.3.2.3. Диаграма на класовете наследници на класа arcade.util.WebBean

4.3.3. Реализация на случаите на употреба

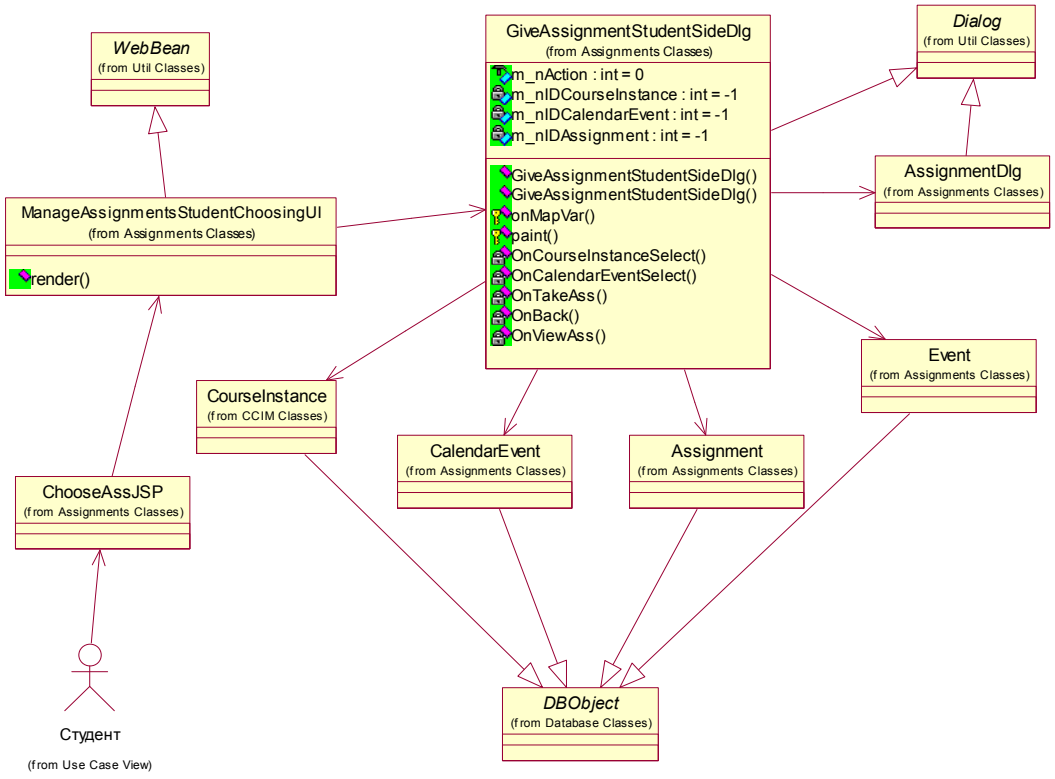
Реализацията на случаите на употреба представлява кооперирane в модела на дизайна, което описва как се реализират специфичните случаи на употреба и как се представят в термините на дизайн класове и техните обекти. Реализацията на определен случай на употреба в дизайна се състои от текстово описание на потока на събитията, диаграми на класовете, описващи участващите дизайн класове, и диаграми на взаимодействията, описващи реализацията на отделен, специфичен поток или сценарий на случая на употреба в термините на взаимодействия между дизайн обекти.

Диаграми на класовете Един дизайн клас и неговите обекти често участват в няколко реализации на случаи на употреба. Възможно е също някои операции, атрибути и асоциации на определен клас да са приложими за реализация само на един случай на употреба. Важно е да бъдат координирани всички изисквания за даден клас и неговите обекти, които реализациите на различните случаи на употреба налагат. За тази цел в реализацията на случаи на употреба се използват диаграми на класовете, показващи участващите класове и техните връзки. По този начин се следят участващите елементи в реализацията на случаите на употреба. ***Диаграми на взаимодействията*** Последователността на действията в един случай на употреба започва, когато актьорът извика случая на употреба, като изпрати съобщение на системата. В този момент трябва да има някакъв обект от дизайна, който да получи това съобщение от актьора. След това дизайн обектът извиква някой друг обект и така обектите взаимодействат, за да реализират и осъществят случая на употреба. В дизайна това се описва чрез диаграми на последователностите, тъй като основната цел е откриването на подробна хронологична последователност на взаимодействията. С помощта на диаграмите на последователностите, взаимодействието на обектите се илюстрира чрез предаване на съобщения между техните линии (линии на обектите се използват в самите диаграми на последователности).

Поток на събитията Често диаграмите, особено тези на взаимодействията, са трудни за четене. Затова се използва поток на събитията, което представлява текстово описание на диаграмите и техните етикети. Текстът е в термините на обектите, които взаимодействат, за да изпълнят случая на употреба. В текстовото описание не се споменават атрибути на обектите, операции или асоциации, тъй като тези неща се променят често и описанието би станало прекалено сложно. Потокът на събития е особено полезен, когато има много диаграми на последователностите, които описват реализацията на един и същи случай на употреба, или когато има диаграми, които представят сложни потоци. Потокът на събития може да се използва за няколко диаграми, описвайки как те са свързани помежду си.

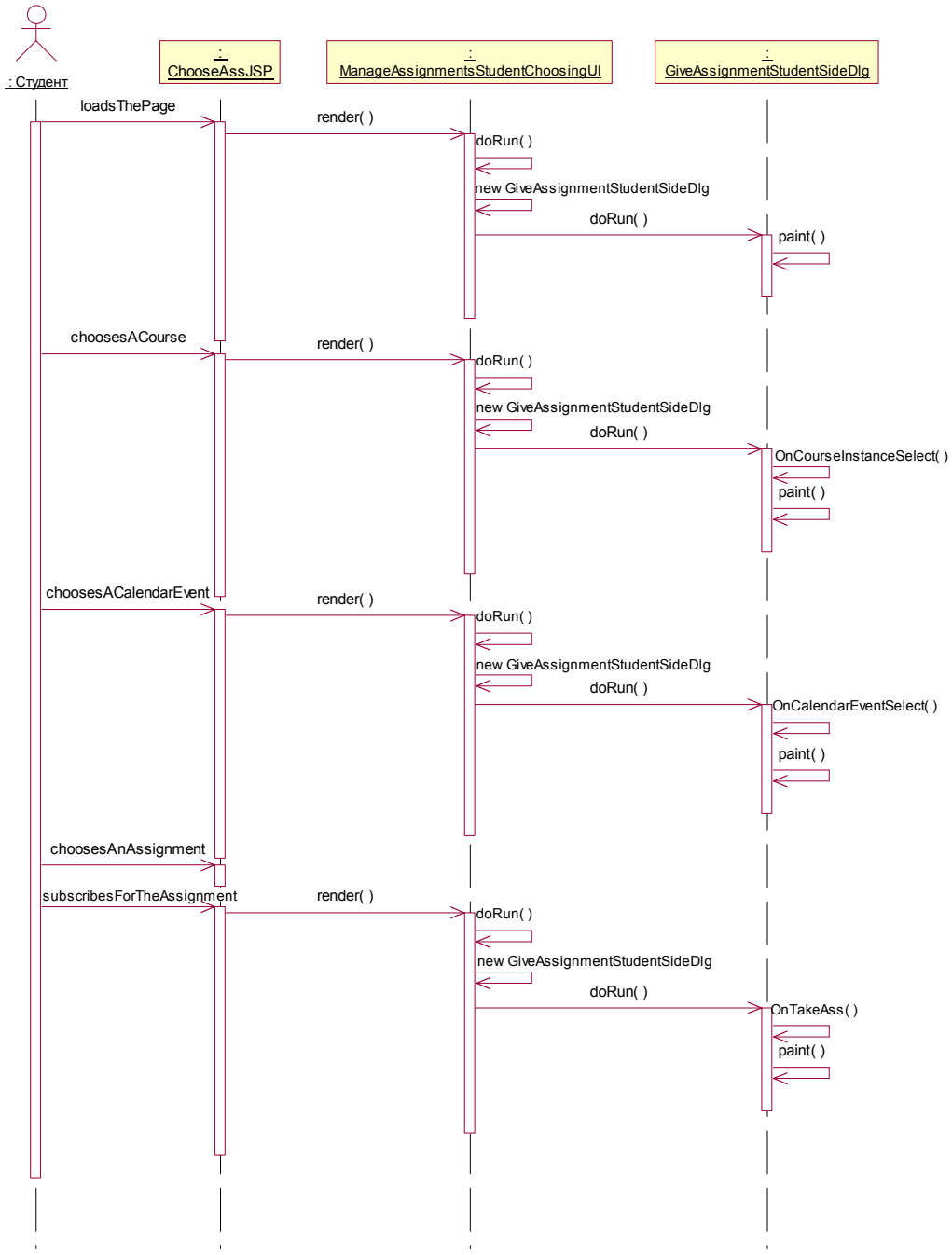
4.3.3.1. Реализация на случай на употреба **Записване за задание**:

Фигура 4.3.3.1.1 изобразява диаграмата на класовете:



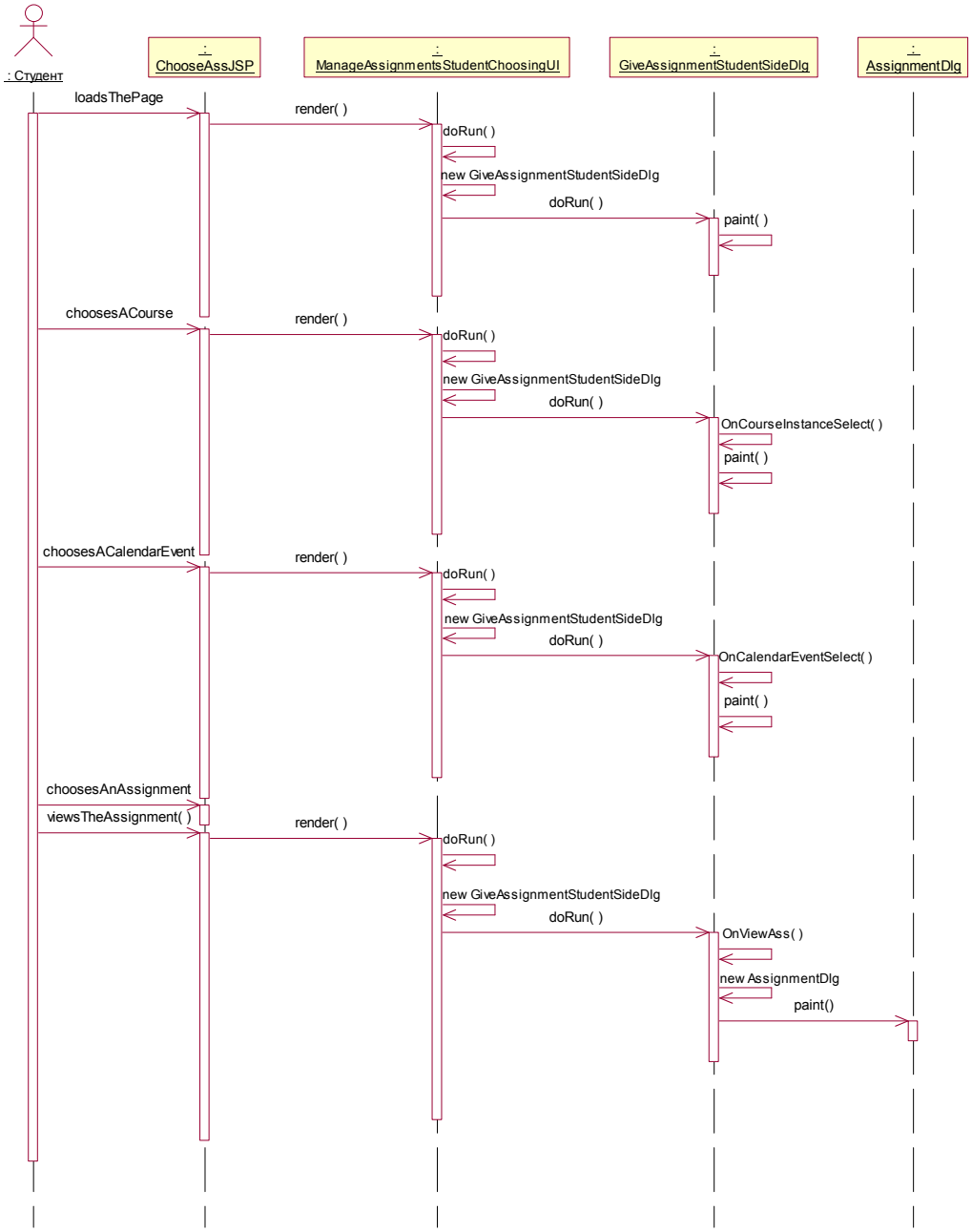
Фигура 4.3.3.1.1. Диаграма на класовете за случай на употреба Записване за задание

Фигура 4.3.3.1.2 изобразява диаграмата на последователностите:



Фигура 4.3.3.1.2. Диаграма на последователностите за случай на употреба Записване за задание

Диаграмата, реализираща разглеждане на задание в случая на употреба Записване за задание, е представена на фигура 4.3.3.1.3:



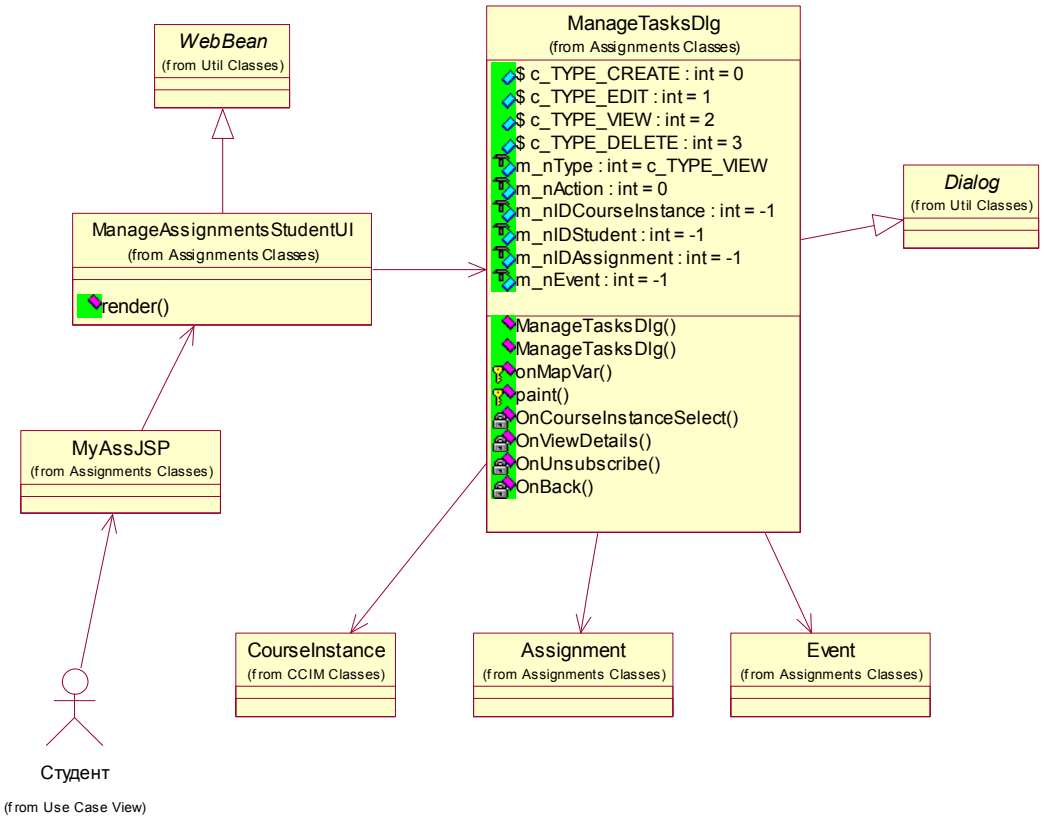
Фигура 4.3.3.1.3. Диаграма на последователностите, реализираща разглеждане на задание в случая на употреба Записване за задание

Поток на събитията: Обектът Студент комуникира със системата чрез обекта ChooseAssJSP. Най-напред Студентът зарежда страницата за избор на задание. Обектът ManageAssignmentsStudentChoosingUI създава диалога GiveAssignmentStudentSideDlg, който с помощта на метода paint()

генерира необходимия потребителски интерфейс. Студентът трябва да избере курса, за който ще се отнася заданието, календарното събитие и самото задание. Методът paint() на обекта GiveAssignmentStudentSideDlg съдържа бизнес логиката, необходима за обработката на избраните от Студента курс, календарно събитие, както и записването за избраното задание. В случай, че Студентът избере да разгледа избраното задание, GiveAssignmentStudentSideDlg създава обект от тип AssignmentDlg, който чрез своя метод paint() показва характеристиките на избраното задание.

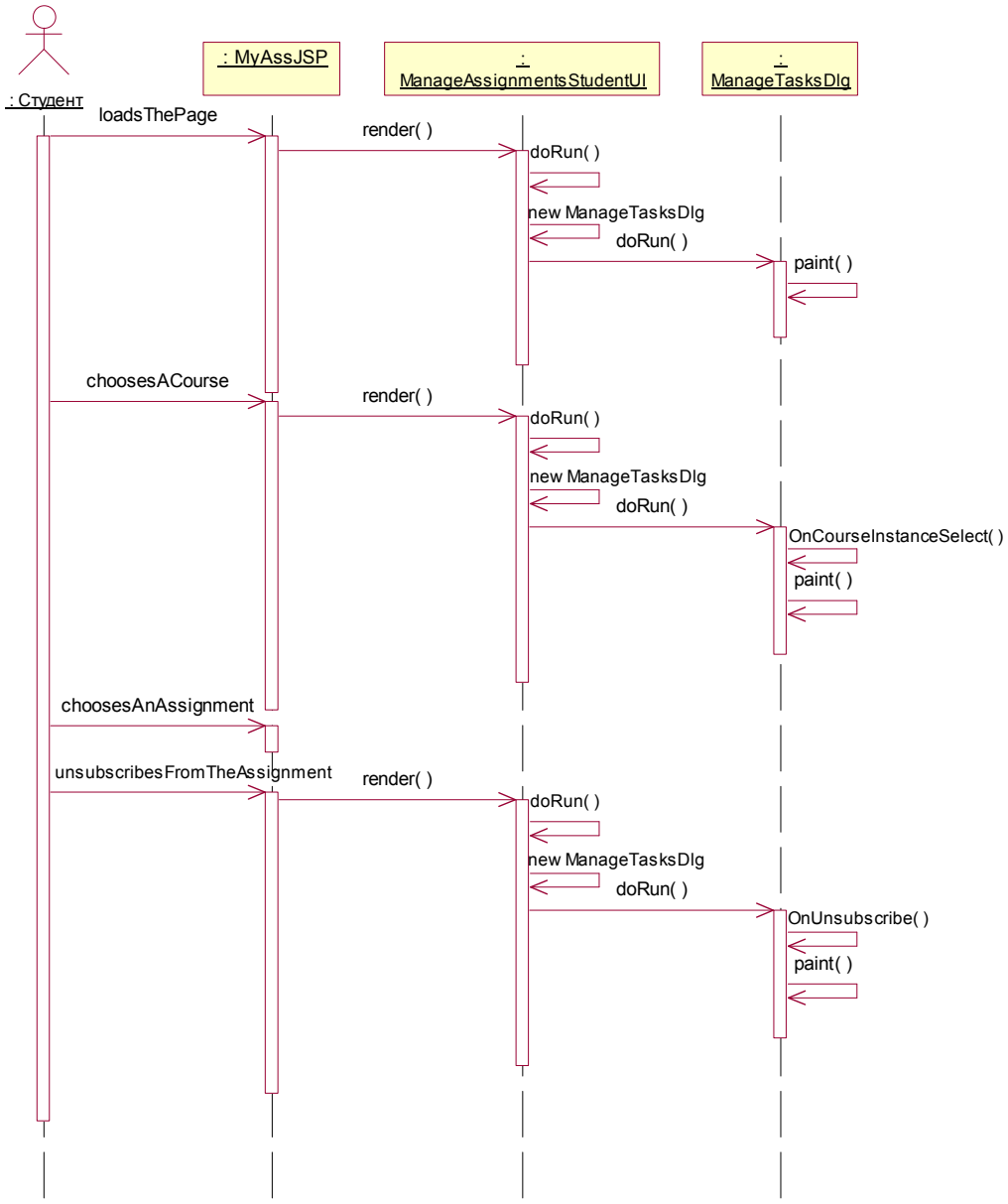
4.3.3.2. Реализация на случай на употреба **Отписване от задание**:

Фигура 4.3.3.2.1 изобразява диаграмата на класовете:



Фигура 4.3.3.2.1. Диаграма на класовете за случай на употреба Отписване от задание

Фигура 4.3.3.2.2 изобразява диаграмата на последователностите:



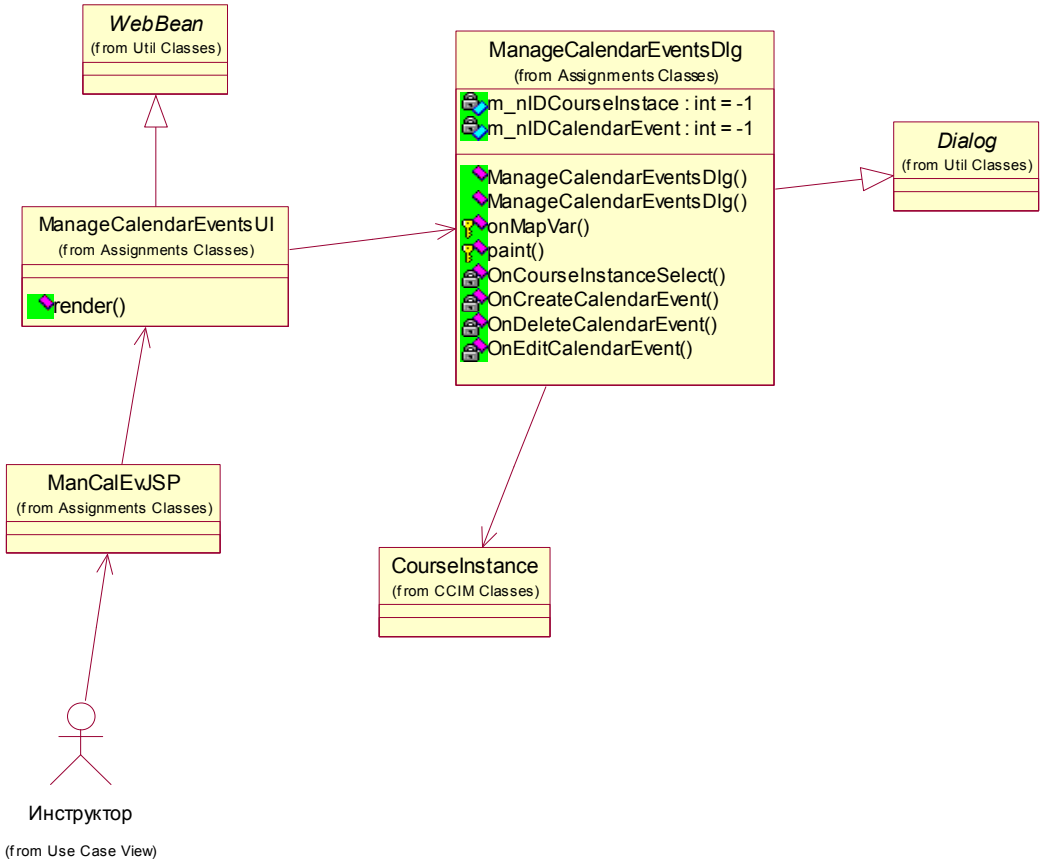
Фигура 4.3.3.2.2. Диаграма на последователностите за случай на употреба Отписване от задание

Поток на събитията: Обектът Студент комуникира със системата чрез обекта MyAssJSP. Най-напред Студентът зарежда страницата, съдържаща разпределените му задания. Обектът ManageAssignmentsStudentUI създава диалога ManageTasksDlg, който с помощта на метода paint() генерира необходимия потребителски интерфейс. Студентът трябва да избере курса,

за който ще се отнася заданието и самото задание, от което желае да се отпише. Методът paint() на обекта ManageTasksDlg съдържа бизнес логиката, необходима за обработката на избрания от Студента курс и отписването от избраното задание.

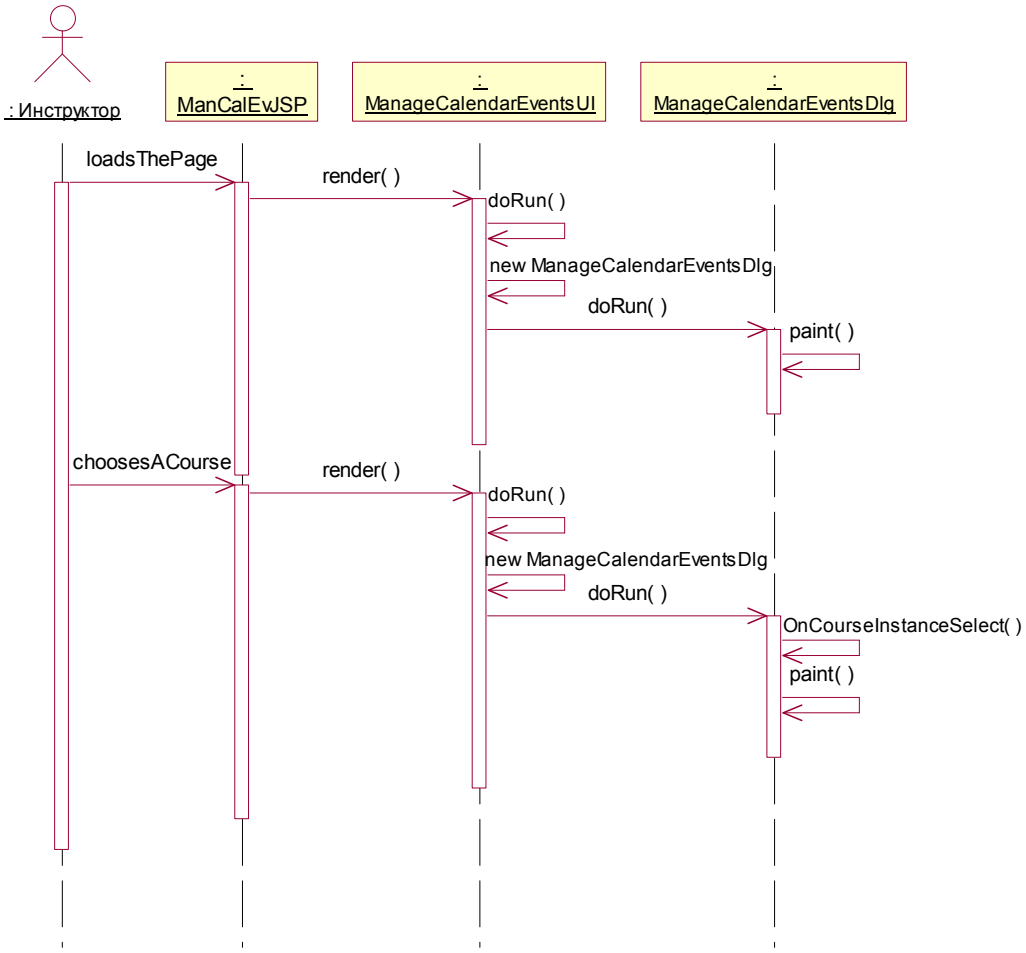
4.3.3.3. Реализация на случай на употреба **Управление на календарни събития**:

Фигура 4.3.3.3.1 изобразява диаграмата на класовете:



Фигура 4.3.3.3.1. Диаграма на класовете за случай на употреба Управление на календарни събития

Фигура 4.3.3.3.2 изобразява диаграмата на последователностите:

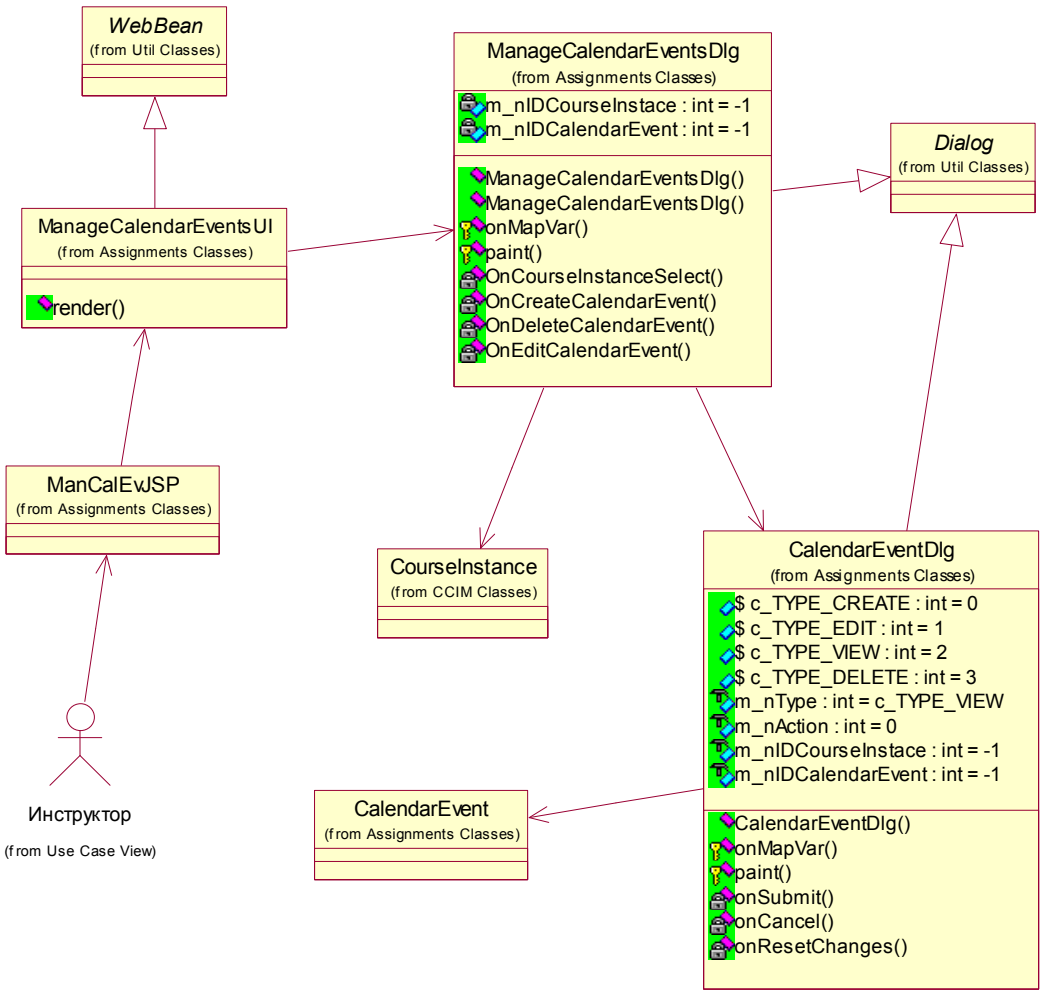


Фигура 4.3.3.3.2. Диаграма на последователностите за случай на употреба Управление на календарни събития

Поток на събитията: Обектът Инструктор комуникира със системата чрез обекта ManCalEvJSP. Най-напред Инструкторът зарежда страницата за управление на календарни събития. Обектът ManageCalendarEventsUI създава диалога ManageCalendarEventsDlg, който с помощта на метода paint() генерира необходимия потребителски интерфейс. Инструкторът избира курса, за който се отнасят календарните събития. Методът paint() на обекта ManageCalendarEventsDlg съдържа бизнес логиката, необходима за обработката на избрания курс. За избрания курс Инструкторът има възможност да избере между три действия – създаване, редактиране и изтриване на календарно събитие.

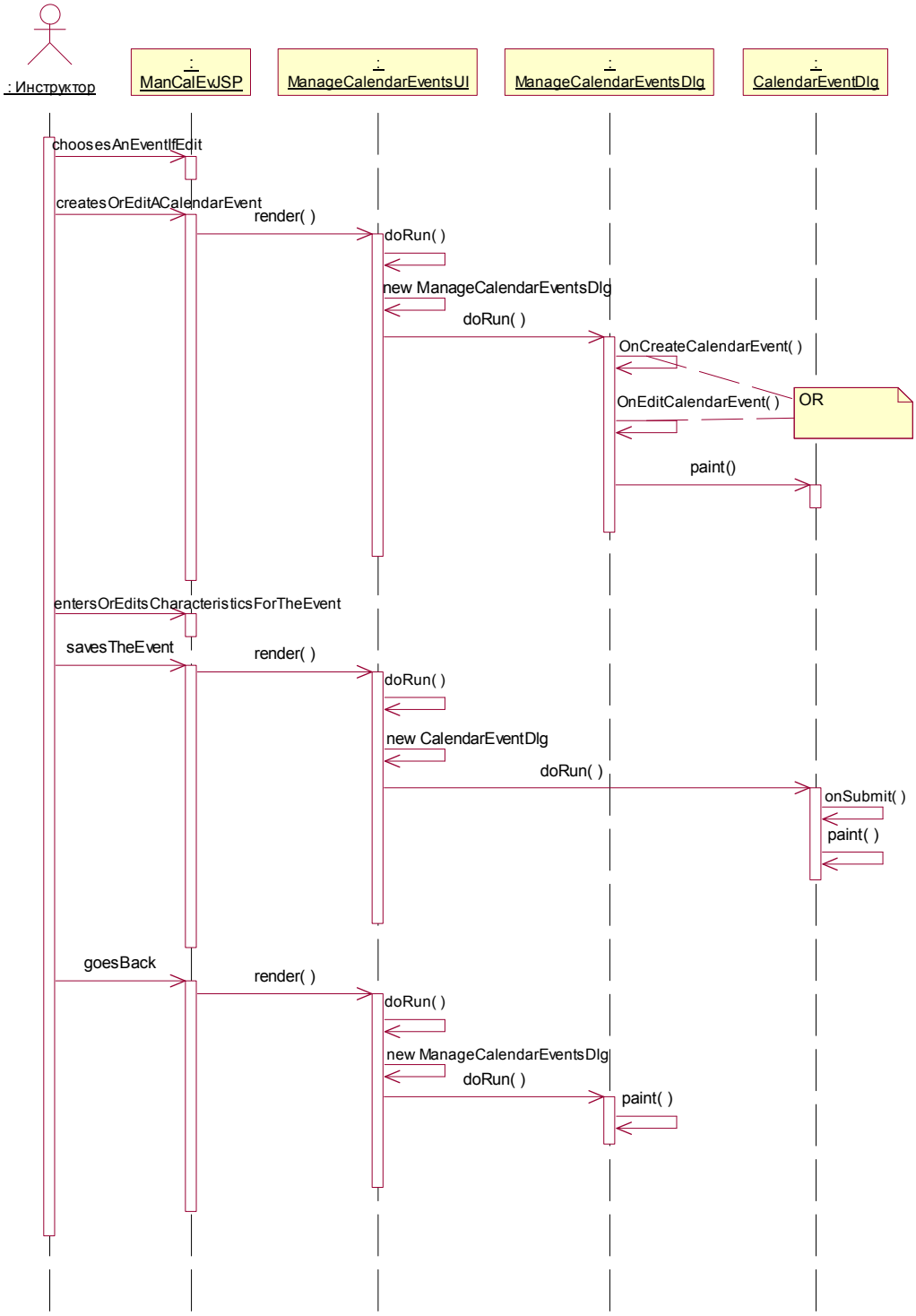
4.3.3.4. Реализация на случаи на употреба **Създаване/Редактиране/Изтриване на календарно събитие**:

Фигура 4.3.3.4.1 изобразява диаграмата на класовете:



Фигура 4.3.3.4.1. Диаграма на класовете за случаи на употреба Създаване/Редактиране/Изтриване на календарно събитие

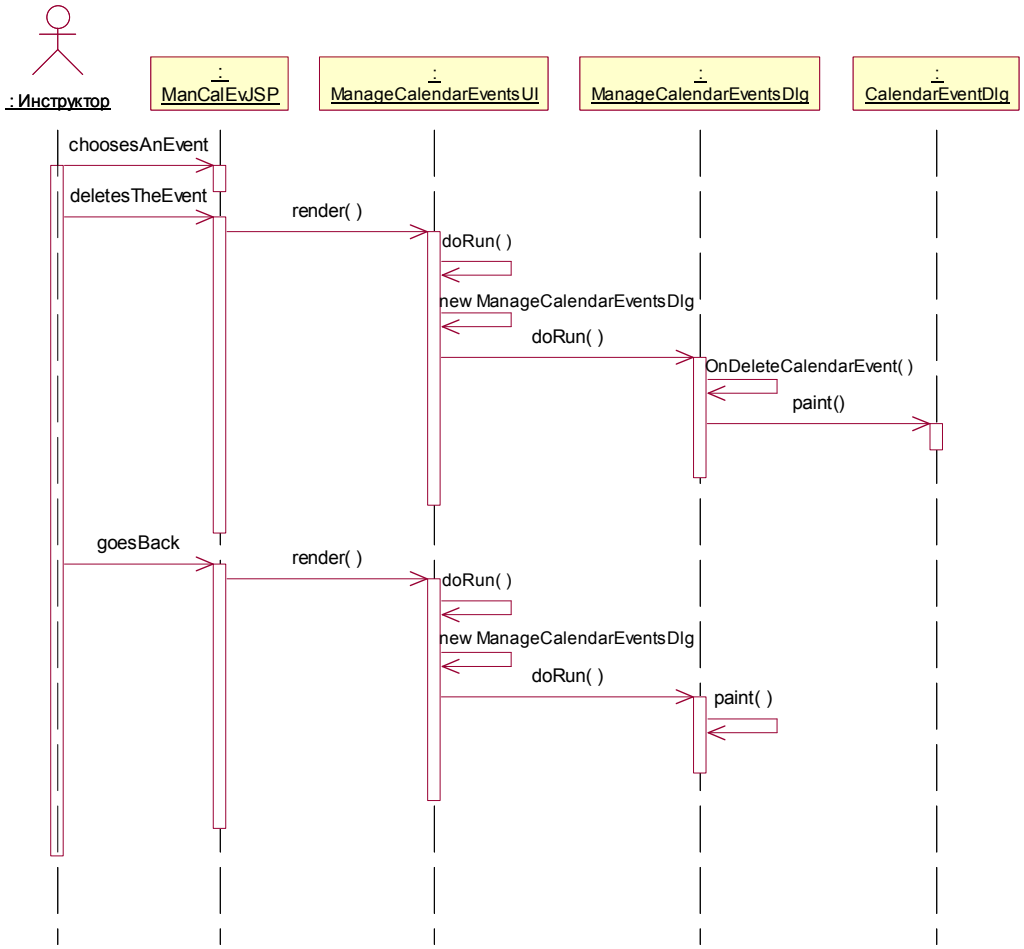
Диаграмата на последователностите за създаване и редактиране на календарно събитие е изобразена на фигура 4.3.3.4.2:



Фигура 4.3.3.4.2. Диаграма на последователностите за случаи на употреба

Създаване/Редактиране на календарно събитие

Диаграмата на последователностите за изтриване на календарно събитие е изобразена на фигура 4.3.3.4.3:



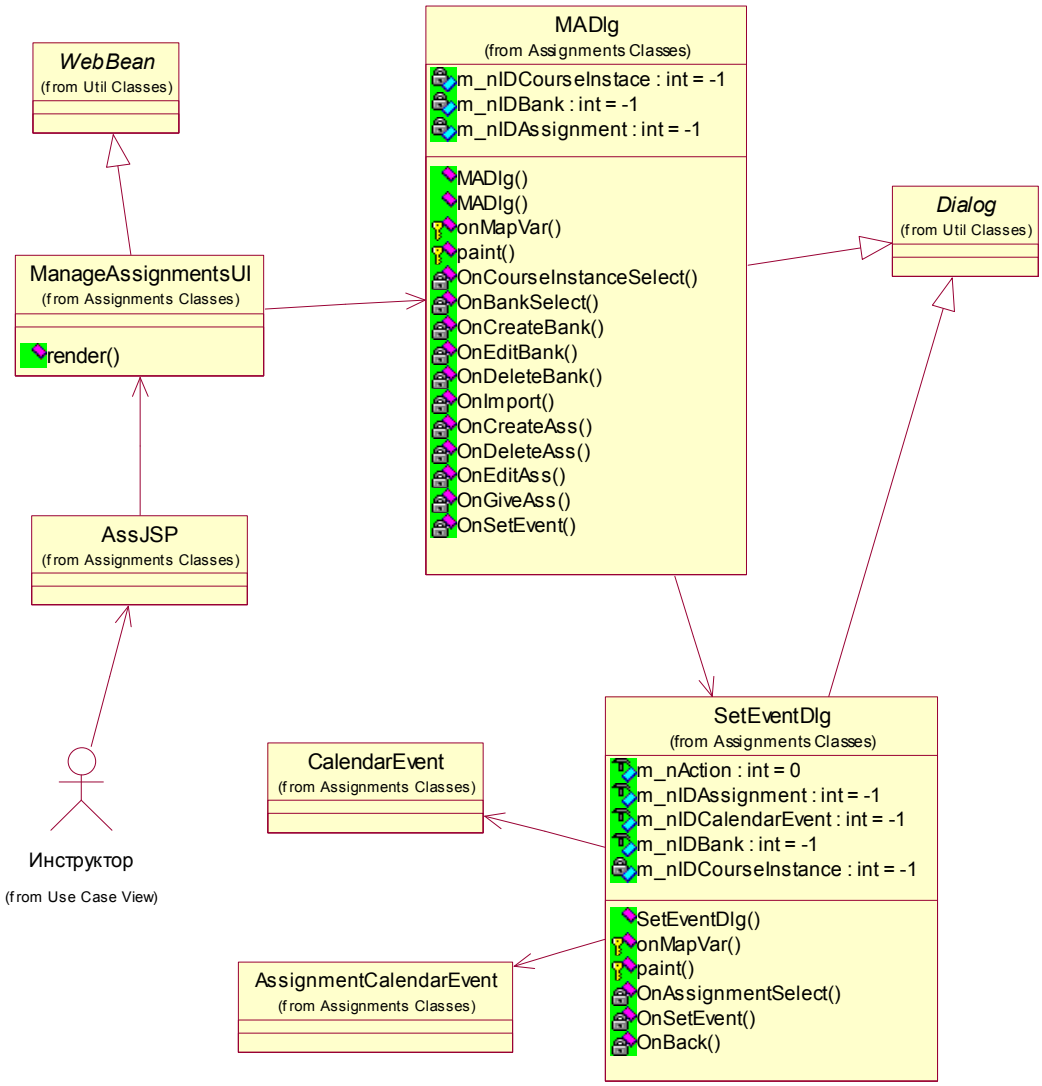
Фигура 4.3.3.4.3. Диаграма на последователностите за случай на употреба Изтриване на календарно събитие

Поток на събитията: Обектът Инструктор комуникира със системата чрез обекта ManCalEvJSP. Той е заредил страницата за управление на календарни събития и е избрал курса, за който се отнасят събитията. За случаите на редактиране и изтриване на календарно събитие Инструкторът избира събитието, за което ще се отнася действието. След това Инструкторът избира действието. Обектът ManageCalendarEventsUI създава диалога ManageCalendarEventsDlg, който от своя страна създава обект от тип CalendarEventDlg, чийто метод paint() генерира необходимия

потребителски интерфейс. Той съдържа също и бизнес логиката, необходима за създаването, редактирането и изтриването на календарно събитие.

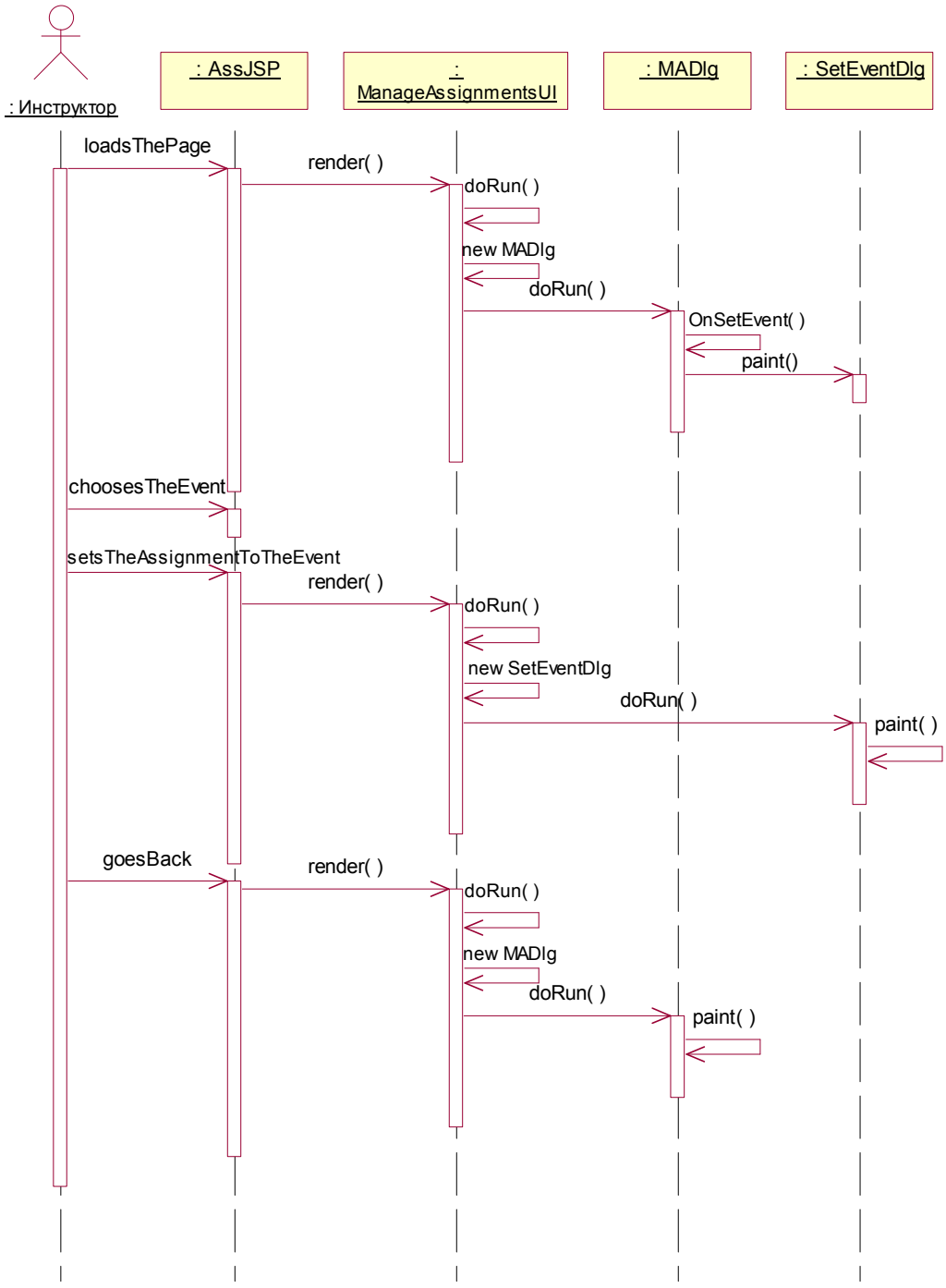
4.3.3.5. Реализация на случай на употреба **Определяне на събитие за задание**:

Фигура 4.3.3.5.1 изобразява диаграмата на класовете:



Фигура 4.3.3.5.1. Диаграма на класовете за случай на употреба Определяне на събитие за задание

Фигура 4.3.3.5.2 изобразява диаграмата на последователностите:



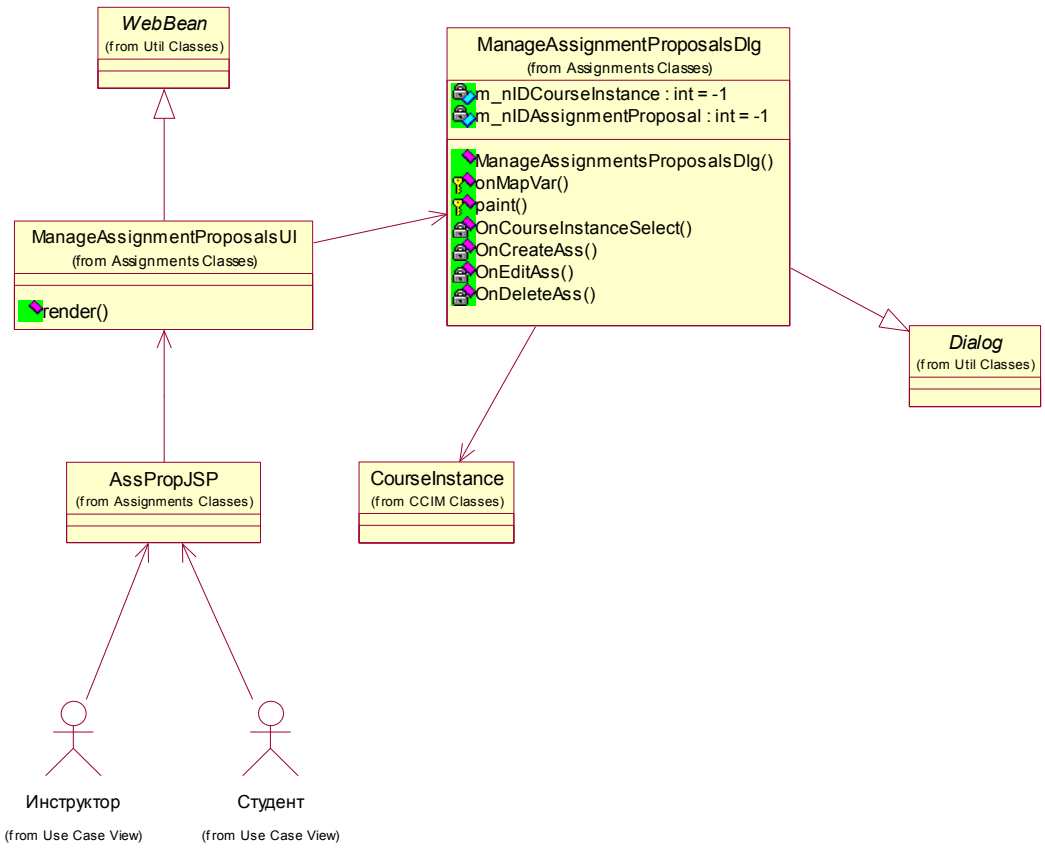
Фигура 4.3.3.5.2. Диаграма на последователностите за случай на употреба Определяне на събитие за задание

Поток на събитията: Обектът Инструктор комуникира със системата чрез обекта AssJSP. Той е заредил страницата за управление на заданията и е

избрал курса, банката със задания и самото задание, за което ще определя календарно събитие. Инструкторът избира да зададе събитието. Обектът ManageAssignmentsUI създава диалога MADlg, който от своя страна създава обект от тип SetEventDlg, чийто метод paint() генерира необходимия потребителски интерфейс. Инструкторът избира желаното календарно събитие. Методът paint() на обекта SetEventDlg съдържа също и бизнес логиката, необходима за определянето на календарно събитие за избрано задание.

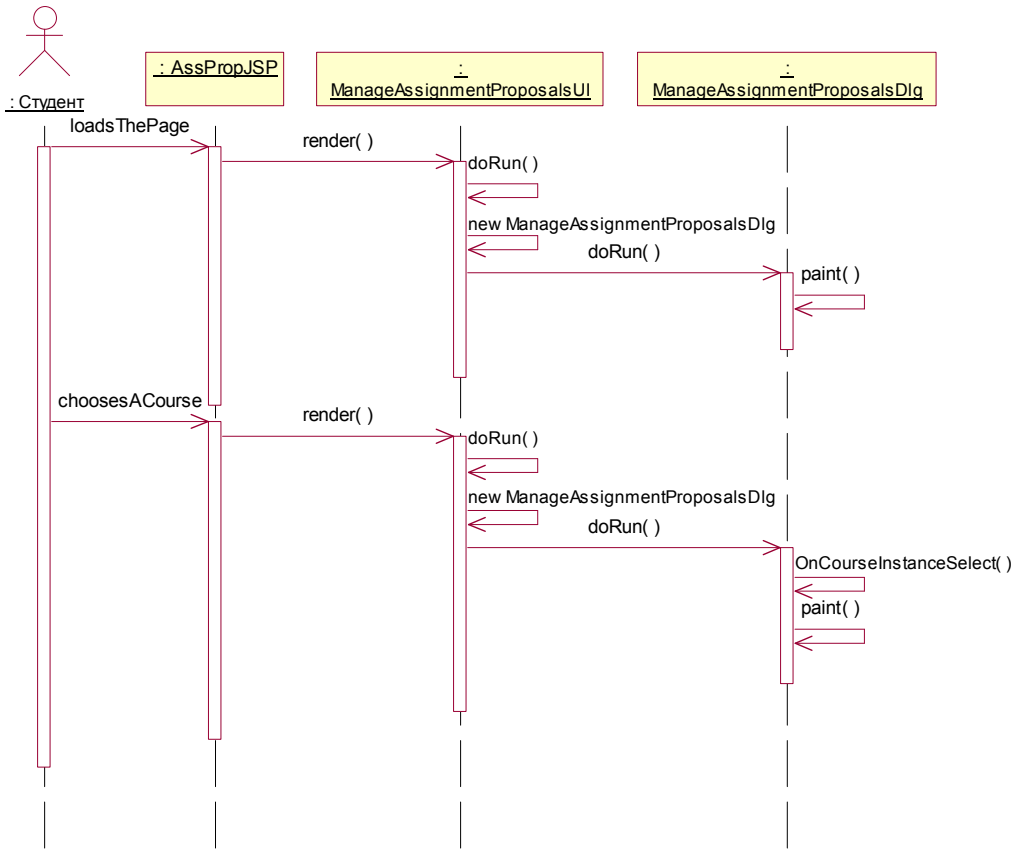
4.3.3.6. Реализация на случай на употреба **Управление на предложения за задания**:

Фигура 4.3.3.6.1 изобразява диаграмата на класовете:



Фигура 4.3.3.6.1. Диаграма на класовете за случай на употреба Управление на предложения за задания

Фигура 4.3.3.6.2 изобразява диаграмата на последователностите:



Фигура 4.3.3.6.2. Диаграма на последователностите за случай на употреба Управление на предложения за задания

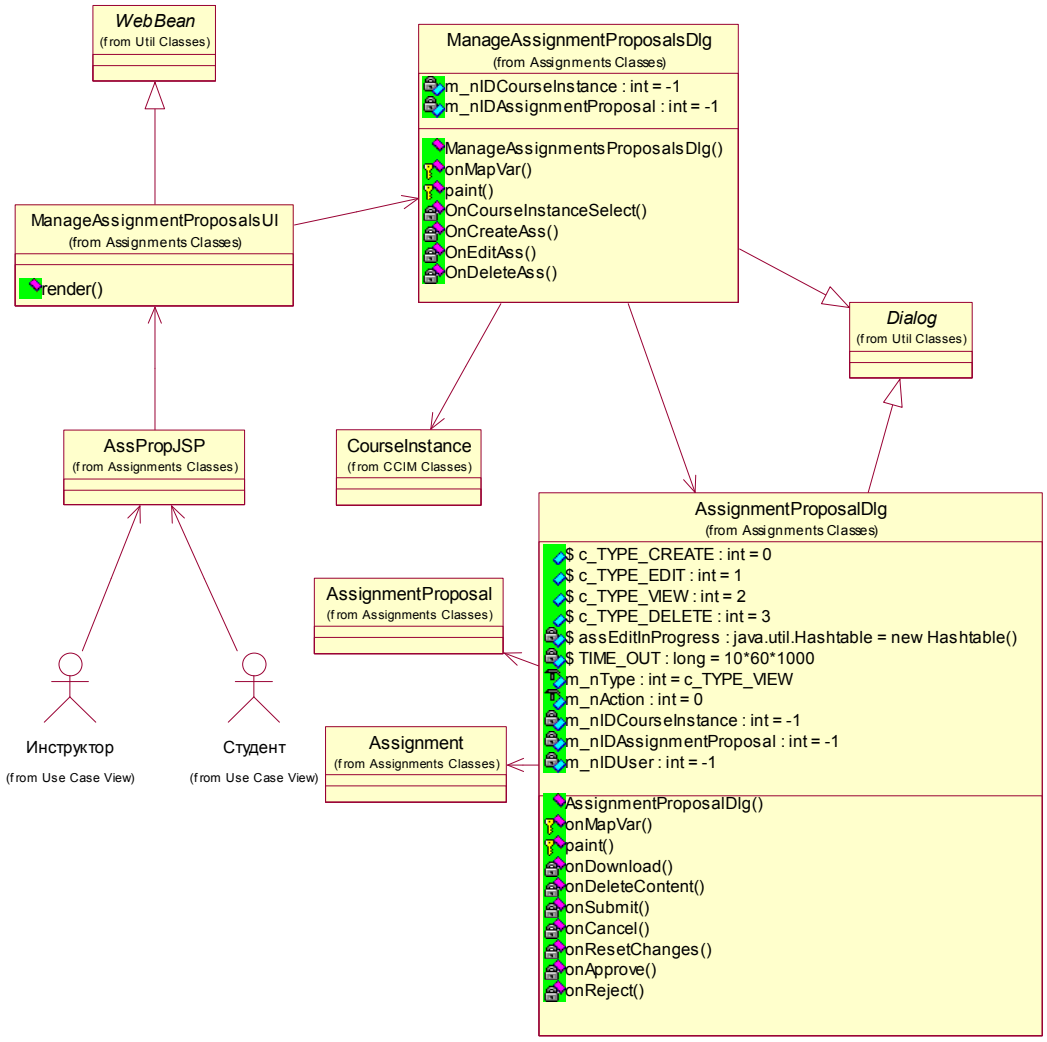
Диаграмата на последователностите за Инструктора е аналогична на тази за Студента.

Поток на събитията: Обектът Студент/Инструктор комуникира със системата чрез обекта AssPropJSP. Най-напред Студентът/Инструкторът зарежда страницата за управление на предложения за задания. Обектът ManageAssignmentProposalsUI създава диалога ManageAssignmentProposalsDlg, който с помощта на метода paint() генерира необходимия потребителски интерфейс. Студентът/Инструкторът избира курса, за който се отнасят предложенията за задания. Методът paint() на обекта ManageAssignmentProposalsDlg съдържа бизнес логиката, необходима за обработката на избрания курс. За избрания курс Студентът има възможност да избере между три действия – създаване, редактиране и

изтриване на предложение за задание. Инструкторът има възможност само за редактиране на предложение за задание, което включва одобрение и отказ на предложението.

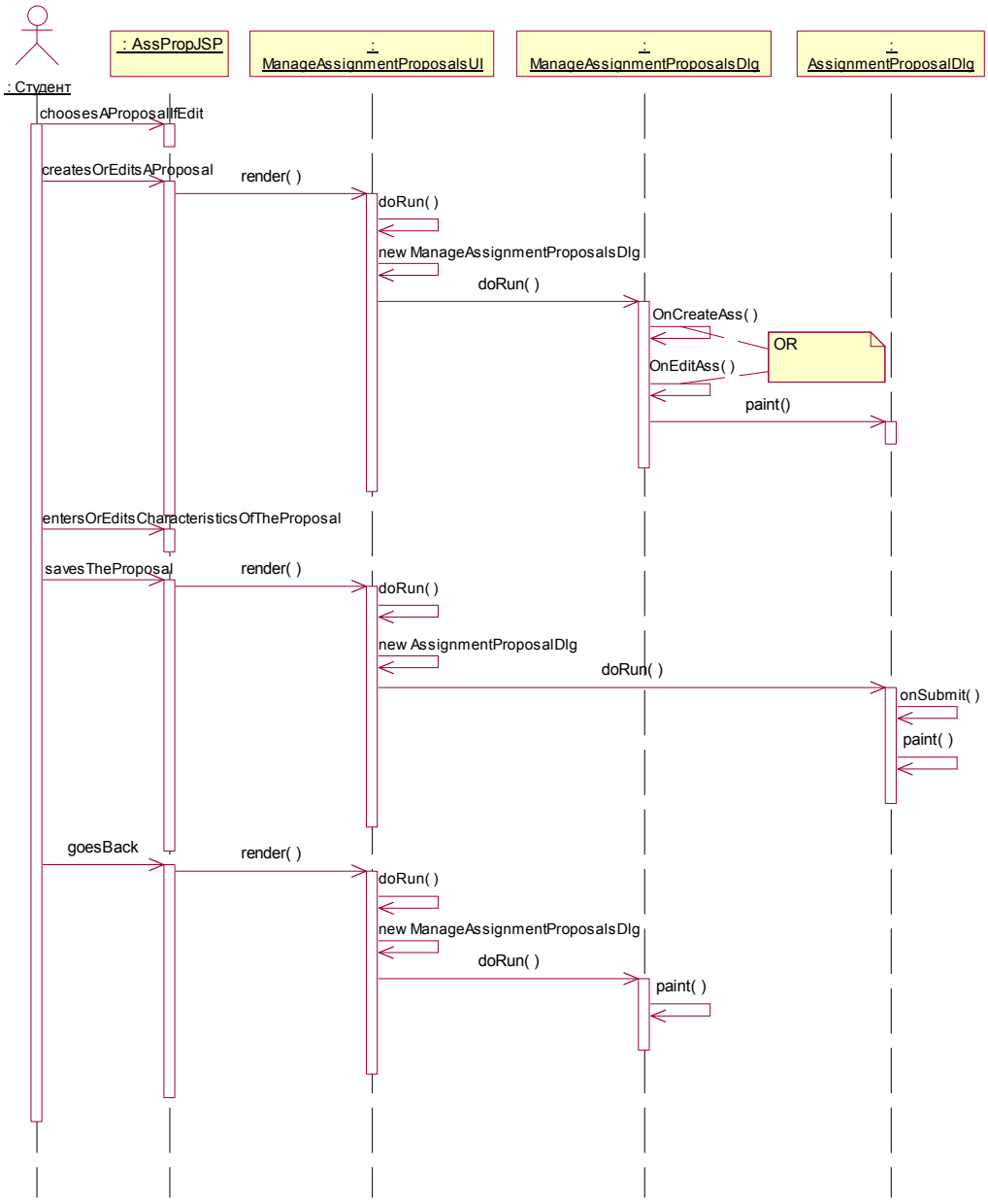
4.3.3.7. Реализация на случаи на употреба **Създаване/Редактиране/Изтриване на предложение за задание**:

Фигура 4.3.3.7.1 изобразява диаграмата на класовете:



Фигура 4.3.3.7.1. Диаграма на класовете за случаи на употреба Създаване/Редактиране/Изтриване на предложение за задание

Диаграмата на последователностите за създаване и редактиране на предложение за задание от студент е изобразена на фигура 4.3.3.7.2:

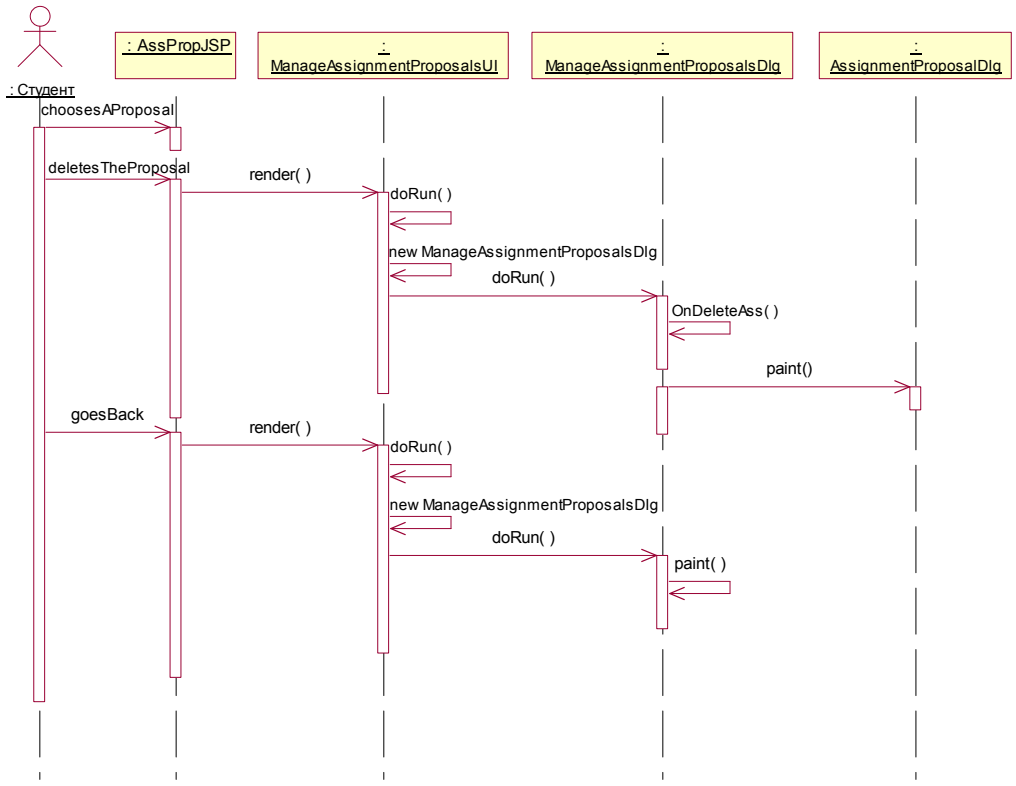


Фигура 4.3.3.7.2. Диаграма на последователностите за случаи на употреба

Създаване/Редактиране на предложение за задание

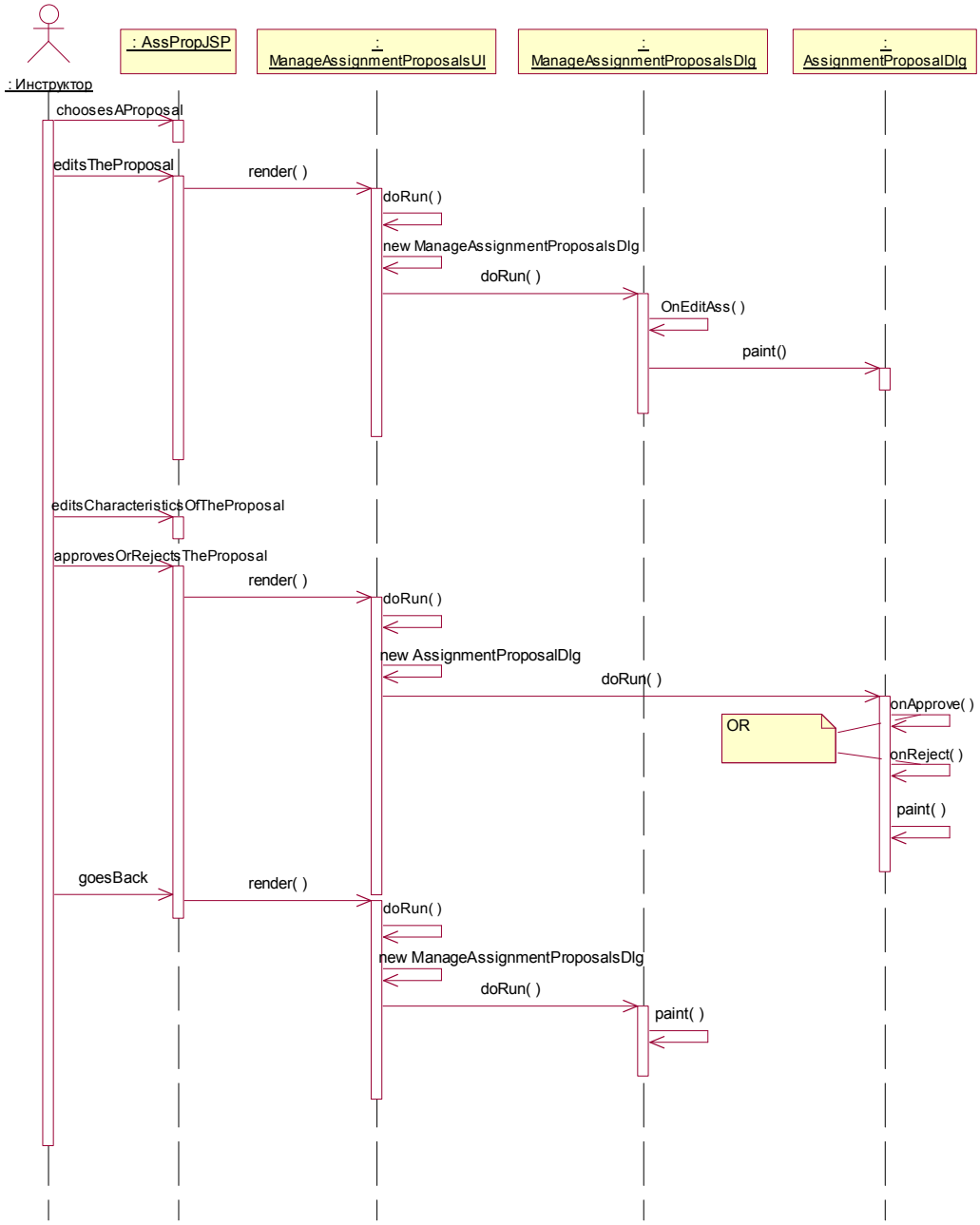
Диаграмата на последователностите за редактиране на предложение за задание от инструктора е аналогична на тази за студента.

Диаграмата на последователностите за изтриване на предложение за задание от студент е изобразена на фигура 4.3.3.7.3:



Фигура 4.3.3.7.3. Диаграма на последователностите за случай на употреба Изтриване на предложение за задание

Диаграмата на последователностите за одобрение и отказ на предложение за задание от инструктора е изобразена на фигура 4.3.3.7.4:



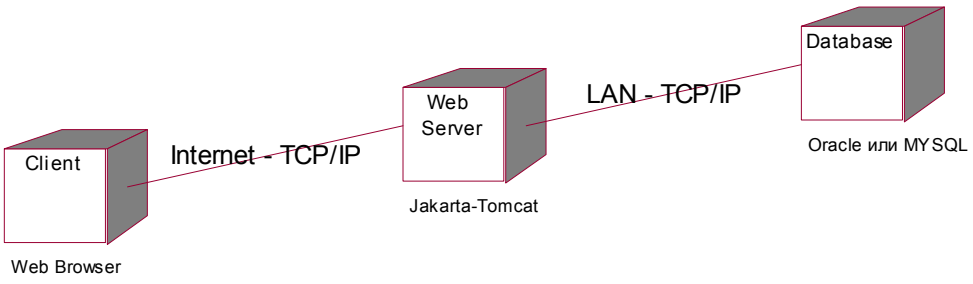
Фигура 4.3.3.7.4. Диаграма на последователностите за одобрение и отказ на предложение за задание

Поток на събитията: Обектът Студент/Инструктор комуникира със системата чрез обекта AssPropJSP. Той е заредил страницата за управление на предложения за задания и е избрал курса, за който се отнасят предложенията. За случаите на редактиране и изтриване на

предложение Студентът/Инструкторът избира предложението, за което ще се отнася съответното действие. След това Студентът/Инструкторът избира действието (Инструкторът има възможност само да редактира предложение за задание). Обектът ManageAssignmentProposalsUI създава диалога ManageAssignmentProposalsDlg, който от своя страна създава обект от тип AssignmentProposalDlg, чийто метод paint() генерира необходимия потребителски интерфейс. Той съдържа също и бизнес логиката, необходима за създаването, редактирането, изтриването, одобрението и отказа на предложение за задание.

4.3.4. Модел на внедряването

В етапа на дизайна се разработва и така наречения модел на внедряването. Това представлява обектен модел, който описва физическото разпределение на системата в термините на начина, по който функционалността е разпределена между компютърните възли. Всеки възел представлява изчислителен ресурс, в най-честия случай процесор или подобно хардуерно устройство. Изобразени са и връзките между възлите, по които се осъществява комуникацията. Диаграмата на модела на внедряването на системата ARCADE е показана на фигура 4.3.4 по-долу:



Фигура 4.3.4. Диаграма на модела на внедряването на системата ARCADE

Потребителят работи със системата посредством Интернет браузър. Най-лявото кубче символизира неговия компютър. Следващият възел е на Web сървъра. Комуникацията между потребителския компютър и Web сървъра се осъществява посредством Интернет. За Web сървъра е избран Tomcat. Последният компонент е базата от данни. Той се разполага на трети отделен компютър, свързан с Web сървъра през локална мрежа.

Конкретната комуникация между бизнес логиката на приложението и базата от данни става с широко разпространения стандарт JDBC (Java Database Connectivity). ARCADE поддържа версии за работа с две системи за управление на база данни – Oracle и MYSQL. За настоящата версия на платформата ARCADE сървърите на Tomcat и Oracle/MYSQL са на един физически компютър. Въпреки че тези сървъри работят на един компютър, връзката между тях не се променя – двата сървъра използват мрежовия протокол, мрежовите ресурси и настройки на компютъра, както и JDBC. Без необходимост от допълнителни настройки, двата сървъра могат да се разделят на самостоятелни компютри, като това не би нарушило работата на системата, нито модела на внедряването на цялото приложение.

4.4. Имплементация

В етапа на имплементацията започваме с резултата от дизайна и имплементираме системата в термините на компоненти, това са програмен код, скриптове, двоични (binaries) и изпълними файлове и др. По-голямата част от архитектурата на системата е определена по времето на дизайна. Най-основната цел на имплементацията е да се попълни архитектурата и системата като цяло. На този етап дизайн класовете и подсистемите, участващи в дизайна, се имплементират като файлови компоненти, съдържащи програмен код.

4.4.1. Модел на имплементацията

Моделът на имплементацията описва как елементите от модела на дизайна, дизайн класовете, са имплементирани в термините на компоненти, като файлове с програмен код, изпълними файлове и др. Моделът на имплементацията описва също и как компонентите са организирани съгласно механизмите за структуриране и разделяне на отделни модули, налични в средата за имплементация и използвания програмен език, и как компонентите зависят един от друг.

4.4.2. Компонент

Компонентът представлява физическо пакетиране на елементи от модела, като например дизайн класове от модела на дизайна. Някои стандартни стереотипи на компоненти включват следните:

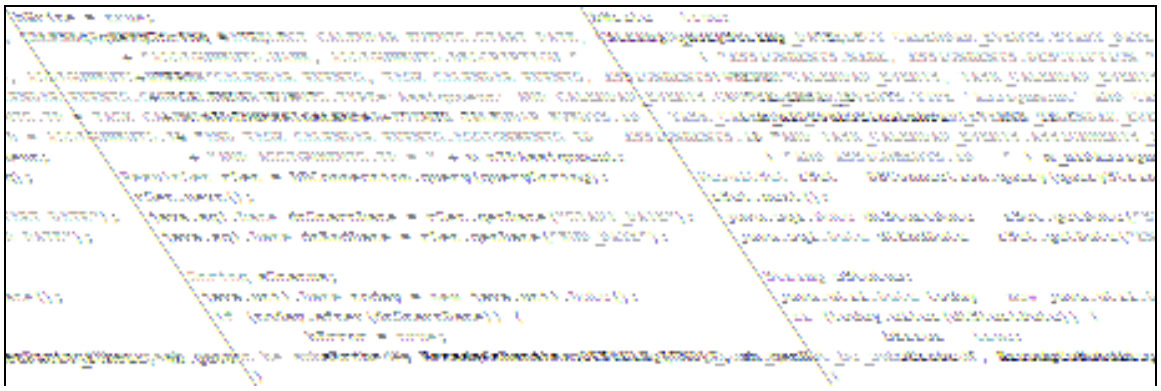
- ✓ изпълним файл – програма, която може да бъде стартирана от определено място
- ✓ файл – файл, съдържащ програмен код или данни
- ✓ библиотека – статична или динамична библиотека
- ✓ таблица – таблица от база от данни
- ✓ документ – документ

Компонентите са свързани с елементите от модела, които те имплементират. Често се случва един компонент да имплементира няколко елемента, като например дизайн класове. Това зависи до голяма степен от начина, по който са организирани файловете с програмен код на съответния програмен език, който се използва. Компонентите предоставят същите интерфейси като елементите от модела, които те имплементират.

В разработваната дипломна работа компонентите са 4 вида: java класове, компилирани class файлове, jsp страници и таблици в базата данни. Java класовете са следните:

- ✓ ManageAssignmentsStudentChoosingUI.java
- ✓ GiveAssignmentStudentSideDlg.java

Чрез тези класове е реализирана функционалността за разпределяне на заданията отдолу нагоре, възможността студентите сами да избират заданията и да се записват за тях. Фигура 4.4.2.1 изобразява фрагмент от кода, реализиращ записването на студент за задание.



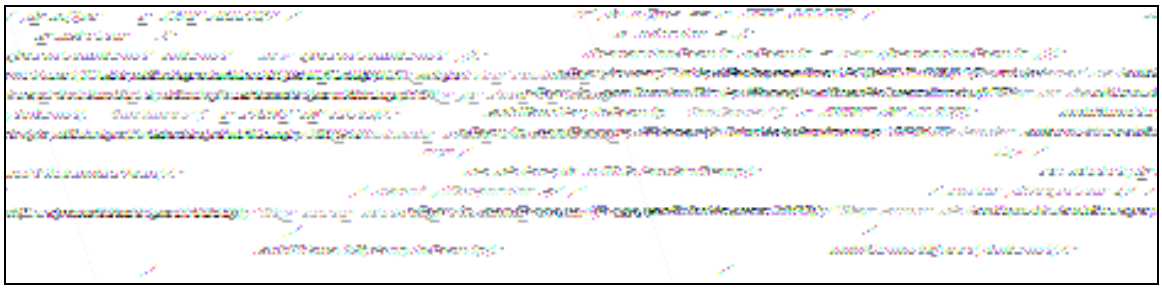
фигура 4.4.2.1. Фрагмент от кода, реализиращ записването на студент за задание

- ✓ CalendarEvent.java
- ✓ CalendarEventDlg.java
- ✓ ManageCalendarEventsUI.java
- ✓ ManageCalendarEventsDlg.java

Тези класове реализират управлението на календарните събития, създаване, редактиране и изтриване. Фигура 4.4.2.2 изобразява фрагмент от кода, реализиращ създаването и редактирането на календарно събитие.

Фигура 4.4.2.2. Фрагмент от кода, реализиращ създаване и редактиране на календ. събитие

Фигура 4.4.2.3 изобразява фрагмент от кода, реализиращ изтриването на календарно събитие.



Фигура 4.4.2.3. Фрагмент от кода, реализиращ изтриването на календарно събитие

Класовете, реализиращи определянето на събитие за задание са:

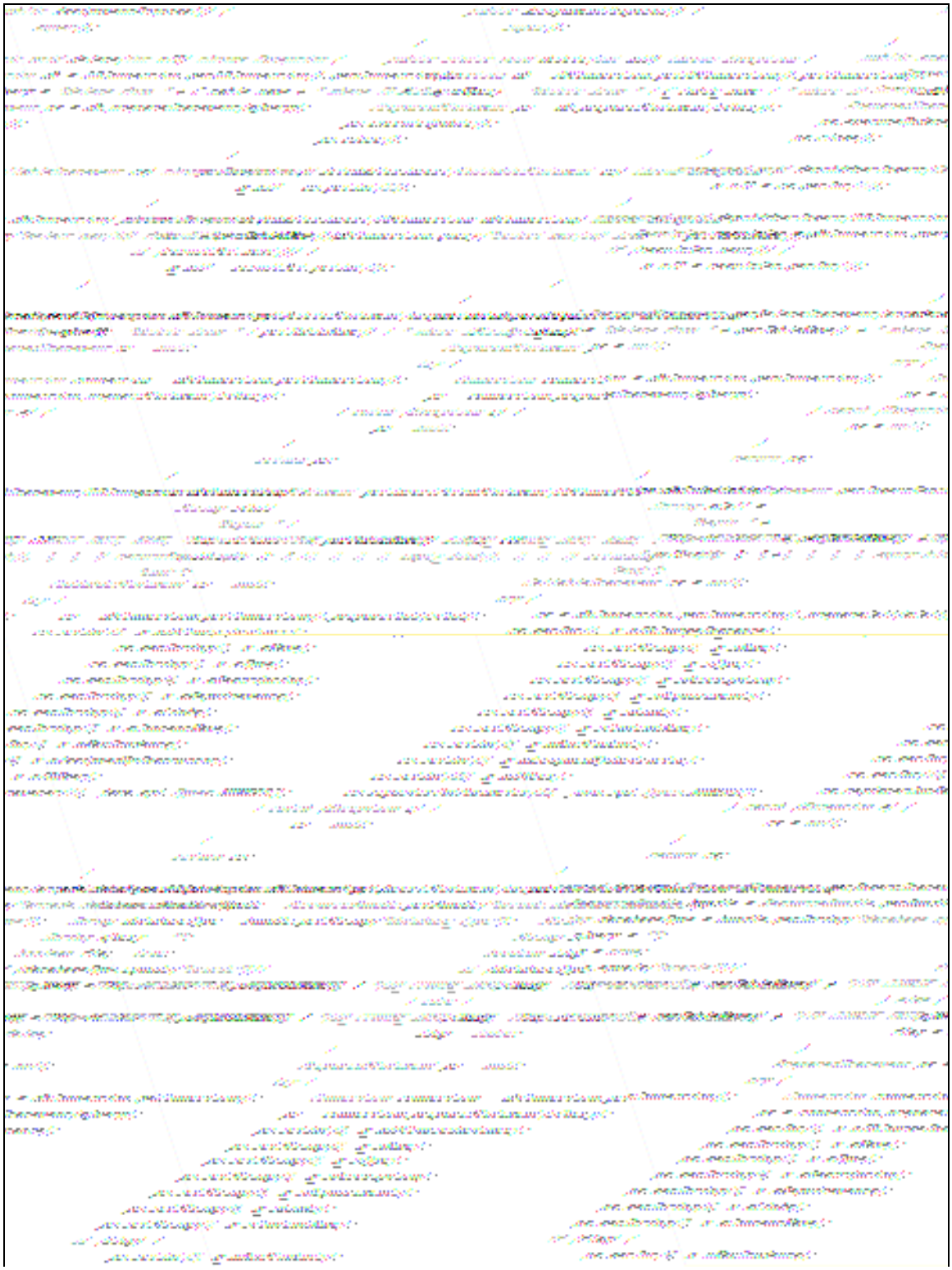
- ✓ SetEventDlg.java
- ✓ AssignmentCalendarEvent.java

Класовете, реализиращи управлението на предложения за задания са:

- ✓ AssignmentProposal.java
- ✓ AssignmentProposalDlg.java
- ✓ ManageAssignmentsProposalsUI.java

- ✓ ManageAssignmentsProposalsDlg.java

Фигура 4.4.2.4 изобразява фрагмент от кода, реализиращ създаване, редактиране и изтриване на предложения за задания.



Фигура 4.4.2.4. Фрагмент от кода, реализиращ създаване, редактиране и изтриване на предложения за задания

Всеки java клас се компилира до съответен class файл. Това са

- ✓ ManageAssignmentsProposalsUI.class
- ✓ GiveAssignmentStudentSideDlg.class
- ✓ CalendarEvent.class
- ✓ CalendarEventDlg.class
- ✓ ManageCalendarEventsUI.class
- ✓ ManageCalendarEventsDlg.class
- ✓ SetEventDlg.class
- ✓ AssignmentCalendarEvent.class
- ✓ AssignmentProposal.class
- ✓ AssignmentProposalDlg.class
- ✓ ManageAssignmentsStudentChoosingUI.class
- ✓ ManageAssignmentsProposalsDlg.class

Class файловете са изпълними файлове.

JSP страниците в разработката са следните:

- ✓ Assignments3.jsp
- ✓ Assignments4.jsp
- ✓ Assignments5.jsp
- ✓ Assignments6.jsp
- ✓ AssignmentsAll.jsp

Пълният текст на java файловете и jsp страниците може да бъде видян на компакт диска, приложен към дипломната работа.

Използваните таблици в разработваната дипломна работа са:

- ✓ assignments – съдържа информация за съществуващите задания в системата
- ✓ assignments_proposals – съдържа информация за предложените от студентите задания
- ✓ calendar_events – съдържа информация за календарните събития в системата

- ✓ task_calendar_events – съдържа информация за това кое задание към кое календарно събитие се отнася

Таблицата assignments съдържа 11 полета:

- id – идентификатор на записа, числово поле
- id_banks – идентификатор на банката, към която е заданието, числово поле
- name – име на заданието, текстово поле
- type – тип на заданието, текстово поле
- description – описание на заданието, масив от байтове
- requirements – изисквания на заданието, масив от байтове
- links – връзки, които могат да се ползват за заданието, масив от байтове
- content_name – текстово поле
- content – съдържанието на заданието, масив от байтове
- max_students – максимален брой студенти, които могат да участват в това задание, числово поле
- assigned_by_instructor – това поле съдържа информация дали заданието се задава от инструктора или студентът може сам да си го избира, битово поле

Таблицата assignments не е нова за системата ARCADE, нови са само полетата max_students и assigned_by_instructor. Скриптът, който добавя тези нови полета в таблицата, е следният:

ALTER TABLE assignments ADD COLUMN MAX_STUDENTS TINYINT(3) UNSIGNED 1, ADD COLUMN ASSIGNED_BY_INSTRUCTOR TINYINT(1) 0

Таблицата assignments_proposals съдържа 12 полета:

- id – идентификатор на записа, числово поле
- id_course_inst – идентификатор на курса, към който се отнася предложението за задание
- name – име на предложението за задание, текстово поле
- type – тип на предложението за задание, текстово поле

- description – описание на предложението за задание, масив от байтове
- requirements – изисквания на предложението за задание, масив от байтове
- links – връзки, които могат да се ползват за предложението за задание, масив от байтове
- content_name – текстово поле
- content – съдържанието на предложението за задание, масив от байтове
- max_students – максимален брой студенти, които могат да участват в това задание, числово поле
- assigned_by_instructor – това поле съдържа информация дали заданието ще се задава от инструктора или студентът може сам да си го избира, битово поле
- id_user – идентификатор на потребителя, предложил заданието

Скриптът, който създава таблицата, е следният:

CREATE TABLE assignments_proposals (ID INT(11) NOT NULL AUTOINCREMENT PRIMARY KEY, ID_COURSE_INST INT(11) NOT NULL, NAME VARCHAR(100), TYPE VARCHAR(100), DESCRIPTION BLOB, REQUIREMENTS BLOB, LINKS BLOB, CONTENT_NAME VARCHAR(255), CONTENT BLOB, MAX_STUDENTS TINYINT(3) UNSIGNED 1, ASSIGNED_BY_INSTRUCTOR TINYINT(1) 0, ID_USER INT(11) 0)

Таблицата calendar_events съдържа следните полета:

- id – идентификатор на записа, числово поле
- type – тип на събитието, текстово поле
- name – име на събитието, текстово поле
- descr – описание на събитието, текстово поле
- course_inst_id – идентификатор на курса, за който се отнася събитието, числово поле
- start_date – начална дата на събитието, поле от тип дата
- end_date – крайна дата на събитието, поле от тип дата
- status – статус на събитието, текстово поле
- points – брой точки, които носи събитието, числово поле

- controlstyle – това поле съдържа информация дали събитието е задължително, битово поле

Скриптът за създаване на таблицата calendar_events е следният:

CREATE TABLE calendar_events (ID INT(11) NOT NULL AUTOINCREMENT PRIMARY KEY, TYPE VARCHAR(100), NAME VARCHAR(100), DESCR VARCHAR(100), COURSE_INST_ID INT(11), START_DATE DATE, END_DATE DATE, STATUS VARCHAR(50), POINTS INT(10) UNSIGNED, CONTROLSTYLE TINYINT(1))

Таблицата task_calendar_events съдържа следните 3 полета:

- id – идентификатор на записа, числово поле
- calendar_events_id – идентификатор на календарното събитие
- assignments_id – идентификатор на заданието

Скриптът за създаване на таблицата task_calendar_events е следният:

CREATE TABLE task_calendar_events (ID INT(11) NOT NULL AUTOINCREMENT PRIMARY KEY, CALENDAR_EVENTS_ID INT(11) UNSIGNED, ASSGNMENTS_ID INT(11) UNSIGNED)

4.4.3. Имплементационна подсистема

Имплементационната подсистема предоставя средство за организиране на артефактите на модела на имплементацията. Една такава подсистема се състои от компоненти, интерфейси и други подсистеми. Пример за имплементационна подсистема в Java е пакетът (package), за Visual Basic това е проект (project). За текущата разработка имплементационната подсистема е пакетът Assignments.

4.5. Тестване

Тестването е последният етап в разработването на продукта. В процеса на тестването се проверява резултатът от имплементацията. Целите на тестването са следните: планиране на необходимите тестове; проектиране и имплементиране на тестовете чрез създаване на тестови случаи, които указват какво ще се тества, тестови процедури, които указват как да бъдат изпълнени тестовете, и изпълними тестови компоненти за автоматизиране на тестовете при възможност; изпълнение на различните тестове и обработка на резултатите.

4.5.1. Тестов модел

Тестовият модел е основният резултат от тестването. Той описва как изпълнимите компоненти от имплементацията трябва да бъдат тествани. Тестовият модел представлява съвкупност от тестови случаи, тестови процедури и тестови компоненти.

4.5.2. Тестов случай

Тестовият случай указва един конкретен пример за тестване на системата, като включва какво да се тества, с какви входни параметри, какви резултати трябва да се получат и под какви условия. Най-често това, което се тества, е някакво изискване на системата или съвкупност от изисквания. Тестовите случаи най-общо са два вида: тестови случаи, които указват как да се тества случай на употреба, и тестови случаи, които указват как да се тества реализация на случай на употреба в дизайна на системата. Няколко тестови случаи могат да бъдат подобни и да се различават само по входен параметър или резултат. Това може да се случи при тестване на различни сценарии на един и същи случай на употреба.

4.5.2.1. Тестов случай **Записване за задание**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming
- ✓ Име на събитие: Assignment 1
- ✓ Име на задача: Chat Application

Резултат:

- ✓ Студентът е записан за задачата с име Chat Application

Условия:

- ✓ Текущата дата трябва да е преди крайната дата за записване
- ✓ Броят на записалите се до момента студенти за съответното задание трябва да е по-малък от определения брой студенти за него
- ✓ Студентът не трябва да е записан за друго задание от същото календарно събитие

4.5.2.2. Тестов случай **Отписване от задание**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming
- ✓ Име на задача: Chat Application

Резултат:

- ✓ Студентът вече не е записан за задачата Chat Application

Условия:

- ✓ Заданието трябва да бъде от тип за избор от студента, задания, разпределени от инструктора (с тип на разпределение отгоре-надолу) не могат да бъдат отписвани от студентите)
- ✓ Текущата дата трябва да е преди крайната дата за записване за това задание

4.5.2.3. Тестов случай **Управление на календарни събития 1**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming

Резултат:

- ✓ Системата показва екрана за създаване на календарно събитие

4.5.2.4. Тестов случай **Управление на календарни събития 2**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming
- ✓ Име на събитие: Assignment 1

Резултат:

- ✓ Системата показва екрана за редактиране на събитието Assignment 1

4.5.2.5. Тестов случай **Създаване на календарно събитие**:

Входни данни:

- ✓ Тип: Assignment
- ✓ Наименование: Assignment 1
- ✓ Описание: Java приложение
- ✓ Начална дата: 2005-05-31
- ✓ Крайна дата: 2005-07-31
- ✓ Статус:
- ✓ Точки: 50
- ✓ Задължителен: 1 - Да

Резултат:

- ✓ Създадено е календарно събитие с име Assignment 1

Условия:

- ✓ Задължителните характеристики (тип, начална и крайна дата) за календарно събитие трябва да бъдат въведени
- ✓ Датите трябва да бъдат въведени във формат 'гггг-мм-дд'

4.5.2.6. Тестов случай **Редактиране на календарно събитие**:

Входни данни:

- ✓ Тип: Assignment
- ✓ Наименование: Assignment 1
- ✓ Описание: Java приложение
- ✓ Начална дата: 2005-04-30
- ✓ Крайна дата: 2005-07-31
- ✓ Статус:
- ✓ Точки: 100
- ✓ Задължителен: 1 - Да

Резултат:

- ✓ Календарното събитие с име Assignment 1 е редактирано

Условия:

- ✓ Задължителните характеристики (тип, начална и крайна дата) за календарно събитие трябва да бъдат въведени
- ✓ Датите трябва да са във формат 'гггг-мм-дд'

4.5.2.7. Тестов случай **Изтриване на календарно събитие**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming
- ✓ Име на събитие: Assignment 2

Резултат:

- ✓ Календарното събитие Assignment 2 е изтрито от системата

4.5.2.8. Тестов случай **Управление на задания**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming

- ✓ Име на банка със задачи за самостоятелна работа: Java
- ✓ Име на задачата за домашна работа: FTP Client Application

Резултат:

- ✓ Системата показва екрана за определяне към кое календарно събитие ще бъде задачата с име FTP Client Application

4.5.2.9. Тестов случай **Определяне на събитие за задание**:

Входни данни:

- ✓ Име на задача: FTP Client Application
- ✓ Име на календарно събитие: Assignment 1

Резултат:

- ✓ Задачата FTP Client Application принадлежи на календарното събитие Assignment 1

4.5.2.10. Тестов случай **Управление на предложения за задания 1**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming

Резултат:

- ✓ Системата показва екрана за създаване на предложение за задание

4.5.2.11. Тестов случай **Управление на предложения за задания 2**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming
- ✓ Име на задачата за домашна работа: Chat Application

Резултат:

- ✓ Системата показва екрана за редактиране на предложението с име Chat Application

4.5.2.12. Тестов случай **Създаване на предложение за задание**:

Входни данни:

- ✓ Наименование: Chat Application
- ✓ Тип: Project
- ✓ Описание: Приложение за чат в реално време
- ✓ Изисквания: Java
- ✓ Връзки към източници на информация: Интернет

- ✓ Съдържание:
- ✓ Максимален брой студенти за тази задача: 3
- ✓ Начин на задаване: 2 – Не се задава от инструктора

Резултат:

- ✓ Създадено е предложение за задание с име Chat Application

Условия:

- ✓ Задължителните характеристики за предложението за задание трябва да бъдат въведени

4.5.2.13. Тестов случай **Редактиране на предложение за задание**:

Входни данни:

- ✓ Наименование: Chat Application
- ✓ Тип: Project
- ✓ Описание: Приложение за чат в реално време
- ✓ Изисквания: Java
- ✓ Връзки към източници на информация: www
- ✓ Съдържание:
- ✓ Максимален брой студенти за тази задача: 2
- ✓ Начин на задаване: 2 – Не се задава от инструктора

Резултат:

- ✓ Предложението за задание с име Chat Application е редактирано

Условия:

- ✓ Задължителните характеристики за предложението за задание трябва да бъдат въведени

4.5.2.14. Тестов случай **Изтриване на предложение за задание**:

Входни данни:

- ✓ Име на инстанция на курс: Java Programming
- ✓ Име на задача за домашна работа: SMTP Application

Резултат:

- ✓ Предложението за задание SMTP Application е изтрито от системата

4.5.2.15. Тестов случай **Одобрение на предложение за задание**:

Входни данни:

- ✓ Наименование: Chat Application
- ✓ Тип: Project
- ✓ Описание: Приложение за чат в реално време
- ✓ Изисквания: Java
- ✓ Връзки към източници на информация: www
- ✓ Съдържание:
- ✓ Максимален брой студенти за тази задача: 2
- ✓ Начин на задаване: 2 – Не се задава от инструктора

Резултат:

- ✓ Създадена е задача с име Chat Application
- ✓ Предложението за задание Chat Application е изтрито от системата

4.5.2.16. Тестов случай **Отказ на предложение за задание**:

Входни данни:

- ✓ Наименование: Another Chat Application
- ✓ Тип: Project
- ✓ Описание: Приложение за чат в реално време
- ✓ Изисквания: Java
- ✓ Връзки към източници на информация: www
- ✓ Съдържание:
- ✓ Максимален брой студенти за тази задача: 2
- ✓ Начин на задаване: 2 – Не се задава от инструктора

Резултат:

- ✓ Предложението за задание Another Chat Application е изтрито от системата

4.5.3. Тестова процедура

Тестовата процедура указва как да бъдат изпълнени един или няколко тестови случаи или части от тях. Най-често представлява последователност от действия със системата, които трябва да бъдат извършени за изпълнението на тестовия или тестовите случаи. Една тестова процедура може да специфицира няколко тестови случаи, както и един тестов случай може да бъде описан в няколко тестови процедури.

Тестовата процедура може да бъде подобна на потока на събитията на случаите на употреба, но тя съдържа и допълнителна информация – какви входни данни да бъдат използвани (от тестовия случай), как да бъдат подадени тези данни с потребителския интерфейс и какво да се проверява.

4.5.3.1. Тестова процедура **Записване за задание**:

1) Влезте в системата като *Студент* и отворете менюто *Студент*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Избор на задание*
4) Въведете данните от тестовия случай *Записване за задание* и натиснете бутона *Записване за тази задача*
5) Сравнете резултата с този в тестовия случай

4.5.3.2. Тестова процедура **Отписване от задание**:

1) Влезте в системата като *Студент* и отворете менюто *Студент*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Моите задания*
4) Въведете данните от тестовия случай *Отписване от задание* и натиснете бутона *Отписване от тази задача*
5) Сравнете резултата с този в тестовия случай

4.5.3.3. Тестова процедура **Създаване на календарно събитие**:

1) Влезте в системата като *Инструктор* и отворете менюто *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Създаване / Управление на събития*
4) Въведете данните от тестовия случай *Управление на календарни събития 1* и натиснете бутона *Създай*
5) Сравнете резултата с този в тестовия случай
6) Въведете данните от тестовия случай *Създаване на календарно събитие* и натиснете бутона *Въвеждане*

7) Сравнете резултата с този в тестовия случай

4.5.3.4. Тестова процедура **Редактиране на календарно събитие**:

1) Влезте в системата като *Инструктор* и отворете менюто *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Създаване / Управление на събития*
4) Въведете данните от тестовия случай *Управление на календарни събития 2* и натиснете бутона *Редактирай*
5) Сравнете резултата с този в тестовия случай
6) Въведете данните от тестовия случай *Редактиране на календарно събитие* и натиснете бутона *Въвеждане*
7) Сравнете резултата с този в тестовия случай

4.5.3.5. Тестова процедура **Изтриване на календарно събитие**:

1) Влезте в системата като *Инструктор* и отворете менюто *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Създаване / Управление на събития*
4) Въведете данните от тестовия случай *Изтриване на календарно събитие* и натиснете бутона *Изтрий*
5) Сравнете резултата с този в тестовия случай

4.5.3.6. Тестова процедура **Определяне на събитие за задание**:

1) Влезте в системата като *Инструктор* и отворете менюто *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Управление на задания*
4) Въведете данните от тестовия случай *Управление на задания* и натиснете бутона *Определяне на събитието*
5) Сравнете резултата с този в тестовия случай

6) Въведете данните от тестовия случай *Определяне на събитие за задание* и натиснете бутона *Определяне на събитието*
7) Сравнете резултата с този в тестовия случай

4.5.3.7. Тестова процедура **Създаване на предложение за задание**:

1) Влезте в системата като *Студент* и отворете менюто *Студент*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Предложения за задания*
4) Въведете данните от тестовия случай *Управление на предложения за задания 1* и натиснете бутона *Създай*
5) Сравнете резултата с този в тестовия случай
6) Въведете данните от тестовия случай *Създаване на предложение за задание* и натиснете бутона *Въвеждане*
7) Сравнете резултата с този в тестовия случай

4.5.3.8. Тестова процедура **Редактиране на предложение за задание**:

1) Влезте в системата като *Студент* или *Инструктор* и отворете съответно менюто *Студент* или *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Предложения за задания*
4) Въведете данните от тестовия случай *Управление на предложения за задания 2* и натиснете бутона *Редактирай*
5) Сравнете резултата с този в тестовия случай
6) Въведете данните от тестовия случай *Редактиране на предложение за задание* и натиснете бутона *Въвеждане*
7) Сравнете резултата с този в тестовия случай

4.5.3.9. Тестова процедура **Изтриване на предложение за задание**:

1) Влезте в системата като *Студент* и отворете менюто *Студент*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Предложения за задания*

4) Въведете данните от тестовия случай *Изтриване на предложение за задание* и натиснете бутона *Изтрий*
5) Сравнете резултата с този в тестовия случай

4.5.3.10. Тестова процедура **Одобрение на предложение за задание**

1) Влезте в системата като *Инструктор* и отворете менюто *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Предложения за задания*
4) Въведете данните от тестовия случай *Управление на предложения за задания 2* и натиснете бутона *Редактирай*
5) Сравнете резултата с този в тестовия случай
6) Въведете данните от тестовия случай *Одобрение на предложение за задание* и натиснете бутона *Одобрение на задачата*
7) Сравнете резултата с този в тестовия случай

4.5.3.11. Тестова процедура **Отказ на предложение за задание**

1) Влезте в системата като *Инструктор* и отворете менюто *Преподавател*
2) Отидете на екрана със заданията, като изберете хипервръзката *Задания*
3) Изберете *Предложения за задания*
4) Въведете данните от тестовия случай *Управление на предложения за задания 2* и натиснете бутона *Редактирай*
5) Сравнете резултата с този в тестовия случай
6) Въведете данните от тестовия случай *Отказ на предложение за задание* и натиснете бутона *Отказ на задачата*
7) Сравнете резултата с този в тестовия случай

4.5.4. Тестов компонент

Тестовият компонент автоматизира една или няколко тестови процедури или части от тях. Такива компоненти могат да бъдат разработени с помощта на програмен или скриптов език, може да се използва също и

обектният модел. Тестовите компоненти се използват за проверка на компонентите от модела на имплементацията. Предоставят се тестови входни данни, следи се и се контролира изпълнението на тествания компонент и се отчитат резултатите от теста.

4.5.5. Тестов план

Тестовият план описва начина на провеждане на тестовете, реда на тяхното провеждане, необходими ресурси. Най-общо в този документ се описва кои тестови процедури ще се извършват в каква последователност.

4.5.6. Дефекти

Откритите дефекти заедно с тяхната важност също се документират. Отстраняването на грешките започва от най-важните. Често системата може да влезне в експлоатация без да бъдат коригирани абсолютно всички открити грешки. Некоригираните грешки обаче остават документирани за по-нататъшно коригиране.

4.5.7. Оценка на тестовете

Този документ представлява оценка на резултатите от проведените тестове. В него се описва състоянието на откритите дефекти. Дава се заключение дали системата е готова за представяне.

5. Заключение

Интернет технологиите се развиват изключително бързо. Те навлизат във всяка сфера на обществения и социалния живот, включително и в сферата на образованието. Обект на разглеждане на предложената дипломна работа е платформата за дистанционно обучение ARCADE и по-специално модулът за задания. За сравнение са разгледани и други съществуващи системи за дистанционно обучение, като акцентът на разглеждане е върху възможностите, които те предлагат във връзка с курсовите задания. В дипломната работа са разработени няколко допълнения и подобрения към модула за заданията на работещата към момента система в Софийския Университет. Специално внимание е обърнато на унифицирания процес за разработване на софтуер [1], като са следвани неговите стъпки. Базирайки целия процес на моделирането с UML, разширението и редактирането на продукта значително се улеснява. На базата на потребителските изисквания са изведени случаи на употреба по методологията на UML. Така дефинираният модел е имплементиран и новите допълнения са напълно разработени и функциониращи.

По време на разработката на настоящата работа възникнаха нови идеи, които могат да бъдат реализирани в бъдещи версии на платформата ARCADE. Относно предложенията за задания за момента след одобрението или отказа на предложението от инструктора не се запомня кой е дал предложението, т.е. не се пази история за това кои студенти са предлагали задания, както и кои задания са дадени като предложения от студентите. Добре би било тази информация да може да бъде получена при желание на инструктора. Предложенията за задания са един вид активно участие в процеса на обучение и могат да бъдат вземани под внимание при окончателното оценяване на студентите. Друга полезна нова функционалност в модула за задания би била системата да може да предоставя на инструкторите обобщен поглед на всички задания за даден курс, който той води, както и сравнителна информация за представянето на студентите на отделните задания.

6. Използвана литература

[1] Ivar Jacobson, Grady Booch, James Rumbaugh. The Unified Software Development Process. Addison-Wesley. 1999

[2] Martin Fowler, Kendall Scott. UML Distilled Second Edition A Brief Guide to the Standard Object Modeling Language. Addison-Wesley. 1999

[3] Платформа за дистанционно обучение WebCT, http://www.webct.com

[4] Платформа за дистанционно обучение WebAssign, http://www.webassign.net

[5] Платформа за дистанционно обучение Moodle, http://moodle.org

[6] Официалният сайт на Java, http://java.sun.com

[7] http://jcp.org/en/introduction/faq, http://jcp.org/ja/participation/committee

[8] Документацията на Tomcat, http://jakarta.apache.org/tomcat/

[9] MySQL - http://dev.mysql.com/, http://archives.postgresql.org/pgsql-sql/1999-08/msg00237.php

[10] Bontchev B., D. Vassileva. “Internet Authoring Tool for E-learning Courseware”. Proc. of WSEAS'2003 Int. Conference, Corfu Island, Greece, July 7-10, 2003

[11] Bontchev B., D. Mutafchieva. “Task Assignment and Control in ARCADE”, Proc. of ET'2003, Sozopol, Bulgaria, 24-26 Sept. 2003

7. Публикации по дипломната работа

Върху настоящата работа има направена публикация [11]. В нея се представят накратко архитектурата на ARCADE и потребителските роли и е обсъдена в детайли подсистемата за разпределението на заданията. Показани са начините за дефиниране на курсова задача от инструктора и разпределянето на определено задание на един студент или на група от студенти. Описани са и различните подходи на разпределение на задания, както и процеса на разработване и предаване на курсовите задания.