UML проект на редактор-генератор на структурни схеми

Искра Бонева, Полина Атанасова, Ирина Желязкова

UML project of an Editor-Generator of Structural Schemes: *The work is the first of two papers concerning an editor-generator of structural schemes with a script language. With small differences the tool can be used by the author and learner in the framework of a task-oriented environment for constructing different kinds of structural schemes.*

Key words: *User Interface, Author's Tool, Editing and Generating, Structural Schemes.*

ВЪВЕДЕНИЕ

Известни са усилията на много наши автори и колективи за създаване на Web-базирани системи за практическо обучение по различни дисциплини [3,4,5]. В [2] е представена и концепция за трислойна Web-базирана система за практическо обучение, независима от предметната област. Нейна алтернатива са задачно-ориентираните среди (ЗОС), разработвани от колектив в Русенския университет. Една ЗОС поддържа планирането и изпълнението на упражнения, съдържащи определен познавателен тип задачи със специализиран скриптов език. В [6] са представени проектът и реализацията на WINDOWS-базиран прототип на ЗОС за построяване на различни видове структурни схеми (СС).

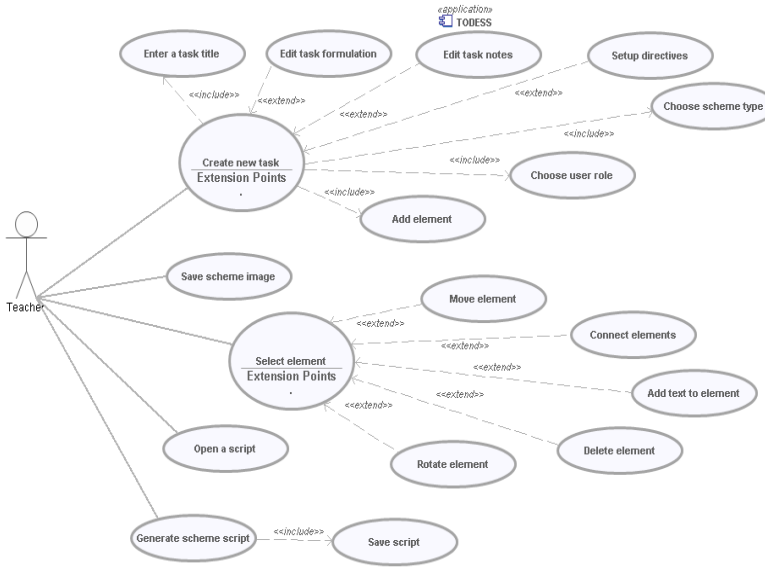
Настоящата работа фокусира върху UML проекта на подобрен JAVA вариант на авторското средство на тази среда, което по същество е редактор-генератор на програми (РГП) на скриптов език. Проектът е представен чрез три диаграми, а именно на: случаите на употреба, последователностите и дейностите [1]. Заключението обобщава постигнатите резултати и бъдещите намерения на авторите.

ДИАГРАМА НА СЛУЧАИТЕ НА УПОТРЕБА

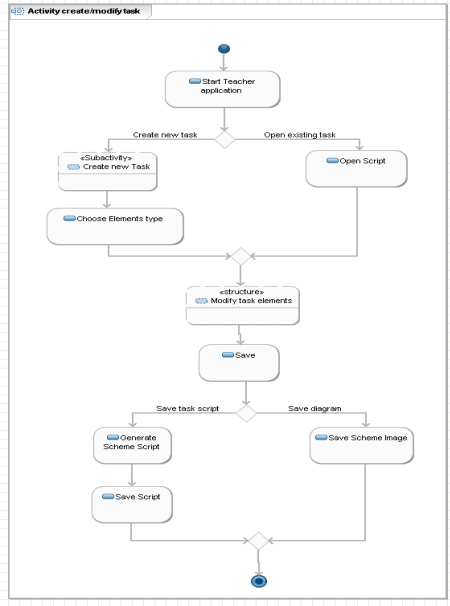
Диаграмата на случаите на употреба (ДСУ) служи за нагледно представяне на функционалните изисквания към РГП на автора (А), означен на фиг.1 с **Teacher**. С малки различия със същото средство ще работи и обучаваният (О), който на фигурата не е показан. С плътни линии са отразени директните връзки на потребителя (П) с основните случаи на употреба, т.е. подпомаганите от РГП потребителски функции. Те, от своя страна, са свързани с други функции чрез пунктирни и наименовани стрелки. Тези връзки между два случая са от тип <<**include**>> или <<**extend**>>. Връзката <<**include**>> показва, че функцията, от която излиза стрелката, зависи от функцията, към която тя е насочена. Връзката <<**extend**>> показва, че изпълнението на дадената функция не зависи от изпълнението на тази, с която е свързана. Основните случаи на употреба са: създаване на нова задача, запазване на построената схема като изображение, запазване на генерирания скрипт като текстов файл, отваряне на съществуващ скрипт и работа с елементите на схемата. За създаване на една задача е необходимо: да се избере потребителска роля (А или О), типът на елементите и да се въведе уникално име на задачата. След избор на елемент той може да се мести, завърта, изтрива, свързва с текст, както и с други елементи. При желание на П скриптът на текущата задача може да се прегенерира. След създаването й може да се променят: формулировката, обясненията и директивите на задачата.

ДИАГРАМА НА ДЕЙСТВИЯТА

Поради силната интерактивност и паралелизъм на работа, логиката на изпълняваните със средството действия тук е представена с няколко отделни диаграми на действията (ДД). Показаната на фиг. 2 диаграма касае създаване/модифициране на задача.



Фиг.1. ДСУ на средството от потребителя

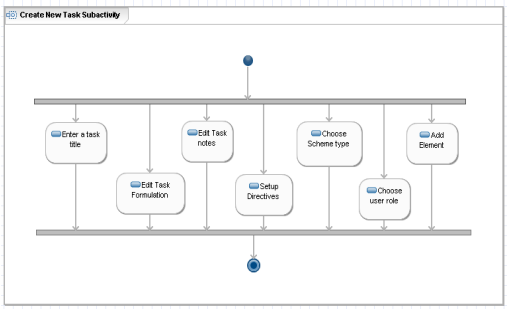


Фиг. 2. ДД за създаване/модифициране на задача

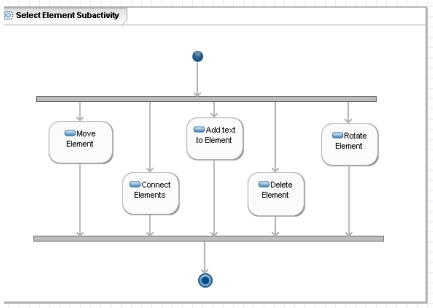
След стартиране на средството А може да създаде нова задача чрез няколко спомагателни диалогови прозорци, след което може да избере типа на елементите. При желание за редактиране на съществуваща задача, той може да зареди съответния скриптов файл за редактиране. След като задачата се инициализира, тя

може да бъде модифицирана. След приключване на редактирането П може да запази задачата в два различни файлови формата, т.е. като преизползваем скрипт и като растерно изображение. Ако П е приключил с текущата задача, той може да приключи с изпълнението на приложението.

На фиг. 3 е дадена поддиаграмата за създаване на нова схема. Показани са допустимите действия за създаване на валидна задача: въвеждане на име на задачата, избор на тип на схемата (електрическа или логическа), избор на потребителската роля, чрез която се създава/отваря СС. След първоначалното установяване на задачата, може да бъде добавена: формулировка, словесно обяснение, директиви за изпълнение на задачата от О, както и да се конструира СС като множество от елементи и връзки между тях. Фиг. 4 представлява ДД над даден елемент: преместване, изтриване, добавяне на нов елемент, завъртане на елемента (ако е позволено) и свързване на два елемента. Ще отбележим, че действията и на фиг. 3 и на фиг. 4 се изпълняват паралелно, което повишава гъвкавостта и продуктивността на работата на П.



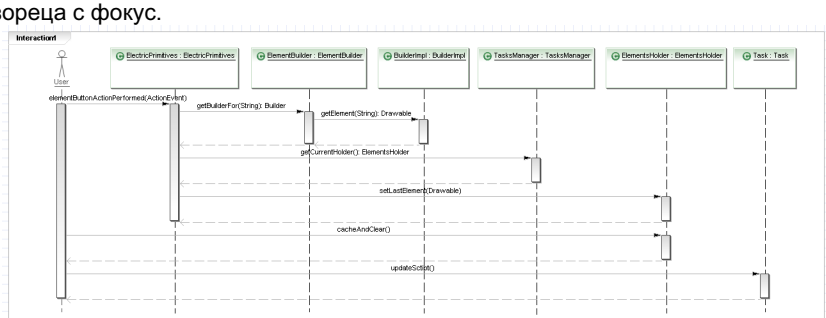
Фиг. 3. ДД при създаване на нова задача



Фиг. 4. ДД при избор на елемент

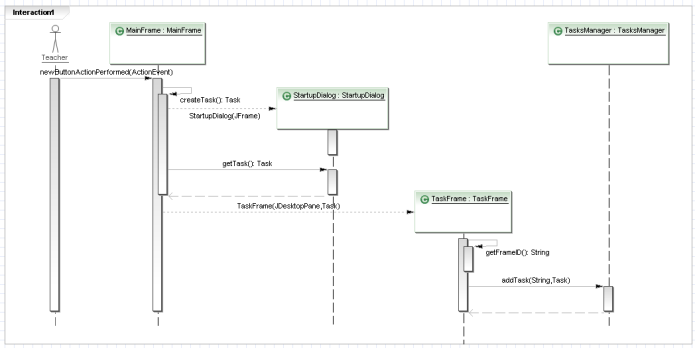
ДИАГРАМИ НА ПОСЛЕДОВАТЕЛНОСТИТЕ

Докато предишните две диаграми са езиково и технологично независими, диаграмата на последователностите (ДП) е зависима от обектно-ориентираната реализация на JAVA. На фиг. 5 хоризонтално е показана последователността от стъпки за добавяне на елемент от момента на избирането му до поставянето му в схемата и в списъка с елементи на текущата задача. След избор на елемента от лентата с наличните графични примитиви, се извиква клас, който конструира дадения тип елементи и създава нов обект, представляващ натиснатия бутон, или копира вече съществуващ в паметта му. Върнатият елемент се добавя към текущата задача, която е извлечена от класа **TasksManager**. Той следи текущия прозорец и при смяна на фокуса на друг прозорец сменя текущата задача да отговаря на прозореца с фокус.



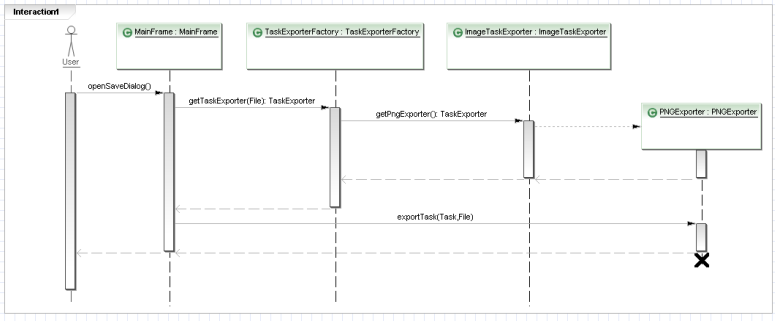
Фиг. 5. ДП за добавяне на елемент към схема

На фиг. 6 са показани стъпките при създаване на нова задача. При избор на съответната меню команда се извиква класът-диалог **StartupDialog**, помагащ за спазването на стъпките за създаване на нова задача. След като се създаде, задачата се използва за първоначално установяване на прозореца, който се грижи за визуализирането й от класа **TaskFrame**. Той я добавя към списъка на отворените задачи в класа **TasksManager**, задавайки текущия прозорец и задача като активни.



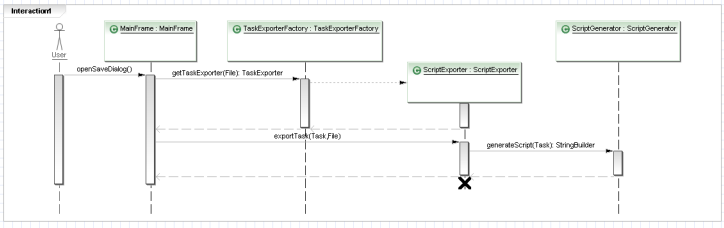
Фиг. 6. ДП за създаване на нова задача

Фиг. 7 представлява ДП с класовете и техните методи за съхраняване на схемата като растерно изображение във формат **Portable Network Graphics** (**PNG**). ДД на фиг. 8 показва класовете с техните методи при генерирането и записа на скриптовия файл на задача, а ДП на фиг. 9 последователността на действията при отваряне на този файл.

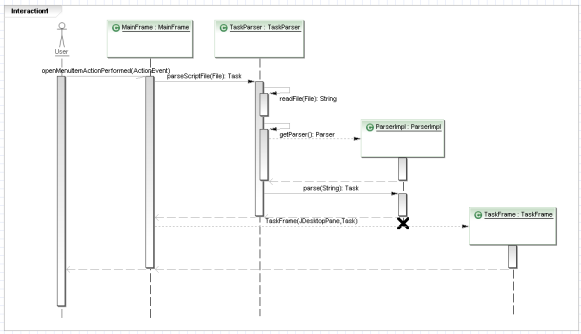


Фиг.7. ДП за запазване на схемата в PNG формат.

След избор на името на файла, той се прочита от класа **TasksParser**, който анализира и проверява дали даденият скрипт е валиден. Ако да, извлича нужните данни за изчертаване на схемата, след което скриптът се визуализира в нов прозорец от тип **TaskFrame**.



Фиг. 8. ДП за генериране на скриптов файл.



Фиг. 9. ДП за прочитане и визуализиране на скриптов файл

ЗАКЛЮЧЕНИЕ

Представен е UML проект от три популярни вида диаграми, касаещи нов JAVA вариант на авторско средство за построяване на структурни схеми. Този вариант се отличава със силно интерактивен и паралелен алгоритъм на работа с потребителя, който може да бъде и обучаван. Потребителският интерфейс е даден в друга работа.

ЛИТЕРАТУРА

[1] Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modelling Language, 3rd edition, Addison Wesley, 2004.

[2] Dimitrova S., Lyubenov D. Three-Layer Architecture of Environment for Web-Based Practical Education, Proceedings of Fourth International Conference COMPUTER CSIENCE, Kavala, Greece, Sept. 18^{th} –19^{th}, 2008, pp. 581-585.

[3] Димитрова, С, П. Радойска П. Методика за генериране и решаване на задачи с електрически вериги в Web-базирани приложения, сп. Автоматика и Информатика, кн. 4/2005, стр. 60-63.

[4] Матеев В., Виртуален инструмент за построяване на таблици на истинност на аналитично зададени булеви функции, Сп. “Автоматика и Информатика”, кн. 2/2008, стр. 45-47.

[5] Якимова Е. Подпомагане на дейността на обучаваните чрез решатели на задачи от математическата логика, “Автоматика и Информатика”, кн. 3/2007, стр. 63-66.

[6] Zheliazkova I. I., Georgiev G. T., Valkova P. L. A Task-Oriented Environment for Structural Schemes Design, *Information Technologies and Control*, Vol. 2, 2006, pp. 2-13.

За контакти:

Доц. д-р Ирина Желязкова, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, тел.: 082-888 711, e-mail: Irina@ecs.ru.acad.bg

Докладът е рецензиран.