Графичен редактор за създаване и редактиране на Java клас-диаграми

Валентин Великов, Каталина Григорова, Александър Илиев

Abstract: Graphic Editor for Java Class Diagram Creating and Editing: *The article discusses some problems with the automated software generation, and in particular - one of the first stages - describe and model the subject area. Appropriate choice for these are graphical environments for UML-diagrams. The necessity of own Java-class diagrams editor determinated, its functional capabilities, requirements, architecture.*

Key words: *Computer Science, graphical UML editor, Java class-diagrams editor.*

ВЪВЕДЕНИЕ

Автоматизирано генериране на софтуер - това е проблем, свързан не само с печелене на време и човешки труд за разработване. Започвайки от проектирането и моделирането на предметната област, до получаването на крайния продукт - генерацията на програми позволява създаването на синтактично чисти и логически коректни модули, което от своя страна също е свързано с времето и средствата за създаване на софтуер [1]. За една от първите стъпки - моделирането на предментната област – се използват различни видове диаграми. Тяхна разновидност са клас-диаграмите. За тяхното създаване и редактиране се използват различни инструменти, обикновено – графични редактори. Съществуват множесто системи: с отворен код, web-базирани и др.[2].

По-голямата част от тези системи обаче не са с отворен код, т.е. – вътрешно-машинното представяне на структурите не е налично (Know-How). Дори да бъде установено, под въпрос е евентуалната манипулация на тези структури (авторски права и др.). Освен това - по-добрите продукти са комерсиални. Цената им е такава, че дори големите български разработчици и потенциални клиенти биха се замислили. Това отваря пазарна ниша за нови разработки в областта.

ИЗЛОЖЕНИЕ

Като поредна стъпка в изграждана теория, и като първа стъпка в практическата реализация на съответната приложна система, може да бъде разгледан създаденият графичен редактор (самостоятелна и функционираща подсистема) за създаване, визуализация и редактиране на Java клас-диаграми.

В областта на софтуерното инженерство клас диаграмите [3] са част от Унифицирания език за моделиране(UML) [4]. Целта им е подпомагане описанието на проектирана софтуерна система чрез нагледно показване на класовете, атрибутите и методите им, както и връзките между тях. Клас диаграмата е основен градивен елемент при обектно-ориентираното моделиране.

Клас-диаграмите се използват като всеки от класовете се представя чрез отделен собствен сектор, съдържащ името на класа, атрибутите и методите. Отделните класове се асоциират с връзки помежду си, ако това е нужно. Интуитивният подход към моделирането, който те предлагат, ги прави едно от най-често използваните решения при моделиране на обектно-ориентирани системи.

Нуждата от прецизни и ефективни инструменти, подпомагащи моделирането, е довела до разработката на набор от различни компютърни системи, даващи възможност за разработка и редактиране на клас диаграми.[2] Такива софтуерни системи често са ориентирани към цялостна UML поддръжка, включваща работа с останалите компоненти на езика.

Общо за почти всички инструменти, поддържащи UML, е стремежът за предоставяне на графична среда, чрез която потребителят да работи. Алтернативни

решения съществуват, но те предизвикват набор от неудобства, основно поради трудното представяне на общия вид на проекта в неграфичен режим. Основните проблеми, които се поставят пред всички системи с графичен интерфейс, са:

- ефективност – средата трябва да улеснява потребителя; да предоставя технологии за спестяване на време; визуализацията на даден проект трябва да е ясна и да представя решението точно;
- надеждност – системата не трябва да е предразположена към грешки; данните трябва да бъдат съхранявани и обработвани коректно;
- бързодействие – графичната среда изисква повече технически ресурс за ефикасна работа; системата трябва да е оптимизирана и да не бави процеса на работа

По отношение на поддръжката на работа с клас диаграми, системите трябва да поддържат графична среда, чрез която потребителят да оформя проекта, да въвежда данни и да анализира крайния резултат. Всички продукти спазват общоприети норми за оформлението на средата, използваните фигури за представяне, връзките между тях и начина на работа.

От своя страна, клас диаграмите, изложени графично на потребителя, имат за цел по-удобно представяне на елементите на един клас. Освен това, те визуализират структурата на класовете и връзките между тях еднозначно, независимо от програмния език, използван за реализация.

При създаването на тази система следва да се има предвид, че тя не е предназначена за широк кръг потребители, а за неголям кръг специалисти, познаващи добре предметната област.

Тези специалисти може да не са програмисти (най-вероятно), за това към тази подсистема не се поставя задача за генериране на Java-код, а само за описание на обекти и връзки между тях, на логическо ниво. На по-късен етап, друга подсистема - за генериране на Java-код - ще поиска (вече от програмисти на някакво ниво) типове на величините, модификатори и др.; структури, в които те да се разполагат (масиви, файлове, ArrayList, Vektor…); допълнително описание на подпрограмите за обработка (описание на параметри – брой и тип, както и тип на резултата).

Създаваният редактор на диаграми ще предостави и още нещо: известно вътрешно-машинно представяне на обектите от логическия проект, без което създаването на следващите подсистеми от проекта би било много трудно.

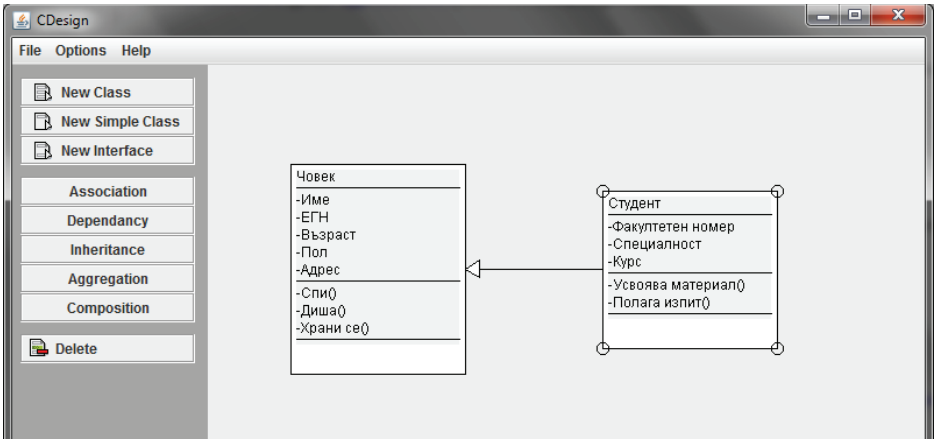
Този редактор представлява първа стъпка от реализацията на система за автоматизиране на рутинни дейности, свързани с проектиране, създаване или редактиране на Java SE/Java EE приложения.

Поставени изисквания към създаваната система:

Изискванията, наложени върху системата, се формулират спрямо поставената задача и проблемите, които възникват при реализирането на проекта:

- Изисквания към интерфейса и средата:
  - Процесът на създаване и редактиране на отделните елементи, съставляващи клас диаграма, да бъде извършван в графична среда (;
  - Средата (фиг.1)да представя нагледно съвкупността от данни, елементи и готови диаграми; да позволява интуитивна и ефективна работа с тях;
  - Интерфейсът да предоставя необходимите инструменти за боравене с програмата; те да бъдат достъпни и интуитивни за ползване
  - Представянето на диаграмите да става според стандартите на UML 2.0. Класовете да могат да бъдат визуализирани, заедно с техните атрибути и методи, както и връзките между тях.
- Изисквания към функционалността на системата:

- Да бъде реализирано изчертаване, обработване (чрез въвеждане или изтриване на данни) и преместване на отделните елементи от клас-диаграмите;
- Да съществува възможност за запазване на проект, както и за зареждане на проект от запазен файл;
- Избирането на опции за работа с програмата да става чрез селекция от меню; за работа да се използва мишка;
- Въвеждането и изтриването на текстови данни (имена на атрибути, методи и т.н.) да става посредством клавиатура.



Фигура 9 Графична среда на редактора

- Изисквания към управлението на грешки
  - Да бъдат обработвани грешки при запазване и зареждане на проект;
  - Да бъдат обработвани грешки при некоректно въведени данни, невалидно позиционирани елементи и т.н.

Организация на данните и тяхното вътрешномашинно представяне:

Системата поддържа структури, съхраняващи данните за класовете и връзките – двата основни елементи при изграждането на клас-диаграмите. На по-късен етап използваните данни се организират в по-сложни структурни единици.

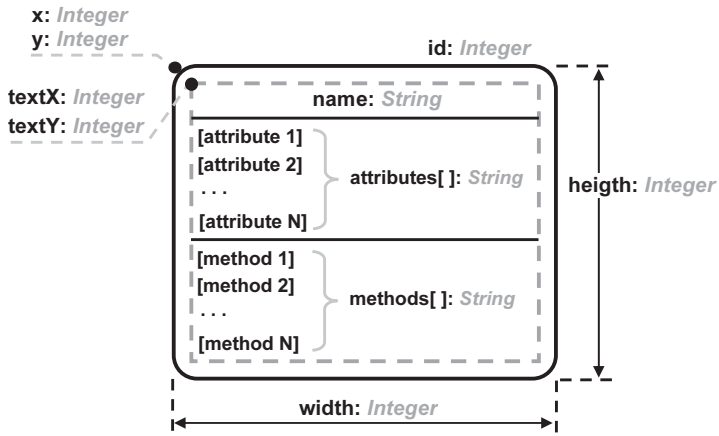
• Данни за клас

За всеки клас трябва да се организира и поддържа информация за следните данни и техни характеристики:

- Име на класа;
- Атрибути на класа (могат да бъдат повече от един) - масив от елементи – ArrayList;
- Методи на класа (могат да бъдат повече от един) - масив от елементи ArrayList;
- Размер на класа в 2D пространството (отнася се за представянето му в графичната среда);
- Координати на класа в 2D пространството (отнася се за позиционирането му в графичната среда);
- Уникален номер на класа (за евентуално различаване на класове с едно и също име);

Фигура 2 онагледява графичното представяне на един клас и основните му параметри:

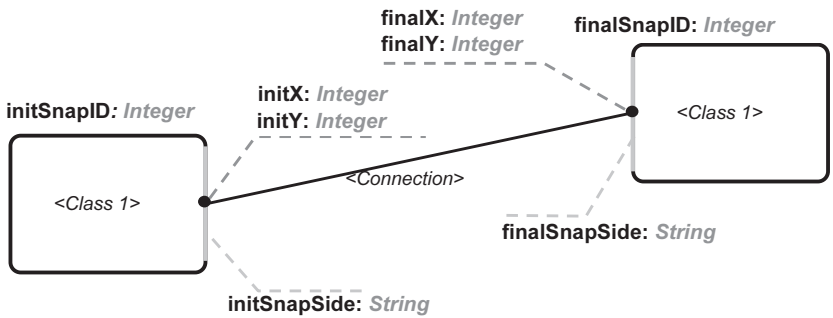
Данните за класа трябва да бъдат структурирани в подходяща база от данни. За всеки клас трябва да бъде поддържана отделна структурна единица, организираща тези данни. Ключово е да могат да бъдат създавани и запазвани произволен (неограничен) брой записи за атрибути и методи на всеки клас. За всеки клас е нужно създаването на отделен запис, съдържащ всичките му данни. Начинът на използване на програмата предполага, че във всеки един момент могат да съществуват нула или произволен брой класове, затова системата трябва да поддържа динамично създаване и изтриване на записи за всеки клас. Това се реализира чрез използването на динамичен масив за съхранение на произволен брой класове. При създаването на нов клас в системата ще бъде добавян нов елемент в този масив, в който елемент ще бъдат въвеждани данните за класа.



Фигура 2. Променливи, представящи един клас

- Данни за връзка

Фигура 3 онагледява графичното представяне на една връзка и основните и



Фигура 3. Променливи, представящи една връзка

За всяка връзка трябва да бъде организирана и използвана информация от системата, описваща следните характеристики:

- Начална точка на връзката в 2D пространството (отнася се за представянето и' в графичната среда: X- и Y-координати);

- Крайна точка на връзката в 2D пространството (отнася се за представянето и' в графичната среда: X- и Y-координати);
- Клас, към който е свързана началната точка на връзката
- Клас, към който е свързана крайната точка на връзката

Данните за всяка една връзка трябва да бъдат организирани в подходяща база от данни, като за всяка връзка се поддържа отделна структурна единица. Във всеки един момент могат да съществуват нула или произволен брой връзки, затова системата трябва да поддържа динамично създаване и изтриване на записи за всяка връзка, т.е. - ArrayList.

При текущата реализация системата не следи дали подредбата на връзките описва наследяване на множество класове (позволено например в езика C) или на само един клас (в Java се позволява наследяване само на един клас). Това дава на потребителя по-голяма гъвкавост при работа и не налага ограничения. Не съществува проблем при необходимост да се добави флаг (с установявана в настройките стойност), чрез който да се разрешава или забранява множествено наследяване.

• Данни за състоянието на мишката

Системата пази данни за състоянието на мишката и ги използва при изчертаване или обработка на елементи в графичната среда. Ключовите данни, които се организират и обработват, са:

- Координатите на курсора в 2D пространството;
- Информация за работа с бутоните на мишката;

Тази информация трябва да бъде опреснявана непрестанно, за да се позволи работа с отделните функции за изчертаване в реално време

• Вътрешни данни за системата

Информацията за системата се отнася до параметри, участващи при представянето на потребителския интерфейс и на компонентите на системата. Данните, които трябва да бъдат организирани, са:

- Информация за наименованието на текущата инстанция на системата;
- Данни за компонентите в различните модули – наименование, координати и размери в 2D пространството.

ЗАКЛЮЧЕНИЕ

Реализираният проект представлява самостоятелно Java приложение (Standalone Java Application). Системата наследява характеристичната за Java портативност, което осигурява възможност за използване под всякакъв вид операционна система. Предимство на този подход е, че системата изглежда и работи еднакво, независимо от платформата, на която е стартирана, или от осигурявания от платформата интерфейс.

Забележка: Настоящата статия е изготвена с финансовата помощ на Европейския социален фонд. Русенският университет „Ангел Кънчев" носи цялата отговорност за съдържанието на настоящия документ, и при никакви обстоятелства не може да се приеме като официална позиция на Европейския съюз или Министерството на образованието и науката."

Проект № BG051PO001-3.3.06-0008 „Подпомагане израстването на научните кадри в инженерните науки и информационните технологии"

ЛИТЕРАТУРА

[1] Stanev I., K. Grigorova. Unified KBASE process, Knowledge Based Automated Software Engineering, Cambridge Scholars Publishing, 2012

[2] Velikov V. Systems for automated software development.// Proceedings of the Union of Scientists - Ruse, Book 5 - Mathematics, Informatics and Physics, 2013, No 10

[3] Class Diagrams, http://en.wikipedia.org/wiki/Class_diagram, , 02.2012г.

[4] Unified Modeling Language (UML),
http://en.wikipedia.org/wiki/Unified_Modeling_Language, 02.2012г.

За контакти:

Гл. ас. Валентин Великов, кат. Информатика и информационни технологии, Русенски университет, GSM: +359 886 011 544, e-mail: val@ami.uni-ruse.bg.

Доц. д-р. Каталина Григорова, кат. Информатика и информационни технологии, Русенски университет, +359 82 888 470, e-mail: kgrigorova@uni-ruse.bg.

Бак. Александър Илиев, GSM: 0899 702 159, e-mail: al.juggernation@gmail.com