Христов, Хр., „Трудности и решения при смяна на парадигмата. Преподаване на обектноориентиран анализ, дизайн и програмиране”, Доклади на международна конференция „Взаимодействието теория – практика: Ключови проблеми и решения”, т. III, Бургас, 24-25 юни 2011, стр. 303-310

ТРУДНОСТИ И РЕШЕНИЯ ПРИ СМЯНА НА ПАРАДИГМАТА. ПРЕПОДАВАНЕ НА ОБЕКТНООРИЕНТИРАН АНАЛИЗ, ДИЗАЙН И ПРОГРАМИРАНЕ

Христо Тошков Христов, ФМИ при ПУ

DIFFICULTIES AND SOLUTIONS WHEN CHANGING THE PARADIGM. TEACHING OF OBJECTORIENTED ANALYSIS, DESIGN AND PROGRAMMING

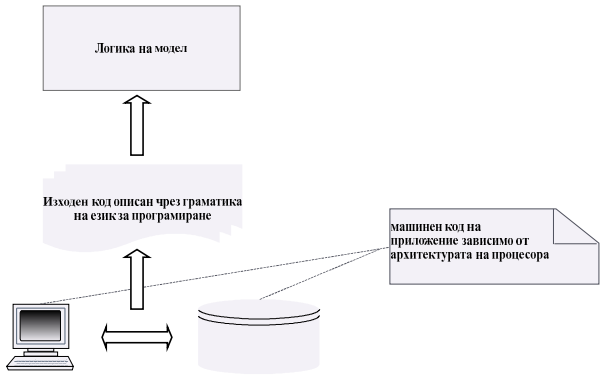
Hristo Toshkov Hristov

Abstract: In the paper is reviewed how the abstraction is carried out in the structural and the object-oriented manner of programming. A thesis is stated: the change of paradigm in the software industry, at present, has not led to change of the paradigm for teaching object-oriented technologies in education. Practical difficulties in teaching of the object-oriented paradigm are analyzed and suggestions for their overcoming are made.

keywords: programming paradigm, object-oriented analysis and design, programming languages, teaching methodology

1. Въведение – разгръщане на абстракцията и развитието ѝ в езиците за програмиране.

В развитието на езиците за програмиране с общо предназначение понятието абстракция е играло и играе ключова роля. Нейното разгръщане може да се погледне от гледна точка на приложението (отдолу-нагоре – фиг. №1) или да се разгледа като реализация на софтуерен процес (отгоре-надолу – фиг. №2). В първия случай се разглежда разгръщането на абстракцията над базовата машина, докато при втория интерес представлява разгръщането като концепция за създаването на софтуер.



Фиг. №1 Разгръщане на абстракцията (отдолу-нагоре) от гледна точка на софтуерното приложение

Разгръщането на абстракцията над базовата машина се е развивало с езиците за програмиране, чиито програмни реализации са зависими от архитектурата на Фон Нойман. Такива са асемблерните езици, които са малки абстракции над машинните езици, и структурните езици, които пък са абстракция над асемблерите. Всички те са абстракции над процесора, т.е. над архитектурата на компютъра. Основна черта на разгръщането на абстракцията над машината е зависимостта на програмата от архитектурата на компютъра. Поради тази причина в програмните реализации програмистът е трябвало да установи асоциация между машинния модел в пространство на решението и модела на задачата, която в действителност се решава в пространството на задачата. Усилието, необходимо за изпълнението на това преобразуване, и фактът, че такова изпълнение е неприсъщо за програмния език, генерира програми, които са трудни за писане и скъпи за поддръжка. Алтернативата на моделирането на машината е моделирането на задачата [1].

2. Моделиране и разгръщане на абстракцията над решението в обектноориентираната парадигма.

Все по-голямото нарастване на сложността на софтуера и обема изходен код е довело до търсене на друг подход за организация на програмите – алтернатива, която сменя моделирането над машината с моделиране на предметната област. Такъв модел на организация предлага обектноориентираният стил на програмиране. Преди неговото разработване много от софтуерните проекти са достигали праг на сложност, над който структурният подход вече не е работел. Обектноориентираната парадигма е създадена, като е взела най-добрите идеи от структурния стил на програмиране и ги е комбинирала с някои нови принципи и концепции. Резултат от това е различен начин за организиране на приложните програми с по-високо ниво на абстракция [2].

С появата на обектноориентирания стил на програмиране се слага повратен момент в създаването на софтуер. Разгръщането на абстракцията започва да се разглежда от гледна точка на концепцията, описваща предметната област и реализираща процес за създаване на софтуер.

При този стил на програмиране разгръщането на абстракцията над базовата машина престава да представлява интерес, тъй като в съвременните обектноориентирани езици изходният код се преобразува до т.нар. байткод, който се интерпретира от виртуална машина, независима от архитектурата на компютъра.

Обектноориентираната парадигма обаче, не се ограничава само с практики по програмиране и изучаване на обектноориентиран език за програмиране.

Изучаването на обектноориентираната парадигма на програмиране включва три елемента: изучаване на обектноориентиран анализ и проектиране, изучаване на обектноориентирано програмиране и изучаване на някой от обектноориентираните езици за програмиране [3]. Нещо повече, при обектноориентирания подход разгръщането на абстракцията като реализация на софтуерен процес създава взаимовръзка между елементите, които се изучават. Тази взаимовръзка се постига, тъй като е допустимо да се моделират обектни модели с различна степен на абстрактност в процеса на създаване на софтуер, т.е. независимо дали етапът е анализиране и създаване на архитектура, проектиране и изграждане на дизайн, програмиране или тестване на софтуерни модули. Освен това, за да се изучава как обектноориентираната парадигма се разгръща при създаването на приложение, е необходимо да се разгледат и преподадат отделните етапи на реализация на софтуерен процес. Следователно, можем да направим следните изводи, които да са предпоставка за изучаване и преподаване на обектноориентирана парадигма:

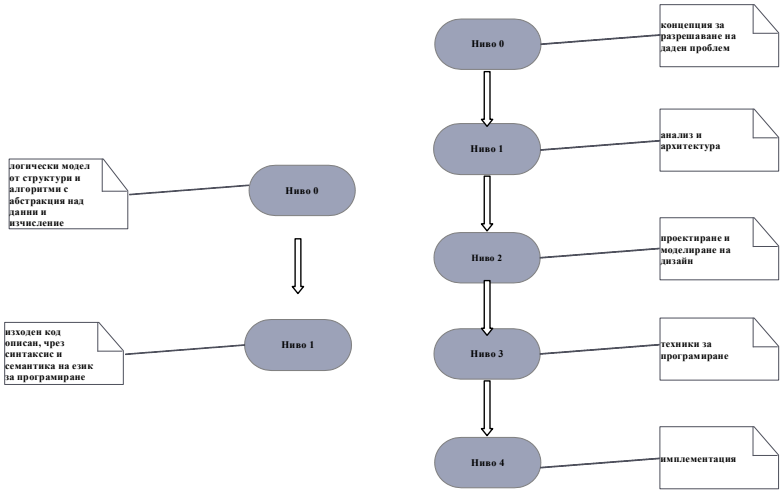
Извод №1: Независимо от етапа на реализация в процеса на създаване на софтуер, е допустимо да се изграждат и моделират обектни модели с различна степен на абстрактност;

Извод №2: За преподаването на обектноориентирана парадигма е необходимо да се изучава как се разгръща абстракцията в различните етапи от процеса за създаване на софтуер.

3. Различие в разгръщане на абстракцията при структурен и обектноориентиран стил на програмиране.

Нека посочим различието в разгръщането на абстракцията при двата стила – структурен и обектноориентиран, за да може, опирайки се на разликата, да разгледаме как това е повлияло по различен начин на развитието в подходите за програмиране и методиките на тяхното преподаване.

При структурния подход абстракцията се разгръща: от логически модел на сложни структури от данни и алгоритми за изчисление, през синтаксис и семантика (т.е. граматиката) на език за програмиране, с които се описва моделът, до машинното представяне на изходния код. При този стил имаме зависимост на софтуера от архитектурата на процесора и езика за програмиране. При обектноориентирания подход абстракцията се разгръща: от концепция за разрешаване на даден проблем, през анализ и архитектура, проектиране и моделиране на дизайн, техники за програмиране, до имплементация на най-ниско ниво, чрез синтаксис и семантика на език за програмиране.



Фиг. №2 Разгръщане на абстракцията (отгоре-надолу) от гледна точка на реализацията на софтуерен процес при структурния и обектноориентирания подходи

Извод №3: Характерно за структурния стил на програмиране е, че софтуерното приложение е зависимо от архитектурата на компютъра, а реализацията – от избора на език за програмиране. Обратно, при обектноориентирания подход имаме независимост на приложението от архитектурата на процесора, а реализацията на софтуерния процес не зависи от избора на езика за моделиране и програмиране.

4. Влияние на стила на програмиране и граматиката на езика върху методиката на преподаване.

Разгръщането на абстракцията е повлияло и върху методиките на преподаване на двата подхода. Който и учебник по процедурно програмиране да разгърнем, неизменно ще открием следната тематична последователност: *обща структура на програмата, прости величини и типове, изрази, оператори за присвояване, оператори за вход и изход, оператори за преходи и за цикли, съставни типове, функции, рекурсия и т.н.* [4].

От посочената последователност за излагане на учебно съдържание ясно проличава, че изграждането на структура за изучаване на този стил на програмиране се влияе и зависи от граматиката на езика за програмиране, т.е. *структурата на учебното съдържание следва структурата на граматиката на езика за програмиране*. Придобитата яснота за структура и последователност на излагане на учебно съдържание с течение на годините е улеснила задачите, които се разрешават от методиките на преподаване на структурен стил на програмиране. Обратно, в обектноориентираният стил на програмиране не е намерена уточнена и общоприета методология на излагане на учебно съдържание [5]. За съжаление, преподаването на обектноориентирано програмиране все още не може да се отърси от структурни практики на преподаване.

5. Различия между съвременните практики за създаване на приложения и начините на тяхното преподаване.

Какви са практиките за създаване на приложения в софтуерната индустрия? С какво тези практики се различават от начина на тяхното преподаване в учебните заведения? Как съвременните практики да се адаптират към спецификите на българската образователна система и потребностите на студента?

Оказва се, че в образователните институции се създава вакуум от разликата между актуалните практики за създаване на софтуер и начините на тяхното преподаване. Университети, колежи и училища не могат да отговорят на динамиката, с която еволюират процесите за създаване на софтуер, в резултат на което формите на обучение не могат да се актуализират и адаптират към бързите промени на съвременните софтуерни практики. В следствие на това страда обучаемият, на когото в някои случаи се налага да изучава начини за създаване на софтуер, които са част от историята на развитие на програмирането, без тези начини да съответстват като приложност на съвременните методи, използвани за реализация на софтуерни процеси.

По-конкретно ще разгледаме как смяната на парадигмата от структурен към обектноориентиран стил на програмиране не е довела до *смяна на парадигмата на преподаване* на обектноориентирано програмиране, причините за което са много и комплексни – някои от които ще посочим.

Теза: *Смяната на парадигмата на програмиране*, към настоящия момент, не е довела до *смяна на парадигмата на преподаване.*

От методическа и педагогическа гледна точка преподаването на обектноориентирана парадигма, като форма на обучение е трудна за реализация и е съпътствана от редица препятствия, повечето от които са с практически характер. В лекционните курсове, където преобладава теорията, тази смяна не е проблематична и се наблюдава. В практическите занятия, където преобладава работа със софтуерни среди и технологии, езици за моделиране и програмиране, тази смяна е сериозен проблем.

6. Причини, поради които е възпрепятствана смяната на парадигмата на преподаване.

От една страна, съществуват възпрепятстващи смяната на парадигмата обективни причини, които е необходимо да се оценят, за да се променят практиките на преподаване. Такива причини са силно повлияни от инерцията на разсъжденията, формирани от структурния стил на програмиране. Те са тясно свързани със структурата и последователността на излагане на учебното съдържание, организирано като абстракция над граматиката на език за програмиране, с цел да се изучи неговият синтаксис и семантика.

Следва да формулираме две обективни причини, които възпрепятстват изучаването на обектноориентираната парадигма:

(1) В преподаването на обектноориентиран стил на програмиране съществува практика, повлияна от структурни разсъждения, при която организацията на учебното съдържание се изгражда с цел изучаване на синтаксис и семантика на език за програмиране, а не въз основа на характеристиките на подхода за програмиране;

(2) Примерите, изучавани в занятията по упражнения, са конструирани с абстракция над граматиката на езика за програмиране, а не над концепцията и принципите на обектноориентираната парадигма.

От друга страна, за да се премахне субективизмът в преценката какво да се изучава, когато се преподават различни способи за създаване на обектноориентирани приложения и за да бъдат включени практики като анализ, проектиране, моделиране и др. в преподавателската дейност, е нужно да се проучи в какъв контекст тези практики се използват при създаването на софтуер.

Към настоящия момент реализацията на софтуер най-често се извършва по утвърден и общоприет фреймуърк (framework) за процеси. Тези работни рамки обикновено включват в себе си два процеса: бизнес процес за управление на проект и процес за създаване на софтуерно приложение. Изучаването на обектноориентиран анализ, проектиране, програмиране и т.н. е свързано с вторите – софтуерните процеси и техните методологии за реализация. При прохождането на обектната парадигма реализацията на софтуерните процеси се е извършвала по формализиран модел с изграден жизнен цикъл. Моделите на жизнения цикъл за създаване на софтуер разделят процеса на разработка на ясно обособени фази, обикновено включващи етапите анализ, дизайн, програмиране, тестване, интегриране и поддръжка [6]. Впоследствие силен тласък на развитие на моделите на жизнените цикли се постига, чрез разработването на множество от т.нар. динамични методи, още наричани гъвкави (agile). Динамичен метод (или методология) е обобщаващ термин, покриващ много процеси за създаване на софтуер, които имат общ набор от стандарти и принципи, дефинирани в *„Манифест за динамична разработка на софтуер”*[1]. Следователно това е първото нещо, което стои като контекст в софтуерните практики и е необходимо да се вземе предвид при преподаването на създаване на обектноориентирани приложения, т.е. необходимо е: *преподавателските практики да са съобразени с принципите на манифеста за динамична разработка на софтуер*. Също така, дейности като анализиране, проектиране, моделиране и т.н. са част от всяка методология за реализацията на софтуерен процес. Следователно второто нещо, което стои като контекст и е необходимо за преподаването на създаване на обектноориентирани приложения, е: *преподавателските практики да се базират на определени методологии за създаване на софтуерни процеси*. Разбира се, в един курс на обучение не може да се засегне в такава степен методологията за създаване на софтуер, както тя би се приложила от

[1] http://agilemanifesto.org/

професионални разработчици. Това от своя страна поставя друг съществен въпрос – в каква степен да се засягат методологиите за създаване на софтуер в преподаването? Съществуват ли подходящи методологии, които да послужат на преподавателската практика, или е необходимо да се създадат нови такива, които да са адаптирани към спецификите на образователна система?

Отговорите на поставените въпроси оставяме отворени за дискусия и няма да разглеждаме в тази работа. В нея ще разгледаме трудностите от практически характер, които възпрепятстват смяната на парадигмата на преподаване, и ще посочим технологии, подходящи за преподаване на обектноориентирана парадигма.

Извод №4: Съвременните методи за преподаване на обектноориентирана парадигма е необходимо да се съобразят с принципите на манифеста за динамична разработка на софтуер и да се базират на определена методология за реализация на софтуерен процес.

7. Трудности от практически характер, които възпрепятстват смяната на парадигмата на преподаване.

Независимо от методологията, теоретично не е трудно споменатите практики да бъдат преподавани, тъй като в лекционното занятие студентът участва преобладаващо като слушател. Трудност обаче настъпва, когато разгледаната теория трябва да придобие практически характер в занятие по упражнение, – тогава, когато на студента му се налага да работи с компютър.

За да се спазят добрите практики, свързани с дейностите в етапите анализ, проектиране и моделиране, е необходимо тези дейности да се визуализират и презентират като модели под някаква графична форма, които модели впоследствие да се имплементират в изходен код. За тази цел в обучението е необходимо да се използва: *графичен език за моделиране* и *обектно-ориентирана платформа за програмиране*, част от която е конкретен език за програмиране. Тези средства се използват, чрез интернирани среди, съответно за моделиране и за програмиране. Изучаването на език за моделиране и запознаването с платформа за програмиране е бавен и продължителен процес, който изисква удачно построена методика на преподаване, съответстваща на възможностите за усвояване на нови знания от студента. Допълнителна трудност представлява придобиването на умение студентът да използва интегрирана среда за моделиране и програмиране. Не без значение е, кои ще са езиците за моделиране и програмиране, в какви среди ще се използват и доколко тези средства са общоприети и стандартизирани.

8. Технологии, подходящи за преподаване на обектноориентирана парадигма.

Унифицираният език за моделиране (Unified Modeling Language) е най-широко приет в обектноориентираната общност и е сравнително лесен за изучаване и усвояване. Той е създаден като стандартно средство за изразяване на дизайн при създаването на обектноориентирани системи. В това си качество е най-подходящият инструмент, който може да се използва за проектиране и моделиране.

Силно развити платформи, които включват два от най-използваните езици за програмиране с общо предназначение – Java и C#, са JavaEE[2], поддържана от Oracle, и .NET Framework[3], подържана от Microsoft. Освен това, граматиките на езиците Java и C# са лесно асоциативни с мета-модела на UML нотацията – още една допълнителна

[2] http://download.oracle.com/javaee/

[3] http://www.microsoft.com/net/

причина тези езици да се препоръчват за изучаване на обектноориентирано програмиране.

Интегрираните среди, които използват UML, са много на брой. Критерии за подбор на подобни среди биха били техните възможности за ползване на UML стандарта в неговите три режима: като скици (sketch) за проектиране и моделиране и водене на дискусия върху дизайна; като подробен план (blueprint) в правото и обратно инженерство (forward and reverse engineering) и като средство за генериране на изходен код и документация. Допълнителни критерии могат да са: интегрирането в средата и на други средства за проектиране и моделиране на архитектура, база от данни, бизнес процеси за управление на проекти и др. Също така, в средата за моделиране може да се интегрира платформа за програмиране плюс среда за работа с платформата.

Конкретно, за проектиране и моделиране препоръчваме следните две среди, използващи всички възможности на UML стандарта, включително и за генериране на изходен код и документация:

- Software Ideas Modeler - http://www.softwareideas.net/
- Smart Development Environment (SDE) - http://www.visual-paradigm.com/product/sde/

Първата среда не изисква по-специално разучаване как да бъде използвана. Тя е лесна за употреба, а олекотената работа с нея я прави подходяща за обучение. Втората среда изисква отделно разучаване относно използването ѝ. Сложността и комплексният ѝ характер се дължи на това, че в нея са добавени допълнителни средства за моделиране освен UML нотацията. Ведно с това, в средата са интегрирани и следните среди за разработка:

- Eclipse - http://www.eclipse.org/
- Visual Studio.Net - http://www.microsoft.com/visualstudio/en-us/
- NetBeans - http://netbeans.org/
- IntelliJ IDEA - http://www.jetbrains.com/idea/

Последните посочени среди използват платформите JavaEE и .NET Framework. Езиците Java и C#, поддържани от тези платформи, са резонен избор като средство за програмиране, предвид, че това са най-дискутираните езици, както от професионални разработчици, така и от преподаватели и студенти.

9. Заключение.

В работата разгледахме как се разгръща абстракцията при реализация на софтуерни процеси със структурен и обектноориентиран подход на програмиране. Посочихме, че различието в разгръщането на абстракцията при двата подхода е повлияло по различен начин практиките на програмиране и методиките на тяхното преподаване, като заключихме, че: *(1) Независимо от етапа на реализация в процеса на създаване на софтуер, е допустимо да се изграждат и моделират обектни модели с различна степен на абстрактност; (2) За преподаването на обектноориентирана парадигма е необходимо да се изучава как се разгръща абстракцията в различните етапи от процеса за създаване на софтуер; (3) Характерно за структурния стил на програмиране е, че софтуерното приложение е зависимо от архитектурата на компютъра, а реализацията – от избора на език за програмиране. Обратно, при обектноориентирания подход имаме независимост на приложението от архитектурата на процесора, а реализацията на софтуерния процес не зависи от избора на езика за моделиране и програмиране.*

Направихме съпоставка между съвременни тенденции и практики за създаване на приложения и начините на тяхното преподаване, като застъпихме тезата, че: *смяната*

на парадигмата на програмиране, към настоящия момент, не е довела до смяна на парадигмата на преподаване.

Разискахме причините, които възпрепятстват смяната на парадигмата на преподаване, като стигнахме до извода, че: *(4) Съвременните методи за преподаване на обектноориентирана парадигма е необходимо да се съобразят с принципите на манифеста за динамична разработка на софтуер и да се базират на определена методология за реализация на софтуерен процес*.

Анализирахме трудности от приложен и практически характер в преподаването и направихме предложения за тяхното преодоляване, като посочихме технологии, подходящи за изучаване на обектноориентирана парадигма.

Литература:

1. Eckel, B. Thinking in Java 4th ed., US, NJ. Prentice Hall Inc, 2006.
2. Шилдт, Х. Java 2 - Ръководство на програмиста. С., СофтПрес ООД, 2001.
3. Донева, Р., Е. Сомова. Обектноориентирано програмиране. П. УИ, 2000.
4. Добрев, Д. Методология на избирателното поведение в процедурното програмиране, Доклади на Тридесет и пета пролетна конференция на Съюза на математиците в България, Боровец, 5-8 април, с.387-392, 2006.
5. Hristov, H. Review and Outlooks of the Means for Visualization of Syntax Semantics and Source Code. Procedural and Object-oriented Paradigm – Differences, Proceeding of the Anniversary International Conference REMIA, Plovdiv, December 10-12, pp. 443-451, 2010.
6. McKeen, J.D. Successful Development Strategies for Business Application Systems. MIS Q., vol. 7, pp. 47-56, 1983.