Христов, Хр. А. Христова, „Модели и гъвкави методи в обучението по компютърни науки“, Наука и технологии, том II, 2012, стр. 59-63, ISSN 1314-4111

МОДЕЛИ И ГЪВКАВИ МЕТОДИ В ОБУЧЕНИЕТО ПО КОМПЮТЪРНИ НАУКИ

Христо Христов*, **Антоанета Христова****

MODELS AND AGILE METHODS IN TEACHING COMPUTER STUDIES

Hristo Hristov*, Antoaneta Hristova**

** ФМИ при ПУ „Паисий Хилендарски”*
гр. Пловдив, п.к. 4003, бул. България 236
e-mail: hth@uni-plovdiv.bg,
hristo.toshkov@gmail.com

*** ПМГ „Иван Вазов”*
гр. Димитровград, п.к. 6400,
ул. Климент Охридски 1
e-mail: pmg@escom.bg

Abstract: The article discusses the development of software models and agile methods, their role in the life-cycle of software processes, and their place in teaching computer studies. Solutions have been suggested, taking into consideration the different needs of secondary education, higher education and software industry.
Keywords: teaching methodology, life cycle model, agile method

Резюме: В работата се разглежда развитието на софтуерните модели и гъвкави методи, тяхната роля в жизнения цикъл на софтуерните процеси и мястото им в обучението по компютърни науки. Предложени са решения за изучаването им, като е направено разграничение между нуждите на гимназиалното и висше образование и обучението в софтуерната индустрия.

1. Въведение

Практиките за разработване на софтуер, водещи началото си от средата на миналия век, постоянно се развиват. От създаването на първата компютърна програма до днес са разработени и продължават да се разработват и развиват множество технологии, софтуерни модели, гъвкави методи, процеси за създаване на софтуер, бизнес процеси и т.н. Постепенното нарастване на сложността на софтуерната разработка е довело до образуване на методологии и работни рамки (frameworks) посредством които да се реализират комплексни процеси за изграждане на софтуерни приложения. Въпреки това бурно развитие на множество софтуерни дейности, целта на всички тях си остава една и съща, да се повиши ефективността и ефикасността и улесни реализацията на софтуерната разработка.

В работата се разглежда развитието на софтуерните модели и гъвкави методи, тяхната роля в жизнения цикъл на софтуерните процеси и мястото им в обучението по компютърни науки.

2. Развитие на практиките за създаване на софтуерни програми.

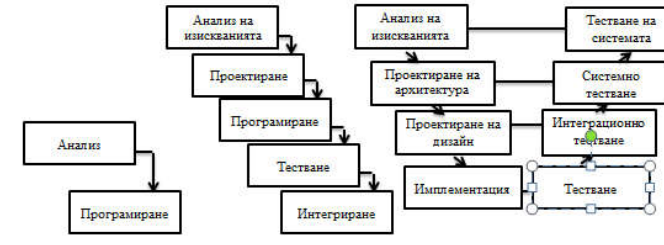
2.1. Началото на процеса на разработка. Основни етапи.

Има два основни етапа, които са общи за всяка компютърна разработка независимо от нейния размер и сложност – първия е *анализ*, втория е *програмиране*[1]. През петдесетте години на миналия век процесите за разработване на софтуер са се състояли от тези два стадия: *етап на анализ,* последван от *етап на програмиране.* Така реални обектите са описвани и моделирани в компютърните програми. За тази цел многократно са се разрешавали две задачи: 1) Дефиниране на абстрактни структури от данни, т.е. начинът, по който реалните обекти ще бъдат моделирани като математически обекти, както и определянето на множество от допустими операции над тях; 2) Реализацията на абстрактни структури от данни, т.е. начинът, по който дефинираните математически обекти ще бъдат

представени в паметта на компютъра, както и начинът, по който ще се реализират операциите с тях[2]. Моделът „Анализ-програмиране" *(фиг №1)* е бил сполучлив за малки програми, но практически той става неефективен и неефикасен за по-големи програми[1]. През 80-те и 90-те години многократното решаване на посочените две задачи е довело до формирането на процеси за реализация на софтуерни приложения по определен модел с изграден жизнен цикъл. Жизненият цикъл на моделите разделя процеса на разработка на ясно обособени фази, обикновено включващи такива етапи като анализ, дизайн, програмиране, тестване, интегриране и поддръжка[3].

2.2. Типове модели с жизнен цикъл.

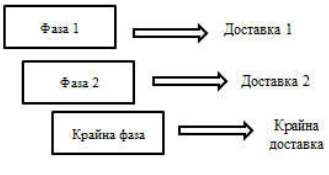
В зависимост от потока на работа и дейностите в етапите на разработка моделите се класифицират като последователни, прогресивни (поетапни) и итеративни[4]. При някои модели етапите на работа се прилагат последователно като фази, при други те са разделени на множество практики, които, обединени като дейности по определени критерии, се изпълняват итеративно. Първите модели са притежавали последователен стил на работа по отношение на етапите в жизнения цикъл. Най-широко срещаните представители на този стил са моделите водопад *(фиг. №2)* и традиционният V-модел *(фиг. №3).* Макар че има разлика в подхода на разработка, от гледната точка на съвременните методологии двата модела притежават сходни недостатъци. Главният проблем на тези модели е липсата на гъвкавост. Те са особено неефективни и неефикасни по отношение на променящите се изисквания на клиентите[5]. Поради тази причина тези модели вече не се използват широко в софтуерното инженерство. Другата категория са т. нар. прогресивни модели *(фиг. №4)* по отношение на фазите на разработка. При тях на всяка фаза се доставя междинна версия на софтуера. Третата категория модели са с итеративен стил на работа. Примитивен жизнен цикъл на итеративния модел се състои от четири фази: анализ на изискванията, проектиране, изпълнение и тест и преглед[6] *(фиг. №5).* При итеративният стил разработката се разделя на части въз основа на реализация на функционалност, т.е. предварително се анализират част от изискванията и се планува постигането на функционалност за определено подмножество изисквания към развиваната система. Характерно при този стил е, че в зависимост от оценката на функционалността се определя итеративното повторение на дейностите. Заради тези си качества итеративните модели са запазили употребата си и продължават да се развиват и усъвършенстват, като се интегрират в различни методологии.



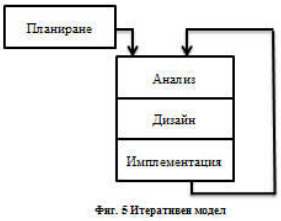
Фиг. 1 Модел „Анализ-Програмиране"

Фиг. 2 Модел „Водопад"

Фиг. 3 V-модел



Фиг. 4 Прогресивен модел



Фиг. 5 Итеративен модел

2.3. Гъвкави методи на разработка.

С еволюцията на езиците за програмиране от общо предназначение сложността на софтуерната разработка непрестанно се е повишавала. Противодействие за справяне със сложността е усъвършенстването на моделите, вследствие на което се появяват множество софтуерни методи. Понятието софтуерен метод е термин, включващ множество дейности и

практики на работа, които са част от процеса на разработка на софтуер. Чрез методите, процесът на разработка се превръща в сложно множество от взаимосвързани практики. Силен тласък на развитие на методите се постига чрез разработването на 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”*[7]. Такива методи, които са се утвърдили с времето и са общоприети от софтуерната общност, са Rational Unified Process/Unified Process (RUP/UP), SCRUM, Extreme Programming (XP), Crystal, Dynamic System Development Method (DSDM), MSF (Microsoft Solution Framework) и др. Гъвкавите методологии намират отговор на потребностите на софтуерната индустрия, като реализират по-лек софтуерен процес и по-бърза разработка. Използването на гъвкава методология за реализация на софтуерен процес е предпочитана от разработчиците. В сравнение с последователните и прогресивни модели, използването на гъвкава методология при итеративните модели предлага някои преимущества. Първо, използването на гъвкави методи може да осигури реализацията на работещ софтуер по-бързо. Гъвкавите методи по добре отговорят на променящите се изисквания по време на разработката. И също така, гъвкавите методи повишават взаимоотношенията между всички заинтересовани страни в процеса на разработка[8].

При такова развитие на подходи и способи за създаване на софтуер какво се случва с формите и средствата за тяхното преподаване и изучаване?

3. Гимназиалното образование, висшето образование и индустриалното обучение по компютърни науки.

Следва да се разгледа състоянието на обучението по компютърни науки в едно, с което се предлагат мерки за неговото осъвременяване, актуални на пазара на труда и реалността в софтуерната индустрия. В работата се прави разграничение между гимназиалното образование, висшето образование и индустриално обучението, поради необходимостта от различни решения в трите области.

3.1. Обучение по информатика в средното образование.

За да се определи състоянието на средното образование по информатика, е необходимо да се види равнището на държавните образователни изисквания (ДОИ) съпоставено на съвременната софтуерна реалност. Те за областта на информатиката са указани в Приложение №3[10] от настояще действащата Наредба[11]. В тези нормативни документи всички изисквания към ученика за придобиване на знания и умения, що се касае до създаването на софтуер, са пряко свързани с практики по програмиране. От друга страна, в методологиите за изграждане на софтуер програмирането е само една от дейностите, чрез които се създават приложни програми. В ДОИ, дейностите, като анализиране на изисквания, изграждане на архитектура, моделиране и проектиране, тестване и интегриране не са заложени. Обратно, обучението по програмиране е заело цялото пространство в процеса на създаване на софтуер. От трета страна, тези дейности са взаимосвързани, част са от единен процес и обхващат различни софтуерни практики, чиято крайна цел е създаването на софтуерен продукт. Поради тази причина е потребно за всеки обучаем те да се изучават. При това е необходимо обучението да се извършва в определена последователност според утвърдена методология и методика за нейното преподаване. Това е особено наложително за профил информатика, при който ученикът, завършил 12-ти клас, е специалист по компютърни науки, съгласно списъка на професиите за професионално образование и обучение[12], а де факто дори не е запознат с условията при които се създават софтуерни приложения! Причината за този парадокс се крие в бурния темп на развитие на технологиите, на които държавните институции не успяват да отговорят. Основни фактори,

поради които не е настъпила промяна във формите и методите на преподаване са: *1) Смяната на парадигмата от структурен към обектноориентиран стил на програмиране през деветдесетте години не е довела до смяна на парадигмата на преподаване на програмирането[13]; 2) Изучаването на стилове и езици за програмиране не се разглежда в контекста на реализация на бизнес и софтуерен процес, като част от методология (модели и гъвкави методи); 3) В обучението липсват разработени методики за преподаване на методологии за създаване на софтуер.*

3.2. Обучението по компютърни науки във висшето образование.

Обобщение за нивото на обучение във висшето образование е трудно да се прави. Висшите учебни заведения (ВУЗ) са с академична автономност и самоуправление и според Закона за висшето образование, самостоятелно разработват и изпълняват учебни планове. В учебните планове и програми на различните ВУЗ-ове фигурират дисциплини обхващащи различни етапи от разработването на софтуер. Интересно явление е, че тези предмети се изучават в последните години от курса на обучение на студента. През първата и втората година от студентското образование фигурират информатични дисциплини предимно от областта на програмирането. От друга страна, в практиката за разработване на софтуер, независимо каква методология и процес се използва, програмирането е предхождано от дейности по анализ, проектиране, моделиране и т.н. В полза на студента е, в учебните планове да бъде заложена последователността на изучаваните предметните области съобразно естествения ход на прилагане на моделите с изграден жизнен цикъл и гъвкавите методи в софтуерния процес, т.е. *дисциплини, които включват посочените дейности, да се изучават преди тези по програмиране*. По този начин още в началните курсове студентът ще знае мястото на програмирането при създаването на софтуерни приложения. Нещо повече, дейностите по анализ и проектиране са слабо технологично зависими и са по-лесни за усвояване.

3.3. Обучението по компютърни науки в софтуерната индустрия.

Съвремието ни е свидетел на най-различни учебни курсове. Обучението по компютърни науки в софтуерната индустрия е най-трудно да се анализира, тъй като то се развива на свободни пазарни принципи. Това води, от една страна, до наличието на много курсове с ниско качествено. От друга страна, големите софтуерни компании и фирми предлагат сертифицирани курсове, които са плод на дългогодишен опит и са съобразени с тенденциите на развитие. Такива курсове безспорно имат място в процеса на обучение по компютърни науки, а по-тясното сътрудничество между академични и бизнес институции е от взаимна полза и е потребно за студента.

4. Решения.

Принципно-организационни решения:

1) Ограничението на обучението само до програмистки практики налага промяна на ДОИ, чрез която да се одобри и стандартизира изучаването и на други дейности, част от съвременните методологии за създаване на софтуер.

2) Необходимо е да се намери подходяща структура на учебните планове във ВУЗ-вете, в която последователността на изучаване на учебните дисциплини да съответства на методологиите за реализация на софтуерни процеси, т.е. предмети, които обхващат дейности като анализ, проектиране, моделиране и т.н. да предхождат предмети, в които се изучават стилове, техники и езици за програмиране.

3) Нужно е да се потърсят форми на тясно сътрудничество между стандартизираните учебни курсове в големите софтуерните компании и кредитните изборни дисциплини и практикуми във ВУЗ.

Христов, Хр., А. Христова, „Модели и гъвкави методи в обучението по компютърни науки“, Наука и технологии, том II, 2012, стр. 59-63, ISSN 1314-4111

4.2. Практични решения.

За целите на образованието удачно за преподаване на съвременни софтуерни практики е проектът Eclipse Process Framework[14], разработван от Eclipse общността и поддържан от IBM. Участниците в проекта са разработили и развиват интегрирана среда за моделиране с графична и текстова нотация - Eclipse Process Framework Composer. Екосистемата на проекта предлага богат набор от софтуерни практики с отворен код достъпни от интернет страницата и интегрираната среда за работа. Проектът поддържа и развива процеса за създаване на софтуер OpenUP и методологията Business Driven Development. Чрез средата за работа могат да се използват и развиват всички интегрирани софтуерни практики и процеси, както и да се създават нови такива.

Преподаването на процеса OpenUP като част от проекта Eclipse Process Framework и експерименти за изучаване на водещи софтуерни практики посредством средата EPF Composer се извършва във ФМИ на ПУ в изборна дисциплина „Анализ, дизайн и изграждане на софтуер“.

5. Заключение

В образователните институции се е създал 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 в софтуерните практики. Считаме, че е нужна по-интензивна работа от всички заинтересовани страни, за да може това статукво да бъде променено.

Цитирана литература:

1. Royce, W. 1970. Managing the development of large software systems, Proceedings of the IEEE WESCON, 1-9.
2. Наков, П., П. Добринков, 2005. Програмиране Програмиране=++Алгоритми. III., София, ISBN 954-8905-06-X.
3. McKeen, J.D. 1983. Successful Development Strategies for Business Application Systems, MIS Quarterly, vol. 7, No. 3, 47-65
4. Xihui Zhang, Tao Hu, Hua Dai and Xiang Li, 2010. Software Development Methodologies, Trends and Implications: A Testing Centric View. Information Technology Journal, vol. 9 (8), 1747-1753
5. Sommerville, I., 2007. Software Engineering. 8th edn., Addison-Wesley, Boston, MA.
6. IPL, 2011. Software Testing and Software Development Lifecycles. Information Processing Ltd., www.ipl.com/pdf/p0821.pdf, последно посещение на 21.02.2012
7. Официален сайт, 2001, http://agilemanifesto.org/, последно посещение на 14.04.2012.
8. Martens, R., I. Gat, 2009. Guest view: Wrestling with scaling software agility. Software Development Times, http://www.sdtimes.com/link/33988, последно посещение на 23.02.2012
9. Kroll, P., W. Royce, 2005, Key principles for business-driven developmen. Developer Works, IBM , http://www.ibm.com/developerworks/rational/library/oct05/kroll/index.html, последно посещение на 03.02.2012
10. Официален сайт на МОН, ДОИ, Приложение№3, http://mon.bg/top_menu/general/doi, последно посещение на 23.03.2012.
11. Официален сайт на МОН, ДОИ, НАРЕДБА № 2 от 18.05.2000 г. за учебното съдържание, http://mon.bg/top_menu/general/doi, последно посещение на 23.03.2012.
12. Официален сайт на МОН, Професионално образование, Списък на професиите за професионално образование и обучение - допълнен и изменен със Заповед № РД 09-1805/09.12.2011 г., http://mon.bg/top_menu/vocational, последно посещение на 23.03.2012.
13. Христов, Хр. 2011, 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, стр. 303-310
14. Официален сайт на фондация Eclipse, проект Eclipse Process Framework, http://www.eclipse.org/projects/project.php?id=technology.epf, последно посещение на 11.04.2012.