Христов, Хр, Хр. Крушков, „Обучението по информатика и софтуерни технологии: развитие и перспективи“, Образование и технологии, Бургас, година VI, кн.6, 2015г., стр. 388-394. ISSN 1314-1791 (Print), ISSN 1314-1791 (Online) http://itlearning-bg.com/magazines/Spisanie2015/resources/spisanie_e_book_2015.pdf

ОБУЧЕНИЕТО ПО ИНФОРМАТИКА И СОФТУЕРНИ ТЕХНОЛОГИИ: РАЗВИТИЕ И ПЕРСПЕКТИВИ

ас. Христо Тошков Христов, доц. д-р Христо Димитров Крушков ПУ „Паисий Хилендарски“, Факултет по математика и информатика, гр. Пловдив

EDUCATION IN INFORMATICS AND SOFTWARE ENGINEERING: DEVELOPMENT AND FUTURE DIRECTIONS

Hristo Toshkov Hristov, Hristo Dimitrov Krushkov Plovdiv University „Paisii Hilendarski“, Faculty of Mathematics and Informatics

* Авторите изказват благодарност към научен проект НИ 15 ФМИ-004 към Фонд „НИ“ на ПУ, „Иновативни фундаментални и приложни научни изследвания по компютърни науки, математика и педагогика на обучението“ за частичното финансиране на настоящата работа.

Abstract: This paper traces the development of teaching informatics and software engineering in Bulgarian higher education, which follows and pursues professional and scientific development of these areas. Prerequisites are analysed of software engineering to evolve from a discipline to bachelors and masters degree programmes. A review is made of developments and trends in the methods and tools for creation and implementation of software in the terms of software production. A model was developed based on critical analysis and evaluation, revealing prospects for future education in informatics and software engineering. Several levels of training are presented with different abstraction and difficulty called „Software practices“, „Object-oriented approaches, techniques and practises of software development“, „Modelling, creation, and application of software processes“, and „Management and implementation of software projects“.

Keywords: Software engineering, software development, teaching methodology

Въведение

През изминалия век и особено отчетливо през последните две десетилетия човечеството е изправено пред необходимостта да обработва все по-голям обем от информация. Намирането на решения за удовлетворяването на тази обществена потребност е комплексно социално явление, обединяващо редица различни по характер научни областни, всички от които вземат за свой предмет на изследване обекта информация. С времето, провеждането на проучвания и множество систематизирани изследвания над информацията довеждат до формирането на нова научна област, днес добре известна с името информатика.

Произход и развитие на образованието и обучението по информатика и компютърни науки в българското висше образование

За начало на обособяването на информатиката като самостоятелна научна дисциплина се приемат 30-те и 40-те години на XX в. [1]. За първи път терминът „информатика“ се използва от немския учен К. Щайнбух през 1957 г. в статията „Информатика: автоматизирана обработка на информацията“. Във Франция понятието информатика се въвежда от френския инженер Ф. Драйфус през 1962 г. като съкращение на израза „information automatique“ [2]. Информатиката е наука за автоматично обработване на информацията. Тя се занимава със събирането, систематизирането, представянето, структурирането, анализирането, съхраняването, преобразуването и разпространяването на информация [3]. Системното изследване и изучаване на тези информационни процеси обособява научната област и поставя началото на обучението по информатика и компютърни науки. Прецизната датировка е трудно да се проследи, тъй като обучението по информатика прохожда във висшето образование като част от процеса на обучение в някои математически дисциплини. Първоначално някои дисциплини, сред които „Теория на информацията“, „Дискретна математика“, „Теория на графите“, „Булева алгебра“, „Криптография“ и др., са се приемали и разглеждали като математически основи на информатиката. С разрастването на научната сфера на информатиката се появяват дисциплини като „Формални и програмни езици“, „Алгоритми и структури от данни“, „Анализ и дизайн на алгоритми“, „Дискретни структури и езици“, „Автомати и изчислимост“, „Бази от данни“ и др. Общо за тези учебни курсове е разработването на подходи за формализиране на неформални информационни обекти. Този процес на формализация на информационни обекти, подходящи за обработка от компютърна

технология, поставя научните и образователни основи на бъдещето учебно съдържание по информатика.

Според източниците [4, 5] в България образователната дейност по информатика датира от началото на 70-те години. Секторите на Единния център по математика и механика – „Основи на кибернетиката и теория на управлението“ и „Основи на математическото осигуряване“, са първите институционални организационни единици от тип катедра. Десетилетие по късно, през 1986 г., Факултетът по математика и механика на Софийския университет (СУ) е преименуван на Факултет по математика и информатика (ФМИ). През същата година е формирана и първата самостоятелна специалност „Информатика“ [6, 7]. Четири години по-късно Математическият факултет на Пловдивския университет (ПУ) също е преименуван на Факултет по математика и информатика (ФМИ). Днес всички университети в страната притежават факултет или катедра, които са специализирани в обучение по информатика.

Софтуерните технологии – от образователна дисциплина до образователна специалност

Ролята на информатиката да автоматизира информационни дейности, като отговоря на обществената потребност да се обменя и обработва голям обем информация, дава силен тласък на развитие на информационните и комуникационните технологии (ИКТ). Фундамент на това развитие е факторът софтуерно производство – от теоретичните основи и прилагането на иновационна мисъл, през реализацията на софтуерни проекти, до интеграцията и поддръжката на софтуерни продукти и услуги. Тези комплексни дейности, касаещи жизнения цикъл на софтуерните разработки, най-тясно се свързват с понятието софтуерни технологии (СТ), още назовавани с термина софтуерно инженерство (СИ). Терминът „софтуерно инженерство“ за първи път е дискутиран през 1968 г. на конференция на НАТО, посветена на софтуерната криза [8, 9]. В специализираната литература се срещат различни определения за СТ. Определенията, придобили по-широка популярност, са тези на Б. Наур, 1969 г.: установяването и използването на здрави принципи за разработване с цел по икономичен начин да се произведе софтуер, който е надежден и функционира ефективно върху реална машина; IEEE (Institute of Electrical and Electronics Engineers, Асоциацията на американските електронни инженери): софтуерните технологии са систематичен подход към разработването, експлоатирането, съпровождането и изваждането от експлоатация на софтуер; Феърли, 1984: технологична и мениджърска дисциплина, занимаваща се систематично с производството и съпровождането на софтуерните продукти, които се разработват за време и на основата на точно определени разходи [10, 11]. Общо за посочените по-горе определения е синонимната употреба на глаголите „разработвам“, „експлоатирам“, „произвеждам“, т.е. създаването и употребата на софтуер. Основна цел на СТ е ефективно разработване на качествен софтуер чрез съчетаване на научни подходи с технологични новости, мениджърски похвати и печеливши бизнес практики [12].

В България първият курс по СТ е четен през учебната 1984/85 г. на студенти от ФМИ на СУ. За тези тридесет години техниките и практиките на разработване на софтуер постоянно се развиват, а значимостта на областта в страната се е увеличавала, като от отделна дисциплина СТ са се превърнали в основно научно, професионално (т.е. индустриално) и образователно направление на информатиката. Днес този курс присъства в учебните планове на бакалавърските специалности на направление „Информатика и компютърни науки“, като при един и същ хорариум се забелязват различия най-вече в избора и задълбочеността на представяне на материала в лекциите и начина на организиране на упражненията [13]. Нещо повече, изучаването на направлението отдавна е преминало пределите на отделната дисциплина, като по настояще то се учи специализирано като самостоятелна бакалавърска и/или магистърска специалност във ФМИ на СУ, ФМИ на ПУ, ФМИ на ШУ, НБУ, ТУ-София, ТУ-Варна и др. Този процес на разширяване на обучението по СТ ще продължи и през следващите години с оглед икономическата, социалната и управленската (т. нар. е-управление) значимост на направлението.

Еволюция на софтуерното производство

В настоящата точка представяме синтезирано динамичния процес на еволюция на принципи, концепции, стандарти, подходи, методи, техники и практики за създаване на софтуер. Паралелно с анализа на този процес обобщаваме и обособяваме четири йерархични софтуерни области. Целта на обособяването на областите е да се изгради модел, чрез който се разкрива парадигмата на разработване, реализация и интеграция на софтуер и перспективата, която тази парадигма открива пред настоящето и бъдещо образование и обучение по информатика и софтуерни технологии. При изграждането на модела степенуваме отделните области като абстракция и сложност, т.е. щом една

област е поставена по-високо в модела, то тя е съставена от градивни елементи (т.е. принципи, концепции, стандарти и т.н.), които са с по-висока степен на абстракция по отношение на разработката на софтуер и с по-голяма степен на сложност по отношение на софтуерните проблеми, които разрешават.

Критерии за обособяване и обобщаване на софтуерни области

При обобщаването и обособяването на областите се придържаме към следните критерии: областта да притежава генерализиращи свойства по отношение на градивните елементи, а самите елементи, т.е. софтуерните принципи, концепции, стандарти, подходи и т.н., бидейки част от практиката в софтуерната индустрия, да се характеризират с масово приложение и употреба, трайност във времето и степен на технологична независимост от софтуерни платформи и системи.

Развитие на програмирането

Софтуерните дейности, които исторически и професионално са най-развивани през последните седем десетилетия, са: анализирането на софтуерни проблеми и програмирането на софтуерни решения. През 50-те години на миналия век „процесите“ (моделите) за разработване на софтуер са се състояли от тези два стадия: фаза на анализ и фаза на програмиране. С времето дейностите анализиране и програмиране са преминали през множество концептуални и технологични промени. Това, което прави всяко програмиране трудно, до голяма степен е мащабът. Програми, които се създават лесно за един компютър, стават значително по-сложни, когато се пренасят през пространството (до много потребители) и времето (да съществуват извън времето за изпълнение на единична програма) [14]. От позицията на настоящето може да обобщим, че езиците за програмиране, създадени през втората половина на миналия век, са оформили четири главни стила (парадигми) на програмиране: императивен, функционален, логически и обектно-ориентиран [15]. Обектно-ориентирания стил на програмиране се справя с нарастващата сложност на програмите и се е наложил в индустрията, предложен е от Алан Кей, считан за баща на обектно-ориентираното програмиране (ООП) [16]. Чисто технологичните предимства на обектно-ориентираните езици за програмиране пред останалите групи при решаването на задачи с общо предназначение са неоспорими. Обектно-ориентираната технология (ООТ) се е развила като средство за управление на сложността, присъщо на много различни видове системи. ООП моделът се е доказал като мощна и обединяваща концепция [17]. В потвърждение на констатациите говорят и данните в Таблица №1, съставена от индекса TIOBE [18] за ползваемост на езиците съобразно стила на програмиране. Както може да видим, групата на ООП езиците заема най-голям процентен дял, при това през последните години се запазва тенденцията този процент да нараства. Така, въз основа на изложения анализ, отчитайки, че основните ООП подходи, техники и практики включват дейностите анализ, дизайн и програмиране, съобразявайки се с гореописаните критерии, първата софтуерна област, която обособяваме, наричаме: обектно-ориентирани подходи, техники и практики за създаване на софтуер.

Таблица №1: Рейтинг на категориите програмни езици според индекса TIOBE

Категория езици	Рейтинг – май 2011	Рейтинг – май 2012	Рейтинг – май 2013
Обектно-ориентирани	57.3%	57.7%	58.2%
Процедурни	37.1%	36.6%	37.0%
Функционални	4.3%	3.9%	3.2%
Логически	1.4%	1.9%	1.7%

Развитие на софтуерните модели и методологии

Според някои автори моделите за разработване на софтуер са се зародили през 50-те и 60-те години [19, 20], като първоначални опити за приложение са направени от софтуерни екипи на IBM през 1957-58г. [21]. Въпреки това, в научната литература първият опит за стандартизиране на модел за разработване на софтуер е известен от работата на Уинстън Ройс, публикувана през 1970г. [22]. В нея е представен т.нар. каскаден модел, още известен с названието – модел водопад. Софтуерните програми от 50-те и 60-те години са разработвани по модел „анализ-програмиране“. Той е бил сполучлив за малки програми, но практически става неефективен и неефикасен за по-големи програми [22]. Това е довело

до създаването на нови модели за разработване на софтуер. Такива модели са били създадени и развивани през 70-те, 80-те и 90-те години на миналия век. През този период моделите, по които се е разработвал софтуер, се класифицират като последователни, прогресивни (поетапни) и итеративни [23]. С разрастване и усъвършенстване на различните модели, процесът на създаване на софтуер по определен модел е започнал да се изследва като софтуерен процес с изграден жизнен цикъл. Моделите на разработване на софтуер разделят процеса на разработка на ясно обособени фази, обикновено включващи етапи като анализ, дизайн, имплементация (програмиране), тестване (проверка), интегриране и поддръжка [24, 25]. По-известни представители за периода са модел „водопад“, V-модел, Dual V-модел, модел „прототип“, еволюционен модел и др. Развитие на софтуерните модели като средство за организиране на работния процес е съпътствано от постоянно създаване на нови концепции, подходи, техники и практики за тяхното прилагане. Този прогрес на развитие с течение на времето води до разглеждането на моделите като рамка за реализация на софтуерните процеси, а структурирането на процеса, по който се разработва софтуерът, започва да се обсъжда в научната литература и софтуерна практика като методология. Някои автори разглеждат този преход, като определят, че: МЕТОДОЛОГИЯ = МОДЕЛ + ТЕХНОЛОГИЯ(И) [26]. Методологиите намират отговор на потребностите на софтуерната индустрия през последните две десетилетия. Те се предпочитат пред традиционните софтуерни модели, тъй като предлагат гъвкаво приложение на софтуерния процес, водещо до по-бързо и качествено разработване на софтуер сравнено с тромавите и детайлно документирани модели. Понятието гъвкава софтуерна методология (agile methodology), още наричана динамична, е термин, включващ множество концепции, подходи, техники и практики за разработване на софтуер. Динамична методология е обобщаващ термин, покриващ много процеси за създаване на софтуер, които имат общ набор от ценности и принципи, определени от *Манифест за динамична разработка на софтуер* [27]. В специализираната научна литература са разработени графики, които разкриват произхода и еволюцията на софтуерните модели и гъвкави методологии [28, 29]. Тези графики могат да се разширят, например с популярните: гъвкаво моделиране (ГМ) и работната рамка MSF Governance Model [30, 31]. Също така могат да претърпят известно видоизменение в зависимост от категоризацията на бурния период на стандартизиране на обектно-ориентираните подходи за анализ, проектиране и моделиране през 90-те години, известен в литературата като „война на методите“ [32, 33]. Тъй или иначе, масово приложение в софтуерната индустрия са намерили гореизброените модели и методологии. Така, въз основа на представения анализ на развитие на моделите и методологиите за създаване на софтуер, съобразявайки се с определените по-горе критерии, втората софтуерна област, която обособяваме, наричаме: методологии за създаване, моделиране и прилагане на софтуерни процеси.

Управление и реализация на софтуерни проекти

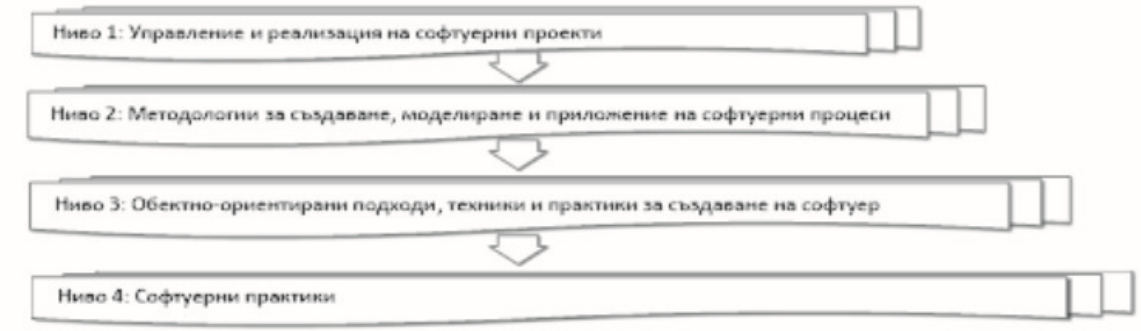
Всяка работа по създаване и разработване на софтуерни приложения започва с планиране на проект, като обикновено разработката се проектира в две перспективи – концептуална и софтуерна, всяка от които има икономически, юридически и мениджърски аспекти. В концептуалната перспектива мениджмънтът засяга въпросите, свързани с поставянето на цели и постигане на резултати, въпроси свързани с бюджета, времето и обхвата на проекта, екипа, който ще го реализира, ресурсите, които ще са необходими, рисковете от провал, възвръщаемост на инвестициите и др. проблеми характерни за изграждането на визия на реализацията. В софтуерна перспектива мениджмънтът е свързан с организацията и управлението, с това как концептуалната перспектива да се реализира като софтуерен процес. При нея се формулират и формализират множество от дейности и практики, определят се роли и отговорности на участниците в проекта с цел да се организира и структурира процесът на създаване на софтуер. При съвременните гъвкави методологии организационните и управленските подходи са стандартизирани. Всяка съвременна методология притежава набор от ръководства за прилагане на контрол и управление на промените, контрол и управление на рисковете, управление на итерациите и т.н. По същество тези подходи притежават мениджърски характер, поради което софтуерният разработчик изпълнява функции на мениджър, т.е. при работа със съвременна гъвкава методология софтуерният разработчик освен специализираните софтуерни компетенции е необходимо да притежава и управленски знания и умения. Гъвкавите методологии обаче не могат да покрият всички знания, умения и компетенции за управлението на софтуерните проекти. В исторически план, много преди да са се развили управленските решения на методологиите, са се утвърдили стандарти за управление и контрол на проекти, по настояще обхващащи и реализирането на проекти в софтуерното инженерство. Такива стандарти са Системата от знания за управление на проекти (СЗУП) и Интегрираният модел за

зрялост на възможностите (ИМЗВ). Стандартите СЗУП и ИМЗВ разполагат с планиране, за каквато пълнота и завършеност гъвкавите методологии не могат да претендират [34]. Така например, не всички съвременни гъвкави методологии покриват пълния жизнен цикъл на създаване на софтуер [28]. Нещо повече, задълбоченият анализ над дейностите за ръководене на софтуерни проекти при по-известните гъвкави методологии ни разкрива, че фокусът при тях е върху приложението на софтуерния проект като софтуерен процес, т.е. фокусът на жизнения цикъл и работен поток е над управлението на софтуерния процес, а не върху управлението на софтуерния проект. Макар в голяма степен да съществува препокриване, това са две различни концепции. Проектът представлява временно начинание за създаването на уникален продукт, услуга или резултат. Временният характер на проектите показва определено начало и край, като краят на проекта настъпва, когато целите на проекта са постигнати или когато проектът е прекратен [35]. При реализирането на по-големи софтуерни проекти е необходимо да се управлява и самият софтуерен проект. Това управление е с по-голям обхват от управлението на процеса и е по-комплексно, тъй като следи по-стриктно параметри като пазарни ниши на приложение, възвръщаемост на инвестициите, административна и правна съгласуваност и пр. норми ведно с технологичния напредък на разработката. Различието между планирането при проектите и процесите може да се изрази чрез съпоставянето на техните концепции за управление, а също така чрез съпоставянето им с концепцията за т. нар. управление на бизнес процеси. Този въпрос е разгледан в изследването [36] като сравнителен анализ между четири гъвкави методологии и два стандарта за управление на проекти. Изследвана е степента на съответствие въз основа на пет променливи величини: обхват, качество, график, бюджет и специфики. Резултатите в проучването сочат, че приблизително 85-95% от аспектите на величините обхват, качество, график, бюджет и специфики при стандартите могат напълно или частично да бъдат покрити от методологиите, като съпоставката е валидна както за всяка една променлива, така и за цялостния жизнен цикъл на проекта. Ясно е, че планирането при стандартите за управление на софтуерните проекти е с по-голям обхват и комплекс от дейности, сравнено с управленските подходи при гъвкавите методологии. Въпреки че разликата не е голяма и в отделни случаи не е съществена, от гледна точка на образователния процес първите притежават съществено предимство изразено с пълна степен на технологична независимост на управление на процеси и дейности. Поради тези причини обособяваме управлението и реализацията на софтуерни проекти като отделна софтуерна област в модела.

Перспективи пред образованието и обучението по информатика и софтуерни технологии

В предходната точка анализирахме обособяването на три софтуерни области. Тяхното обособяване обаче не е в резултат единствено на представения по-горе анализ, а е в следствие на двегодишно задълбочено изследване над проблематиката „Основи на теория на методика на обучението по софтуерни технологии“. Така например в анализа не са включени, но са взети под внимание, мненията, позициите, становищата и препоръките от интервюирането на 30 авторитета за страната, в това число софтуерни разработчици, изследователи и преподаватели. Също така, отчетено е, от една страна, историческото развитие на обучението по информатика и софтуерни технологии в България, а от друга страна, тяхното състояние за учебната 2013/2014г. въз основа на анализ на 484 учебни програми за специалности „Информатика“ и „Софтуерни технологии“ [37]. Както може да се проследи, учебното съдържание на обучението по информатика и софтуерни технологии се изменя почти изцяло през приблизително 10-15 години. Поради тази причина разглеждаме въпроса за перспективата пред обучението по информатика и софтуерни технологии на ниво учебен план, като разкриваме парадигма на софтуерното производство чрез изграждането на модел от софтуерни области (виж фиг. 1). Изграждането на модела обединява научното познание за развитието на софтуерното производство, състоянието на обучението в професионалното направление и специализираната компетентност на разработчици, изследователи и преподаватели, като обособява областите, от които е съставен въз основа на критериите за масовост, трайност и технологична независимост на софтуерни концепции, принципи, подходи, техники, практики и т.н. Предвид, че се цели да се представи образователната перспектива на обучението към трите области в модела, е добавена и четвърта област, наречена „Софтуерни практики“. Четирите области са представени йерархично на нива, разкриващи степени на абстракцията и сложността на разрешаване на софтуерни проблеми. Казахме, че чрез йерархията на областите в модела представяме образователната перспектива на обучението по информатика и софтуерни технологии на ниво образователен учебен план, но трябва да уточним, че в допълнение на представената перспектива е необходимо да се разгледа въпросът за декомпозирането на софтуерните

области от ниво учебен план на ниво учебна програма с цел изготвяне на учебно съдържание. Този проблем развиваме в отделна публикация, тъй като в общия си вид така представеният модел представлява една от компонентите на „Общ йерархичен интеграционен модел за структуриране, разпределяне, формализиране, адаптиране и актуализиране на тематично учебно съдържание, подходи, методи и средства на обучение, разгърнат като технологични области от знания за сферата на софтуерното инженерство“.



Фигура 1. Йерархичен модел на парадигма на разработване, реализация и интегриране на софтуер

Заключение

В работата проследихме развитието на обучението по информатика и софтуерни технологии в България, изследвахме развитието при програмирането, софтуерните модели и методологии и съпоставихме последните със стандартите за управление на проекти. В следствие на задълбочен анализ изградихме модел на парадигма на софтуерното производство, съставен от четири софтуерни области. В заключение, откривайки перспективите пред обучението по информатика и софтуерни технологии чрез представения на Фиг. 1 модел, трябва да отчетем, че към учебната 2013/2014г. във водещите университети на страната не е забелязано да се провежда обучение по управление и реализация на софтуерни проекти, а управленски подходи за реализация на софтуерни проекти при гъвкави методологии на разработване на софтуер се изучават в едва няколко теми на четири дисциплини от общо 484 изследвани учебни програми [37]. Затова повдигаме и оставяме отворени за обсъждане въпросите: Как областите „Управления и реализация на софтуерни проекти“ и „Методологии за създаване, моделиране и приложение на софтуерни процеси“ да се адаптират за целите на образованието и интегрират в учебния план и програма?; С каква интензивност да се изучават?; В какъв етап от курса на обучение?; и пр. предстоящи за разрешаване образователни проблеми на обучението в професионално направление „Информатика и компютърни науки“.

Литература:

1. Манев, К., Н., Манева, Информатика 9. клас, С., Анубис, 2001.
2. Тодоров, Ю., Информатика и количествени методи за историци, С., 2011г., електронно издание: http://www.clio.uni-sofia.bg/todorov/II&C.pdf.
3. Бърнев, П., Г.,Тотков, Р., Донева, В., Шкуртов, К., Гъров, Информатика + 9. клас, Пловдив, Летера, 2001.
4. 60 години БАН-ИМИ основан 1947, С., БАН, 2007.
5. Стоименова, В., Обучението по вероятности и статистика за педагогическите специалности във ФМИ при СУ „Св. Климент Охридски“, Доклади на Четиридесет и втора Пролетна конференция на СМБ, Боровец, 2-6 април, 2013., стр. 437-446.
6. Замфиров, М. Анализ на интересите на кандидат-студентите във ФМИ при СУ „Св. Климент Охридски“ 2005 – 2013 г., сп. Българска Наука, бр. 57, май 2013., стр. 27.
7. Великова, Е., Нишева, М. Обучението във ФМИ – единство на традиция и новаторство, Доклади на XXXVIII Пролетна конференция на СМБ, Боровец, 1-5 април, 2009, стр. 13-22.
8. Sommerville, I., Software Engineering: Seventh Edition,Addison Wesley, 2005, p.26.
9. Naur, P., B. Randell, (Eds.). Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO (1969) pp. 231.

10. Манева, Н., Ескенази, А. Софтуерни технологии, Анубис, София, 2001 г.
11. Laplante, P. What Every Engineer Should Know about Software Engineering, CRC Press, April 25, 2007.
12. Манева, Н. Човешкият фактор в софтуерното производство: спасение или бедствие. [Сборник доклади на Национална конференция „Образованието в информационното общество“], Пловдив, АРИО, 27-28 май 2010, стр. 69-78.
13. Манева, Н. Обучението по софтуерни технологии: очаквания за криза или криза на очакванията [Сборник доклади на Национална конференция „Образованието в информационното общество“], Пловдив, АРИО, 26-27 май 2011, стр. 54-63.
14. Пърсънс, Д. Динамични уеб приложения с XML и Java, Дуо Дизайн, 2010.
15. N"rmarks, K. Functional Programming in Scheme – with web programming examples, Department of Computer Science, Aalborg University, web-textbook, септември 2003, http://people.cs.aau.dk/~normark/prog3-03/pdf/all.pdf.
16. Eckel, B. Thinking in Java 4th ed., US, NJ. Prentice Hall Inc, 2006.
17. Booch, G., R., Maksimchuk, M., Engle, B., Young, J., Conallen, K., Houston, Object-Oriented Analysis and Design with Applications Third Edition, Pearson Education Inc, 2007.
18. Официален сайт: TIOBE Software, http://www.tiobe.com, посл. посещение 10.09.2014г.
19. B. Randell and F.W. Zurcher, „Iterative Multi-Level Modeling: A Methodology for Computer System Design,“ Proc. IFIP, IEEE CS Press, 1968, pp. 867-871.
20. Lehman M.M., Process models, process programs, programming support, Proceedings of the 9th international conference on Software Engineering, Monterey, California, United States, March 1987, p.14-16.
21. Larman, C., V., Basili, Iterative and incremental development: A brief history, IEEE Software, vol. 20 4756, 2003.
22. Royce, W., Managing the Development of Large Software Systems, WESCON ‘70, 1970, reprinted in 9th International Conference on Software Engineering, Washington, D.C.: IEEE Computer Society Press, 1987, pp. 328-338.
23. Zhang, X.,T., Hu, H., Dai, X., Li, 2010. Software Development Methodologies, Trends and Implications: A Testing Centric View. Information Technology Journal, vol. 9 (8), 1747-1753.
24. McKeen, J.D. 1983. Successful Development Strategies for Business Application Systems, MIS Quarterly, vol. 7, No. 3, 47-65.
25. Robey, D., R., Welke, D., Turk, Traditional, iterative, and component-based development: A social analysis of software development paradigms. Inform. Technol. Manage., 2001., 2: 53-70.
26. Sorensen, R., A Comparison of Software Development Methodologies Reed, Software Technology Support Center, 1995.
27. Официален сайт на MASD, http://agilemanifesto.org/, посл. посещение на 14.04.2012.
28. Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods review and analysis. VTT Publications(478), 3-107.
29. Кръстева, И., Гъвкава методология за разработка на софтуерни приложения. [Автореферат на дисертационен труд за присъждане на образователна и научна степен „доктор“, научна специалност 01.01.12.“Информатика“], София, 2011.
30. Официален сайт на: Microsoft Developer Network, https://msdn.microsoft.com/en-us/library/jj161047.aspx, посл. посещение на 7.05.2014г.
31. Официален сайт на: Microsoft MSDN for Visual Studio, http://msdn.microsoft.com/library/vstudio/dd380647.aspx, посл. посещение на 07.05.2014г.
32. Фаулър, М., UML основи, София: СофтПрес ООД, 2004, стр.35-36.
33. Palmer, P., Barker, W. Beginning C# 2008 Objects: From Concept to Code, Apress, 2008, p.241
34. Fitsilis, P., Comparing PMBOK and Agile Project Management Software Development Processes, Advances in Computer and Information Sciences and Engineering, Springer Science Business Media B.V., ISBN 9781-4020-8740-0, 2008.
35. Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 4th Edition, ISBN 978-1-933890-51-7, 2008.
36. Тодоров, Н. Гъвкави методологии за разработка на софтуерни проекти и тяхното приложение, основано на утвърдени стандарти за управление и качество, Автореферат на дисертационен труд

за присъждане на образователна и научна степен „доктор“, научна специалност 01.01.12.“Информатика“, БАН-ИМИ, София, 2013.
37. Христов, Хр., Крушков, Хр., Състояние на обучението по информатика и софтуерни технологии, списание „Образование и технологии“, бр. 6, 2015.