АРХИТЕКТУРА „МОДЕЛ-ИЗГЛЕД-КОНТРОЛЕР“ В ПОМОЩ НА ПРЕПОДАВАНЕТО НА УЕБТЕХНОЛОГИИ[1]

Христо Христов, Христо Крушков

Резюме. През изминалите три десетилетия и половина архитектурата MVC доказа своята полезност в софтуерната индустрия. Широката употреба на MVC е предпоставка за нейното преподаване, изучаване и ефективно прилагане в разработването на уебприложения.

В работата се анализират трудностите, които срещат преподаватели, студенти и ученици при преподаването и изучаването на архитектурата MVC. Предлага се подход на преподаване, като се представят и решения за справяне с трудностите. Подходът разглежда процеса на обучение, като разгръща преподаването на MVC в посока от абстракцията на модели към реализация на изходен код, акцентирайки върху завършеността на софтуерната разработка. Описани са резултатите от проведените педагогически експерименти с ученици и студенти.

Keywords: model-view-controller, teaching methodology, web development

1. Въведение

Архитектурата MVC (model-view-controller, модел-изглед-контролер) е популярен шаблон за дизайн, описан официално за пръв път като част от езика за програмиране SmallTalk (Пърсънс, 2011). Нейният създател от своя уебсайт[1] посочва: „Изобретих MVC като очевидно решение на общ проблем за предоставяне на потребителски контрол над информацията, когато тя се разглежда от различни перспективи“. MVC се посочва като основа на архитектурни решения в няколко работни рамки (frameworks) за създаване на уебприложения, като ASP.Net, Rails, Struts и др. (Grove & Ozkan, 2011). През изминалите три десетилетия и половина от своето създаване MVC набира все по-широка популярност благодарение на приложността, която архитектурата намира при реализацията на софтуерни продукти и услуги. Фактически MVC се е превърнала в стандарт и е станала част от практиките за разработване на софтуер на почти всички софтуерни методологии. Навярно няма уебпрограмист, който да не е имплементирал архитектура MVC. Потребителите на интернет пространството също ежедневно срещат реализации на MVC, когато използват уебсайтове за електронно разплащане, пазаруват онлайн или

се забавляват в реално време с уеббазирани игрови стратегии. Архитектурата MVC е доказала своята полезност в индустрията, но също така може да бъде ефективно използвана и при учебни проекти, които използват както традиционни, така и обектноориентирани методологии (Zant, 2006). Широката й употреба е предпоставка за нейното преподаване и изучаване като част от софтуерния процес при създаване на уебприложения или в частност като дейност за проектиране и имплементиране на дизайни. Такива разработки се изучават в курсовете по уебпрограмиране. Преподаването им не е лека задача и затова, може би с някои изключения, в училищата не се среща подобна тематика. В университетите, за разлика от училището, съществуват курсове, които разглеждат проблематиката, но препятствия не липсват. За да се запълни част от този дефицит, на вниманието на читателя се предлага подход на преподаване, който да е в помощ на преподавателската практика.

В работата се анализират трудности, които срещат преподаватели, студенти и ученици при изучаването на архитектурата MVC. Представя се подход на преподаване, като се дават и решения за справяне с трудностите. Подходът разглежда процеса на обучение, като разгръща преподаването на MVC в посока от абстракцията на модели към реализация на изходен код. Той представлява част от методика на преподаване, в частност на архитектурата MVC, базирана на концепция, чиято основна идея е реализацията на абстрактните модели до функционалност, т.е. като завършен софтуерен продукт или услуга. В статията също така паралелно са представени *методическите аспекти на преподаване,* касаещи дейността на преподавателя и *практиката на разработване на софтуер*, което е предмет на взаимна работа между преподавател и студент. Методическите аспекти обхващат структурата и съдържанието на учебните материали, тяхната организация и последователност на излагане. Акцентът при разработването на софтуер е върху пълнотата на реализацията, т.е. резултатът от работата на студента да е завършено софтуерно приложение.

2. Фактори, които налагат допълнителни съображения в методиката на преподаване на софтуерни технологии

Преподаването на MVC като организация на работа принципно не се различава от разработването на софтуер, базиран на архитектурата. В известен смисъл е допустимо да се прави аналогия между двата процеса: на преподаване и на разработване. Разликата между *подхода на преподаване на MVC* и *професионалната реализация на архитектурата* е в обхвата и мащаба на разработките. На практика преподавателят трябва да използва дидактически материали, които са част от реализацията на архитектурата. Нещо повече, на него му се налага така да подготви материалите, че в курса на обучение те да се използват като помощни образци и илюстрации, а не да се сглобяват „наготово“ реализирани части. При предос-

тавянето на готови материали ключово за обучението е да се даде приоритет на творческото мислене, на самостоятелната работа на обучаващия се да проектира и програмира, като същевременно са подготвени помощни модели на работа. Тези обстоятелства налагат преподавателят да извърши адаптация на технологиите за професионално разработване на софтуер с характера на обучение. Такава предварителна подготовка предполага към вече общоутвърдени норми на преподаване да се добавят допълнителни методически изисквания. От особено значение е фактът, че обучаемият е новак по отношение на много от технологиите, които използва. Той обикновено не познава техните принципи и характеристики на работа. Това особено важи за преподаването на архитектурата MVC. Освен вземането на мерки за преодоляване на технологичната бариера от обучаемия, преподавателят също така е необходимо да се съобрази с входното ниво, с често срещаната чувствителна разлика в познанията и способностите на отделните ученици и студенти, със средата, в която се провежда обучението, и обичайните административни спънки при настройката и конфигурацията на компютърните системи, с мотивацията, възрастовата рамка, броя часове и др. разнородни по характер причини, които е наложително предварително да се съгласуват с протичането на учебния процес. Всички тези причини и обстоятелства могат да се обединят и класифицират в няколко фактора, а именно: *социален фактор* – средата, формата и начина, по който се извършва обучението; *технически фактор* – технологиите, като средство и инструмент за работа; *професионален фактор* – техническата и педагогическата компетенция на преподавателя и входното ниво знания на обучаемия.

Първият фактор е чест обект на внимание от страна на методолози и институционални ръководители, но той пък, като политика и институционална стандартизация, е локализиран само в подобряването на учебните бази и техническата инфраструктура. При него приоритетно се разглеждат осигуряването на среда за работа, докато формата и начинът на обучение се приемат за даденост, оставят се на педагогическите компетенции на преподавателя или се следва инерцията на остарели методически подходи. Другите два фактора, за разлика от социалния, тепърва намират своето място в методиката на преподаване. При изучаването на подходи за реализация на софтуерни проекти, визирайки бурното развитие на ИТ индустрията през последните 10 – 15 години, липсват методики на преподаване. Това е обяснимо, като се вземе предвид както развитието на индустрията, така и обстоятелството, че на преподавателите в областта на компютърните науки се налага през няколко години да преподават напълно нови и непознати технологии. Професионализмът на педагога информатик изисква широки познания и висота на преподаване при използването на дидактически материали. Едно е, когато дидактическият материал е маркер и бяла дъска, съвсем различна е ситуацията, когато това

са модели и изходен код, прилаган чрез инструменти за разработка на софтуер, при това от обучаем, а не само от преподавател. За съжаление, образователната общност все още не е намерила единна концепция по въпросите, свързани с методиката на преподаване на съвременни технологии. Дефицитът на подобна концепция е толкова сериозен, че специалисти и образователни експерти дори при съставяне на стандарти някак остават поставени във вакуума на ежедневието и дискутират „Какво да се изучава?“, при което не се търси смислена интерпретация на въпроса „Как да се учи?“. Проблемът обаче остава за решаване пред преподаватели и учители по информатика, и то не само в рамките на учебния час, но и на ниво държавни образователни стандарти, тъй като съвременното обучение по програмиране няма как да е изолирано от интернет пространството, а и не бива от него да се изключват знанията и уменията за програмиране на мобилните компютри, таблети, смартфони и т.н. Изходната позиция, от която да се даде отговор на двата въпроса, поставени в логичната си последователност, е естествено определима, когато се види какво е неизбежното потребление на средства и технологии в обозримо бъдеще. Такъв е и случаят с архитектурата MVC. Независимо от иновативните решения в близко бъдеще MVC ще е фундамент при разработването на уебприложения.

3. Специфика на методиката на преподаване на MVC

3.1. Нивата на абстракция.

През последните години сложността на разработване на софтуер предполага началната работа върху архитектурата и дизайна на приложенията да започва с *високо ниво на абстракция* и *в независимост на компонентите*, които впоследствие се програмират. Тези два признака са характерни при преподаването и изучаването на архитектурата MVC. Въпреки че са разработвани и много по-сложни архитектури, MVC е интересна за изучаване, защото нейната простота я прави приемлива за практикуващите, а освен това тя е много добре позната в индустрията (Tao et al., 2010). Както се разбира от името, MVC се състои от три компонента: *модел, изглед и контролер. Моделът* е обектът (компонентът) на приложението, *изгледът* е неговото представяне на компютърен екран, а *контролерът* е дефинирането на начина, по който потребителският интерфейс реагира на действията на потребителя (Гама, 2006). Гледната точка при тази дефиниция представя MVC като модел, който е с най-високо ниво на абстракция в рамките на архитектурата на едно уебприложение. Представянето на MVC от подобна перспектива в известен смисъл е непълно и погледнато от определен ъгъл създава неточно разбиране.

Архитектурните модели често се прилагат в комбинация с други шаблони. Взаимовръзката между тях трябва да се взема предвид, когато се прилага комбинирането им в система. Затова ефективното интегриране на архитектурни

модели в софтуера е предизвикателство. Така е, защото интегрирането на всеки два архитектурни модела може да приема няколко различни форми (Kamal et al., 2011). Тези виждания споделя и създателят на MVC, развивайки началната си идея върху архитектурата, като й придава допълнителна интерактивна характеристика (Reenskaug & Coplien, 2009). Разграничавайки се от повечето платформи, предоставящи реализации на MVC като синхронизация на състоянията на модела, изгледа и контролера, авторът посочва, че всеки от компонентите всъщност представлява роля, която се изпълнява от обекти, „генерирани“ от потребителя. В продължение на тези разсъждения в източника се твърди, че за да се разбере по-детайлно перспективата на архитектурата, която цели да отдели представянето на информацията от взаимодействието с потребителя, е необходимо в MVC да се разглежда и ролята на потребителя. Тази перспектива той нарича *модел-изглед-контролер-потребител* (MVC-U, model-view-controller-user). Според Grove & Ozkan от 2011, в резултат на еволюцията на архитектурата може да се говори за MVC-Web модел. Източникът посочва четири отговорности на модела MVC-Web: устойчивост на данните; управление на транзакциите; интерактивно взаимодействие с потребителски интерфейс; обработване на заявки. От казаното става ясно, че на MVC архитектурата се гледа с различни нива на абстракция, а нейните реализации варират от готови платформени решения, които имплементират шаблони за дизайн, до сложни архитектури, които реализират комплексни интерактивни взаимодействия между потребител и компютър. Логично възниква въпросът как преподавателят да представи нивата на абстракция на архитектурата пред обучаем с минимален технически опит. Отговорът на поставения въпрос в предлагания подход е последователно изложен.

3.2. Справяне с абстракцията.

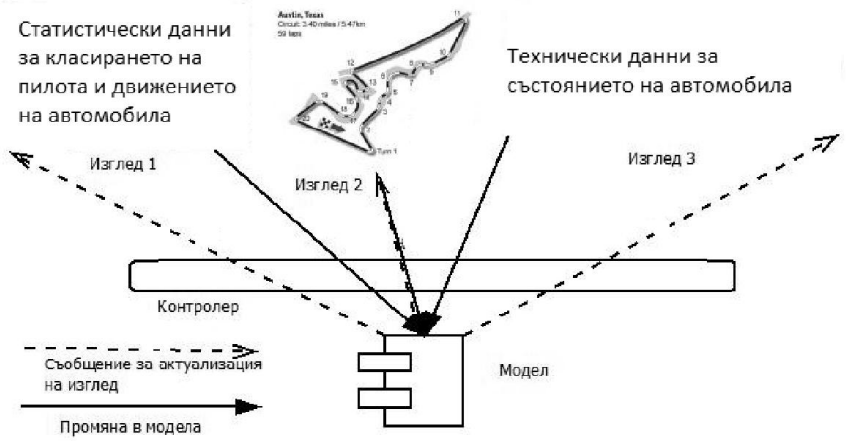
Първоначално в подхода се разглежда „Симулация на реална ситуация“, която е добре позната и интересна за ученика и студента. Всеки модел, независимо от нивото си на абстракция, е лесен за възприемане чрез подходящ пример, който е представен удачно. В нашия случай разглеждаме мениджърски контролен пункт на пилот от Формула 1. Темата е подбрана така, че да предразполага обучавания към провеждане на разговор.

Следва описание на примерна симулация, посредством която се представя идеята на архитектурата MVC: за да изпълнят стратегиите си по време на състезание, мениджърите на състезателните пилоти следят няколко компютърни екрана с информация, чиято актуалност се променя постоянно. На фигура № 1 са представени три примерни перспективи. *Изглед 1* – статистически данни за класиране на пилота, средна скорост на автомобила, данни за последна обиколка и т.н.;

Изглед 2 – местоположението на автомобила върху пистата; *Изглед 3* – технически параметри за температура на спирачната система, количеството на горивото, атмосферни условия и др.

Обсъждане на подобни примери в час носи педагогически ползи, тъй като между обучаващ и обучаван се създава атмосфера за провеждане на беседа, в резултат на която учениците и студентите правят предложения, допълват изложената информация, съпоставят няколко различни гледни точки за визуализация на изгледите, класифицират информацията според собствената си обща култура, а впоследствие това води и до по-задълбочено разбиране при представянето на модели, структури, типове данни и т.н.

Втори ключов момент при преподаването на MVC е конкретизацията на идеята в контекста на концепциите на обектноориентираното програмиране и модела „клиент-сървър“, независимо от технологиите. За тази цел първоначално е нужно да се представи принципът на клиент-сървър приложенията посредством някоя схема. Впоследствие трябва да се обърне внимание на изгледа – като програмиране от страна на клиента, на контролера – като програмиране от страна на сървъра, и на модела – като проектиране на класове, структури и/или база от данни. Например за частния случай, който разгледахме като симулация на този етап от преподаването на MVC, е подходящо: първо, да се проектират три модела (blueprint) на обекти (чрез структурата клас в смисъла на обектноориентирано проектиране) – по един за всеки от изгледите 1, 2 и 3. Впоследствие класовете да се развият чрез различни техники на обектноориентираното програмиране, да се препроектират или да отпаднат, тъй като не са били подходящо подбрани. Прецизността на резултата не



Фиг. №1 Симулация на ситуация

е от съществено значение. По-важно на този етап е да се обясни, че тези обекти съставят модела на архитектурата; второ, да се даде за самостоятелна или домашна работа на обучавания да разсъждава как според него е добре да изглеждат данните за всеки един от обектите на модела върху потребителския интерфейс, а впоследствие тези изгледи да се мултиплицират в различни екранни форми на визуализация, да се обогатят и стилизират. Независимо че учещият се все още няма да познава определена технология за изобразяване (rendering) на информация върху уеббраузъра, той ще придобие представа как според него е добре да изглеждат обектите на екрана. Третият компонент – контролерът, е най-сложен за реализация и на този етап е достатъчно да се обясни, че следва да се изучава като програмиране от страна на сървъра, след като се добие опит с програмирането от страна на клиента.

Следващ ключов етап в преподаването на архитектурата, след първоначалното запознаване с идеята и нейната конкретизация в контекста на обектноориентираното програмиране и модела „клиент-сървър", е *„превеждането"* на примерите, които се разглеждат в инфраструктурата на интернет. На този етап е добре да се направят аналогии между примерите и реални системи, базирани на MVC, като например може по-обстойно да се разгледа в часа някой функциониращ онлайн магазин за покупки или система за електронно разплащане. При един онлайн магазин браузърът на всеки клиент представлява индивидуален *изглед* на данните (например отделните потребители могат да разглеждат различни каталози на магазина); сървърният софтуер на приложението, който се грижи за комуникацията, е компонентът *контролер*, а *моделът* – представянето, обработването и съхраняването на продуктите в приложението, т.е. негова база с данни. Независимо от подбраните примери обаче съществените за подхода е да се снижават стъпаловидно нивата на абстракция.

3.3. Технологии за реализация на MVC.

Следващата методическа стъпка на преподавателя е да запознае обучаващите се с технологиите за реализация на MVC. Тъй като реализацията на архитектурата е предвидена да работи в интернет, такива технологии са част от инфраструктурата на интернет и средствата за програмиране в уебпространството. Това включва изучаването на протоколи, описателни езици, програмни езици с общо предназначение, работа с интегрирани среди за разработка, сървърни компоненти - тяхната настройка и конфигурация и т.н. Според Grove от 2007 броят и разнообразието на протоколи и програмни езици, необходими за разработване на уебприложения, представлява проблем. Студентите обикновено се запознават с няколко уебпротокола (TCP/IP, HTTP или SSL/TLS), също така с няколко описателни езика (като

HTML, CSS, JavaScript, XML), един или два езика за програмиране от страна на сървъра (като Java, C#, PHP, Perl и др.), език за работа с база от данни (като SQL). На тях им се налага да научат как да конфигурират сървърна компонента (като Apache, JBoss, Glassfish, ASP/.Net и т.н.), да управляват операционната система и пр. При преподаването на толкова технологии трудностите са неизбежни и за опитен специалист. Как тогава обучаемият да придобива увереност, че може да се справи с технологиите, без да губи интерес? До каква степен да се изучава всяка от технологиите?

Отговорите на поставените въпроси не са еднозначни. Те зависят както от професионалната, техническата и методическата компетенция на преподавателя, така и от мотивацията на студента. Една част на отговорите се крие в начина на запознаване с интернет инфраструктурата: протоколи, сървъри, програмиране от страната на клиента, програмиране от страна на сървъра и т.н. Това, от своя страна, предполага наличието на допълнителни дидактически материали. Например инструкции или видеоуроци за инсталиране и настройка на интегрирана среда и конфигуриране на сървър. За представянето на работата на уебпротоколи пък е удачно да се използват нагледни схематични модели и стимулационни системи. Също така преподаването на принципите на работа на протоколите е добре да се съчетава с демонстрация на реализации посредством определена технология. Например да се покаже обработка на заявка „get“, предадена по http-протокол от JSP технология на Apache Tomcat сървър. Освен това необходимо е да се представят технологиите, с които ще се създава уебприложението, и да се посочи минимумът нужни познания за работа с тях. Друга част от отговорите на поставените въпроси се отнася до изучаването на езиците: описателни, с общо предназначение, езици за обработка на заявки. При тях „златно“ правило е всяка технология да се преподава толкова, колкото е нуждата за разработване на реализацията. Тъй като подобна среща с програмните езици не следва обичайна последователност на преподаване, то от особено значение е подготовката на предварителни примери, които да послужат за основа и образец на работа.

3.4. Контекст на софтуерна разработка.

Училищната информатика и университетските дисциплини, обхващащи обучението по програмиране и разработването на софтуер, за съжаление все още в голяма степен са ограничени от дейности, пряко свързани с програмиране. Особено в училищната информатика създаването на софтуер изцяло е сведено до изучаване на език и среда за програмиране. Такава тясно технологична зависимост на преподаване лишава обучението от *контекст*.

Какво е контекст на софтуерна разработка?

Както при строителната архитектура е необходимо проектиране на основа и носеща конструкция, за да се изградят останалите елементи на една сграда, така и при създаването на софтуер е нужна *архитектурна концепция*. Софтуерните проекти се различават много един от друг. Начинът, по който се подхожда към разработката на софтуер, зависи от вида на системата, използваната технология, разпределянето на екипа, естеството на рисковете и др. фактори (Фаулър, 2004), които е необходимо да се разгледат в обучението на софтуерния разработчик или най-малко да се споменат в учебните часове, за да може учениците и студентите да придобиват по-широк поглед за професионалната работа. Също така на вниманието на разработчика са: планирането на бюджета, времето за реализация, разпределянето на роли, отговорности и задачи в екипа, събирането на информация за нуждите на потребителите от даден софтуер и пр. Според Haumer от 2007 един от двата ключови проблема при реализацията на софтуерен процес е нуждата на разработчиците да разберат методите и ключовите практики на софтуерната разработка. Те трябва да са запознати с основните задачи на разработката – как да извличат и управляват изискванията, как да правят анализи и дизайн, как да реализират дизайн, как да управляват обхвата и промените в проекта и т.н. Именно тези дейности, които касаят съставянето на *архитектурната концепция* като база, въз основа на която да се конструира архитектура, проектира дизайн и имплементира изходен програмен код, наричаме *контекст на софтуерната разработка.*

Разбираемо за всеки специалист от областта е, че обхватът на контекста на софтуерната разработка може да се обособи както в самостоятелна дисциплина (или нейна подобласт), така и да се разпредели в профила на учебен план, като знанията се усвояват последователно и поетапно в няколко поредни учебни години. Този съществен въпрос, касаещ методите и средствата в обучението по информатика, налага създаването на нови подходи на преподаване.

4. Контекст на софтуерната разработка при подхода на преподаване на архитектурата MVC

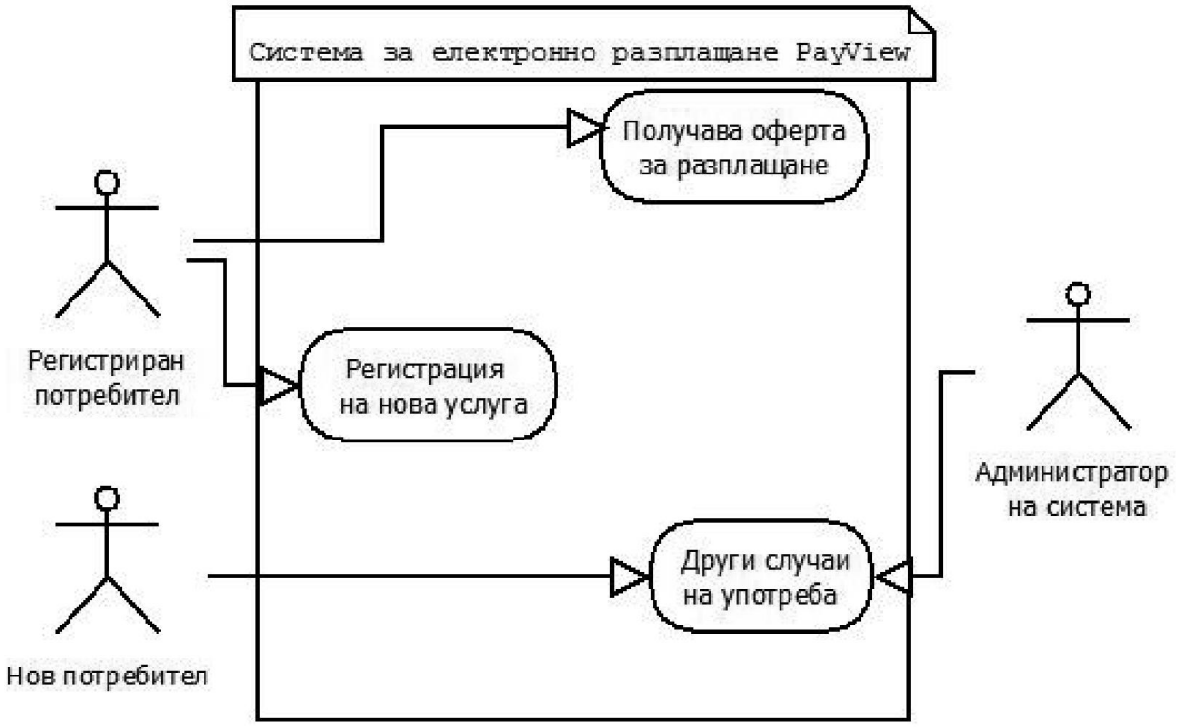
В нашия случай разглеждаме подхода на преподаване, ограничавайки се от хорариум с продължителност четиридесет учебни часа. Опитът, от който изхождаме, са проведените експерименти в съботната школа „Проектиране и изграждане на софтуерни приложения“ в ОМГ „Акад. Кирил Попов“ – Пловдив, и свободно избираемата дисциплина „Анализ, дизайн и изграждане на софтуерни приложения“ във ФМИ на ПУ „Паисий Хилендарски“. При провеждане на занятията се възприе контекст на разработката да е: *общата визия* за изграждане на уебприложение за електронно разплащане, която да се използва за база при *моделиране на случаи на употреба,* като в частност детайлно се изучат проектирането, реализирането и

тестването на най-често използваните *случаи на употреба*. За целите на учебния процес се приеха дефиниции и стереотипът на представяне на отделните елементи, използвани в Bittner & Spance от 2002 и Cockburn от 2001. Важно е да се отбележи, че както визията, така и други дейности по съставяне на терминологичен речник, изграждането на модел на случаите на употреба, описване на случай на употреба, построяване на уебпроцес за реализация на случай на употреба и др., се извършваха в условията на беседване между преподавателя и учениците (студентите).

Следва представянето на визията, модел на случаите на употреба и един конкретен случай на употреба.

А. Визия: Достъп до уеббазирана система за електронно разплащане от всяка точка на интернет пространството, чрез която да се предоставят автоматизирани услуги: за регистрация на нов потребител, регистрация на битова сметка за разплащане, плащане на задължение, извличане на справки за минали плащания и др.

В. Модел на случаите на употреба:



Фиг. №2 Диаграма на модел на случаите на употреба

С. Случай на употреба:

– *Име на случай на употреба:* Плащане на битова сметка.

– *Актьор:* Регистриран потребител.

– *Начална страница:* Плащане на сметка.

– *Описание на случая:*

1) Актьорът зарежда началната страница на система за плащане на сметки.

2) Системата показва списък с опции.

3) Актьорът избира опция за разплащане на битови сметки.

4) Системата генерира списък със сметки и задължения.

5) Актьорът маркира задълженията, които желае да изплати.

6) Системата предоставя списък със сметки за разплащане, за които актьорът да потвърди плащането.

7) Актьорът потвърждава разплащането.

8) Системата извършва разплащането.

9) Системата генерира отчет за извършеното плащане.

10) Актьорът излиза от системата.

Сценарий на употреба: Алтернативен път при „т.2)“

a) Актьорът иска от системата списък с битовите сметки, които може да регистрира.

b) Системата генерира списък с битови сметки.

c) Актьорът избира регистрация на битова сметка „разплащане на електрическа енергия“.

d) Системата генерира формуляр за регистрация.

e) Актьорът попълва необходимите идентификационни данни.

f) Системата извършва регистрация на битова сметка.

5. Стъпки за реализиране на архитектура MVC в учебна среда

Както при разработването на софтуер се разглеждат отделните дейности на разработчика като роли в различни етапи на работа, също така при преподаването на MVC е удачно да се обособят различни стъпки, които, обединени, да представляват цялостната работа по реализирането на архитектурата. Практиката при съвременните разработки на софтуер показва, че такива стъпки имат итеративен характер. При изграждането им от значение е спазването на принципи, които според Grove от 2007 са: технологията следва концепциите; необходима е практика; архитектурата е от фундаментално значение; сигурността е от съществено значение. Разглежданият подход на преподаване на MVC се опира на споменатата практика

и принципи. Нещо повече, считаме, че без тяхното възприемане не би могло да се вникне в пълнотата на решенията.

Стъпки за реализация на MVC в учебна среда:

1) Конструиране на архитектура.
2) Запознаване с технологии за анализ, проектиране и програмиране.
3) Анализиране и приоритетизиране на изискванията към приложението.
4) Съставяне на териториален модел и случаи на употреба.
5) Проектиране на уебпроцес за конкретен случай на употреба.
6) Имплементиране на уебпроцес.
7) Интегриране на компоненти на уебпроцес.
8) Публикуване и тестване на функционалност.
9) Повторение на стъпки 3), 4), 5), 6) , 7) и 8) с цел отстраняване на грешки и оптимизация.

6. Оценяване на разработките на обучаемите и анализ на резултатите от проведените експерименти

Експерименти за преподаване на уебтехнологии посредством архитектурата MVC са проведени със студенти и ученици от Пловдив. Анализирани са резултатите от работата с учениците от специалност „Системен програмист“ на ОМГ „Акад. Кирил Попов“ и студентите от ФМИ на ПУ „Паисий Хилендарски“, записали свободноизбираема дисциплина „Анализ, проектиране и изграждане на софтуерни приложения“. Експериментът с учениците се проведе през първия срок, а със студентите – през зимния (Б) триместър на учебната 2012/2013 г. Конкретните технологии, използвани за обучение при провеждане на експеримента, са:

– Интегрирани среди за работа – NetBeans IDE 7.21 и Notepad++;
– Платформа и език за системно програмиране – Java EE 1.7;
– Сървъри за работа в уебсреда – Apache Tomcat 7 и GlassFish 3.1 (Open Source);
– Сървър за съхранение на данни – MySQL Database 5.6;
– Технология за изграждане на потребителски интерфейс – HTML5;
– Технологии за сървърна обработка на данни – JSP и JavaBean.

Занятията в ОМГ се организираха като извънкласна форма на обучение. В школата по „Проектиране и програмиране“ се записаха 16 десетокласници, като 8 от тях завършиха успешно курса. Условие за поставяне на оценка, с което и успешно да се завърши обучението, бе изискването да се разработи реализация на архитектурата MVC като клиент-сървър приложение, което да представлява завършен и функциониращ софтуерен модул на конкретен случай на употреба. Крайната оценка на завършения софтуерен модул е сформирана от средното

аритметично на сумата от оценки на отделни софтуерни елементи, като предварително се изискваше всяка разработка да е съставена от минимум три елемента: два изгледа – форма за регистрация и отговор на клиентска заявка, JSP контролер за обработка и отговор на заявки и JavaBean модел за обработка на данни. Работите се оцениха индивидуално, като на учениците допълнително бе предоставен списък с изисквания и уточнения за функционалността на софтуерните елементи, които разработките да притежават. От 8-те деца, завършили обучението, бяха създадени 8 работи – 2 завършени с отличие, и 6 оценени с много добър. Резултатите от работата с учениците, отчитайки нивото на сложност на разработките, постигнатото качество на реализирана функционалност на софтуерни модули и 50% успеваемост на завършване на обучението, са обнадеждаващи за изучаването на уебтехнологии в средното училище и обещаващи за използването и приспособяването на подхода на преподаване на MVC в учебните програми по информатика, предназначени за учениците с повишен интерес към програмирането.

За разлика от учениците при студентите и формата, и начинът на оценяване бяха различни. При учениците се оценяваше самостоятелната разработка на всеки поотделно, като – колкото беше броят на учениците, толкова бяха и разработките, докато при студентите се оценяваше индивидуалната работа върху част от разработката, която се създава от екип, съставен от 3-ма или 4-ма участници. За да се постигне обективност на оценяването, на всеки екип се предостави схема с разпределение на *роли* с конкретно разписани *задачи* и *отговорности* на участниците. Обективността на оценяване на всеки отделен член на екипа се гарантира от подробно разписана в табличен вид схема, съобразно която за всеки студент е описано какви роли е изпълнявал и съответно като изпълнител на роля с каква отговорност какви задачи е разрешил (или % от тях). Тук отново оценката на студента се формира като средно аритметично, но в случая сумата се образува от разрешените задачи, съответни на отговорностите в екипа. При тази форма на оценяване интересно се оказа, че предоставената инициатива на студентите самостоятелно да разпределят ролите помежду си и да определят процентното участие за отделните задачи в различните етапи от дейността стимулира тяхната работа и допълнително мотивира екипите да доразвият курсовите си проекти. Статистическите резултати при така предложения модел на оценяване за поток от 30 студенти, общо 8 екипа, са: ~66.7 % отличени оценки, ~16.6% много добър; ~16.6% добър. По-интересно от статистическите числа обаче са постигнатите резултати по отношение на обема и завършеността на работите. По отношение на обема на работите почти всички от проектите няколкократно надвишиха първоначално зададените критерии за обем, а по отношение на завършеността – 7 от

8-те екипа покриха критериите за завършена функционалност. Анализът на експеримента показва, че предоставянето на самостоятелност за екипна работа при ясна концепция – какво и как да се разработва – и готови образци на работа при по-сложни техники на програмиране стимулира студентите да усъвършенстват и доразвиват своите работи над първоначално предвидените критерии за завършен курсов проект, което само по себе си говори, че те придобиват знания и умения, надминаващи предвиденото за усвояване от учебната програма.

7. Заключение.

Със статията се представи подход на преподаване на архитектурата MVC, чиято концепция разглежда разгръщането на обучението в посока от абстракция на модели към реализация на изходен код. В подхода се разгледаха факторите, които налагат съображения и съгласуваност на методиката с допълнителни изисквания при изучаването на MVC, засегнаха се методическите аспекти за снижаване на нивата на абстракция, посочи се ролята на технологиите при създаване на уебприложения, даде се описание и дефиниция на понятието *контекст на софтуерна разработка* и се маркираха основни стъпки за работа на студентите. Изводът от проведените експерименти е, че предлаганият подход може да се прилага в обучението, като по-удачният вариант е прилагането на схема с разпределение на *роли* с конкретно разписани *задачи* и *отговорности* на участниците.

БЛАГОДАРНОСТИ

Статията е частично финансирана по научен проект НИ13 ФМИ-002 (2013 – 2014), тема: „Интеграция на ИТ в научните изследвания по математика, информатика и педагогика на обучението“.

БЕЛЕЖКИ

1. Официален сайт, Welcome to the pages of Trygve M. H. Reenskaug, http://heim.ifi.uio.no/~trygver/index.html, последно посещение на 17.01.2013

ЛИТЕРАТУРА

Гама, Е., Р. Хелм, Р. Джонсън, Д. Влисидес (2006). *Шаблони за дизайн*, София: СофтПрес ООД.

Пърсънс, Д. (2011). *Динамични уебприложения с XML и Java.* София: Дуо Дизайн ООД.

Фаулър, М. (2004). *UML основи*, София: СофтПрес ООД.

Bittner, K., I. Spence (2002). *Use Case Modeling*. Boston: Addison Wesley.

Cockburn, A. (2001). *Writing Effective Use Cases.* Boston: Addison Wesley.

Grove, R. F. (2007). Trends in Teaching Web-Based Development: A Survey of Pedagogy in Web Development Courses. (pp. 361 – 365). In: Cordeiro, J., J. Filipe, B. Encarnação, V. Pedrosa (Eds.). *Proceedings of the Third International Conference on Web Information Systems and Technologies*. Barcelona, Spain. INSTICC Press.

Grove, R. F. & Ozkan, E. (2011). The MVC-Web Design Pattern. (pp.127 – 130). In: Cordeiro, J. & Filipe, J. (Eds.). *Proceedings of the 7th International Conference on Web Information Systems and Technologies.* Noordwijkerhout, The Netherlands. SciTe Press.

Haumer, P. (2007). Eclipse Process Framework Composer – Part 1: Key Concepts, Eclipse Process Framework Project,
http://www.eclipse.org/epf/general/EPFComposerOverviewPart1.pdf,
последно посещение на 03.04.2012.

Kamal, A. W., P. Avgeriou, U. Zdun (2011). The use of pattern participants relationships for integrating patterns: A controlled experiment. *Software – Practice and Experience*, doi: 10.1002/spe.1121.

Reenskaug, T., J. O. Coplien (2009). The DCI Architecture: A New Vision of Object-Oriented Programming. Artima, Inc.
http://www.artima.com/articles/dci_vision.html, последно посещение на 17.01.2013

Tao, P., Sun, L., Bao, H. (2010), Design and implementation of ATM simulation system based on MVC Pattern. *Proceedings of the International Conference on Educational and Information Technology*, 1, 328 – 331.

Zant, R. F. (2006). Model-View-Controller architecture in a systems analysis and design course. In *Proceedings of the Information Systems Education Conference*, 23, §3353. ISSN: 1542-7382.

Христо Христов
✉ Докторант
Катедра „Методика на обучението по математика, информатика и ИТ“
ФМИ при ПУ „Паисий Хилендарски“
бул. „България“ 236
Пловдив, п.к. 4003
E-mail: hristo.toshkov@gmail.com

Христо Крушков
✉ доцент, доктор
Катедра „Компютърна информатика“
ФМИ при ПУ „Паисий Хилендарски“
бул. „България“ 236
Пловдив, п.к. 4003,
E-mail: hdk@uni-plovdiv.bg

MODEL-VIEW-CONTROLLER ARCHITECTURE HELPING IN TEACHING OF WEB TECHNOLOGIES

Abstract. During the last three and a half decades the MVC architecture has proven its usefulness in the software industry. The wide usage of MVC makes a strong case for its teaching, learning and effective application in the development of web applications by pupils and students.

In the paper the difficulties that professors, students and pupils face when teaching and learning about the MVC architecture are analysed. A new way of approaching teaching is suggested as well as solutions to cope with the difficulties. The new approach considers the process of education by further developing the teaching of MVC in a certain direction: from the abstraction of models to the realisation of source code by putting the accent on the completion of the software product.

Hristo Hristov
✉ PhD student
Department of Methodology of Mathematics, Informatics and IT Education
FMI, University of Plovdiv
236 Bulgaria str.
Plovdiv
E-mail: hristo.toshkov@gmail.com

Hristo Krushkov
✉ Associate professor, PhD
Department of Computer Science
FMI, University of Plovdiv
236 Bulgaria str.
Plovdiv
E-Mail: hdk@uni-plovdiv.bg

ПЪРВИЯТ БЪЛГАРСКИ УЧЕБНИК ПО ТЕОРИЯ НА ВЕРОЯТНОСТИТЕ

Пламен Матеев

Резюме. Целта на статията е да запознае читателя с първия учебник по теория на вероятностите за средното училище на български език, издаден през 1890 г. Представено е накратко съдържанието му, което може да послужи за сравнение със сегашното състояние на образованието по стохастика у нас.

Keywords: statistical education, history of mathematics

Точно преди 300 години на бял свят се появява Ars Conjectandi на Яков Бернули (Bernoulli, 1713), една от най-значимите за развитието на науката книги. Няма съмнение, че тази годишнина е определяща за обявяването на 2013 г. за Международна година на статистиката във века на стохастиката, последвал вероятностната революция на XX век. Началото на статистиката в Третата българска държава виждаме през 1881 г. в първото преброяване на населението, което е организирано от Михаил Сарафов, завършил Математическия факултет на Мюнхенската политехника. През 1880 – 1881 г. Сарафов е министър на народното просвещение и идеята му за създаване на Висше училище в София се реализира през 1888 г. Една година по-късно е формиран вторият факултет – „Физико-математическият“. Един от „главните“ предмети на специалността „Математика“ е „Теория на вероятностите и метод на най-малките квадрати“. Скоро след това се образува „Юридически факултет“, в чийто „Стопански отдел“ се изучава „Статистика“. Какво се е изучавало, какво е било съдържанието на тези предмети, можем да гадаем по запазените учебници и друга литература в библиотеката на Математическия институт (Шоурек, 1912). Измежду тях е и първият ни известен учебник на български език по теория на


Кратка Теория
на
Вероятность та.
Отъ
Хр. Д. Бахтадженевъ
(Учитель при Първата реална гимназия „Александръ I“).
Пловдивъ
1890 г.

Фиг. 1. Корицата на учебника