Шаблони, подпомагащи облачните изчисления

Мария Армянова

Design Patterns Supporting Application Development in Cloud Computing Environment

Marya Armyanova

Abstract

Cloud computing allows use of computing power, data storage and network with access management. The cloud applications development is facing many new challenges and design patterns can facilitate their overcoming. The existing design patterns can be used directly or with transformations can be applied in the cloud computing environment. The paper presents design patterns that can be used for cloud computing establishing and methodology of design patterns usage for software development in the cloud environment.

Keywords: Design Patterns, Cloud Computing, Gang of Four, AWS, Microsoft Azure, cloudpatterns.org

Въведение

Облачните изчисления са водещо направление в компютърната индустрия. Те позволяват да се намалят разходите на бизнеса, като се премахне нуждата от закупуване на софтуерен лиценз за всеки служител; намали се необходимостта от постоянното обновяване на хардуера; премахнат се разходите за наем на физическо място на сървъра, за съхраняване на данни или за базите от данни и т.н.. Доставчиците на облачни услуги предлагат използване на изчислителна мощ, място за съхранение на данни и мрежа, позволяваща управление на достъпа.

Групи шаблони

За да се преодолеят предизвикателствата, пред които се изправя развитието на облачните приложения, различни организации и учени предлагат шаблони, подходящи за използване в облачна среда. Освен ново създадени шаблони за облачните изчисления и някои от вече разработените шаблони за проектиране могат да се прилагат директно или да бъдат прехвърлени в областта на облачните изчисления с корекции. Големите групи шаблони, които се използват в областта на облачните изчисления са разработени от PLoP, AWS, Microsoft Azure, групата, предлагаща шаблони на cloudpatterns.org и архитектурните шаблони на Уайлър. Представени са в таблица 1.

Таблица 1. Групи шаблони.

Групи шаблони	Категории шаблони в групата	Брой
PLoP	Шаблони на Гама и Бушман; шаблони на Хоф и Уолф; шаблони на Хамнер, Шумахер и Фолер; и др.	–
AWS	Основни и мрежови шаблони; шаблони за обработка на динамично и статично съдържание; шаблони за релационни БД и качване на данни; шаблони за достъпност и групова обработка; шаблони за експлоатация и поддръжка	47
Microsoft Azure	Шаблони за разпределени системи; за облачни приложения; шаблони за облачната платформа	34
cloudpatterns.org	Механизми; шаблони ориентирани към услуги (SOA); базови шаблони и смесени шаблони	88
Уайлдър	Архитектурни шаблони за облачни платформи	11

Група от учени, от PLoP[1] разработва редица успешно приложими шаблони за IT домейни. Към тях принадлежат шаблоните за обектно-ориентирано програмиране, предложени от голямата четворка [1, 2], шаблони за базирана на съобщения интеграция на приложения [3], устойчиви на сривове системи [4], или системи с разпределен контрол [5].

Така са определени само част от съществуващите шаблони, подпомагащи облачните изчисления. За да се систематизират многобройните шаблони се използват каталози от шаблони, отнасящи се за облачните изчисления. Съществуват няколко каталога от такива шаблони представени в [6]. В тях шаблоните са класифицирани според различни характеристики – например според приложението си [1, 3], реализираната функционалност, или организацията.

Шаблоните, разработени за други области, могат да се използват и в областта на облачните изчисления. Някои могат да се прилагат директно без промени и в такъв случай могат да се възприемат, като шаблони, подпомагащи облачните изчисления. Други могат да са полезни за редица проблеми, възникващи при облачните изчисления, като е възможно начинът им на приложение в облака да не е предвиден в описанието на оригиналната версия на шаблона. Счита се, че каталогът съдържа съществуващ шаблон, ако той може да се приложи без промени в облачна среда и че съдържа версия на съществуващ шаблон, ако концепцията заложена в шаблона може да се приложи за облачна среда.

Шаблоните за обектно-ориентираните приложения, описани от Гама и Бушман [1, 2] описват обекти и връзките между тях. Компонентите, съставящи облачното приложение в същността си могат да се съпоставят с обектите обхванати от тези шаблони. За да се приложат тези шаблони се търсят общи концепции, например за разделянето на модули на облачното приложение. За реализацията на облачните услуги могат да се приложат класическите шаблони Команда, Посредник, Прототип и др.

Хоф и Уолф [3] описват как приложенията могат да се интегрират посредством размяна на съобщения. Каталогът им от шаблони се позовава на абстрактна концепция за обмен на съобщения, т.е. съобщителен канал и начин, по който приложенията си взаимодействат със съобщителната система, т.е. съобщителен адаптер. Тези абстрактни концепции се откриват в различни типове адаптери и компоненти на съобщителния канал. Тези шаблони са предназначени за решаване на проблеми със загубата на съобщения, дублирането на съобщения, подреждането на съобщения и т.н.. И тъй като и приложенията, основаващи се на съобщения и облачните приложения са разпределени приложения, този тип шаблони са подходящи за интегриране с шаблоните, подпомагащи облачните изчисления.

Облачните приложения, разчитат на множество ресурси, затова много често възникват грешки по време на изпълнение на облачното приложение. Разработените от Хамнер [4] шаблони, свързани с устойчивия на сривове софтуер, са подходящи за интеграция с шаблоните, подпомагащи облачните изчисления. Шумахер [7] предлага шаблони за сигурност, които са приложими за ИТ приложенията и затова са приложими и в облачна среда.

Фолер [8] описва архитектурни шаблони за софтуер, обслужващ дейността на фирмите. Много от шаблоните описват как бизнес изискванията се моделират и усъвършенстват чрез приложната функционалност, реализираща бизнес модела. Такива шаблони са подходящи за интеграция с шаблоните, подпомагащи облачните изчисления, тъй като бизнес архитектурата трябва да се поддържа и от приложението, работещо в облачната среда.

Първата колекция от шаблони е предложената от PloP Conferences. Авторите [9] откриват неуместната употреба на шаблони за публичните облаци. Например хакер може да качи картинка на виртуален сървър, която клиентите да използват. А места, където

[1] Pattern Languages of Programs

потребителите могат да споделят изображения са много разпространени. По този начин хакерът може да добави и злонамерна функционалност към изображението. Такава функционалност са различни инструменти за следене на клиентите, които изпращат събраната информация на хакера. Авторите [10] откриват подобни шаблони за всички слоеве на облачната архитектура. Фернандес [11] описва още възможностите за неправилна употреба на шаблоните, като се забрани достъпът до услуги. В друго съавторство Фернандес [12] представя два шаблона за изграждане на защитна стена с цел контрол на мрежовия достъп до облачните ресурси. Те подобряват и разширяват съществуващите до този момент шаблони за сигурност [7]. Дара [13] описва два шаблона за симетрично и асиметрично кодиране на достъпа до доставчика на облачните услуги. Хамнер [14] подобрява шаблоните си [4] за устойчивия на сривове софтуер.

Втората голяма група от 47 шаблона са представени от Amazon Web Services. Те описват общи модели за работата на приложенията в облака на Amazon. Представеният в модела начин на работата на приложенията се изисква от доставчика на облачни услуги Amazon. Шаблоните са категоризирани според предназначението си и обединяват опита на различни разработчици на облачни приложения при решаването на даден проблем. Те предлагат решение базирано на облачните изчисления, вместо традиционите разработки, като по този начин осигуряват по-евтино и гъвкаво решение.

25 от предложените шаблони подобряват шаблоните, подпомагащи облачните изчисления или техни аспекти за AWS облак. Напрактика са насочени към конкретните детайли свързани с усъвършенстването на абстрактен шаблон за облачни изчисления и привеждането му в съответсвие с техническите особености на Amazon облак. Останалите 22 шаблона осигуряват разширение на шаблоните, подпомагащи облачните изчисления и позволяват прилагане на технически решения съобразени с предлаганите облачни услуги от Amazon. 15 от тези шаблони са разширение на съществуващи шаблони, но тези разширения не са съобразени с предлаганите услуги от другите доставчици на облачни услуги. Затова и са специфични и предназначени за потребител, който е решил да базира приложението си на Amazon платформата. Възможността тези шаблони да се използват с други доставчици на облачни услуги или в други разновидности на облачните приложения, подлежи на проучване и описание.

Друга група от 34 шаблона е предложената от Microsoft за разработка на системи, поддържани от Windows Azure или друга облачна платформа [15]. Те са предназначени за разпределените системи, облачните приложения и за облачната платформа - Microsoft Azure. Въпреки, че шаблоните се фокусират върху Azure, авторите твърдят, че представените концепции са общо приложими. От представените шаблони 11 са усъвършенстване на съществуващи шаблони, подпомагащи облачните изчисления или техни аспекти с оглед по-добрата им съвместимост с Microsoft платформата. Те показват как даден облачен шаблон, може да се приложи, в съответствие с облачните технологии на Microsoft.

Единият от шаблоните, предназначен за репликация и синхронизация на данните, стъпва върху няколко от съществуващите шаблони, в случай на необходимост от репликация и синхронизация на данните. Оставащите 22 шаблона разширяват шаблоните, подпомагащи облачните изчисления. Повечето от тези шаблони са свързани със системите за управление на данните и системи за предаване на съобщения (месинджери, чат), които не са специфични за облачните шаблони. Шаблоните съответстват на шаблоните на Хоф и Уолф [3] за системи за предаване на съобщения и на Код; [16] за базите от данни.

Също както и при шаблоните на AWS, те могат да се обобщят, за да се използват с различни доставчици на облачни услуги или в облачни приложения, работещи едновременно на няколко облачни платформи. Тези разширени шаблони, позволяват: контролирането на всички необходими ресурси за екземпляр на приложението; директен достъп до хранилището за данни, така, че да се повиши производителността на приложението;

регулацията и делегацията на права на приложенията в сложни архитектури на структури от данни.

Другата група от 88 шаблона са предложени от общността cloudpatterns.org[2] [17]. Каталогът включва 20 механизма, 55 шаблона за проектиране и 13 смесени шаблона. Механизмите не са описани като шаблони, но представят базови концепции и техники за облачни изчисления. Смесените шаблони предлагат списък от шаблони за проектиране, съответстващи на приложението им, но без допълнителни пояснения. Примерите и приложенията не са обвързани с определен доставчик на облачни услуги или технология, затова и нямат конкретно приложение за специфична технология.

Осем от шаблоните са обвързани с шаблоните за разработка на приложения в облака. 35 шаблона описват вътрешната работа на облака и затова остават извън обхвата на шаблоните за облачните приложения. Те обхващат аспекти на технологията за виртуализация, мрежовата организация на хранилищата на данни, обезпечаването на физическия хардуер за доставчиците на облачни услуги така, че да се създаде облачна среда. Шаблоните за за облачните приложения са фокусирани върху архитектурата им и начина на използване на предлаганите от облачните доставчици възможности. Те описват технологиите за създаване на облаци и приложения, хоствани на така създадените облаци. От тази гл. т. подобен каталог може да подпомогне комуникацията между създателите на облаци и на облачни приложения за тях.

Друга група шаблони са архитектурните шаблони за облачни платформи. Уайлдър предлага 11 такива шаблони, като приема за облачна платформа Windows Azure [18]. Примерно тяхно предназначение е за приложения за споделяна на снимки. Три от архитектурните шаблони са съставени от няколко шаблона, подпомагащи облачните изчисления и обединяват възможностите им. Базираните на облачна платформа приложения изискват висока степен на автоматизация на управлението на ИТ ресурсите си, например за избягване на възможните липси на ресурси, поради технически аварии [19, 20]. Автоматизацията е възможна, благодарение на осигурения от доставчиците на облачни услуги, интерфейс за управление на достъпа на приложенията до ресурсите без човешка намеса. Шаблони трябва да предвидят загубата на ресурс по всяко време и да се съсредоточат върху асинхронното взаимодействие. Затова пет от шаблоните на Уайлдър предлагат възможности за разширение чрез насоки за синхронизация на взаимодействията и отработване загубата на ресурс. Три от шаблоните са вариация на разпределени облачни шаблони.

Други особености на облаците, като заплащане единствено за използване и ниските изисквания към индивидуалните ресурси на клиентите, също изискват управление на системата в реално време. Една от тези задачи е управлението на потоците. Може да бъде реализирана с помощта на архитектурни шаблони [19, 21].

Основните архитектурни шаблони позволяват реализацията на фундаменталната структура на облачните приложения. Гъвкавите шаблони определят начините за нагаждане на размера на облачното приложение към текущия работен поток [19]. Тези концепции са фундаментални за полезността и ефективността на приложението, когато принципът на заплащане само за използвани ресурси е водещ [22]. Така се гарантира ефективност на икономията от мащаба, дори в частни облаци, тъй като ресурсите се разпределят най-добре между много приложения, работещи в облак, ако се предоставят само за период, през който са необходими. Други архитектурни шаблони подпомагат реализацията на нивата на достъп в облачните приложения [23, 24]. Те описват начинът, по който облачните приложения и компонентите им се споделят от множество потребители.

Част от шаблоните описват вътрешната работа на облака и затова остават извън

[2] http://cloudpatterns.org/ (9.10.16г.)

обхвата на шаблоните за облачните приложения. Те обхващат аспекти на технологията за виртуализация, мрежовата организация на хранилищата на данни, обезпечаването на физическия хардуер за доставчиците на облачни услуги така, че да се създаде облачна среда. Шаблоните за облачните изчисления са фокусирани върху архитектурата на облачните приложения и начините за използване на възможностите, предложени от облачните доставчици. Те описват технологиите за създаване на облаци и приложения, хоствани на така създадените облаци. От тази гл. т. подобен каталог може да подпомогне комуникацията между създателите на облаци и на облачни приложения за тях.

Класификация на облачните шаблони

Филинг систематизира различните групи облачни шаблони, като предлага класификация на категориите шаблони според предназначението им. Шаблоните са разделени в две направления – шаблони за реализация на приложенията, работещи в облака и шаблони за реализация на самата облачна среда (фиг.1).

Фигура 1. Разделение на шаблоните по области на приложение.

Реализацията на базовите функции на доставчика гарантира създаването на облачна среда. Шаблоните от тази категория описват модели на облачни услуги и типове разполагане в облак, аналогични на NIST определението за облак [25]. Тези шаблони разширяват определението чрез покриване на условия, при които определен модел на облачни услуги и тип разполагане в облак трябва да се използва за облачни приложения. Шаблоните от категория на облачните предложения на доставчика, представят допълнителната функционалност, предлагана от доставчика. Те описват как функционалността може да се използва от приложение за обработка на работните потоци, комуникацията и хранилищата на данни. Също така показват условията, при които трябва да се избере един от предлаганите шаблони и последиците за приложението, което ги използва. Следователно, тези две категории шаблони за облачни изчисления не се прилагат от приложения работещи в облака, а от самата среда на облака. Те описват как доставчикът осигурява услуги и кога те трябва да се използват.

Шаблоните от категорията на архитектурите на облачното приложение описват общата му структура и специфичните му компоненти за потребителския интерфейс, изчисленията и обработката на данни. Шаблоните от категорията на управлението на облачните приложения описват как приложенията могат да се управляват по време на изпълнение, като се използват допълнителни компоненти за управление, които разчитат на функционалността, предоставена от самото приложение; предлаганата от доставчика специфична функционалност на облака; и средата на облака. Шаблоните от категорията на съставните облачни приложения покриват честите комбинации от всички останали категории шаблони в различните случаи на употреба.

Подход за прилагане на шаблоните при разработка на приложение за облак

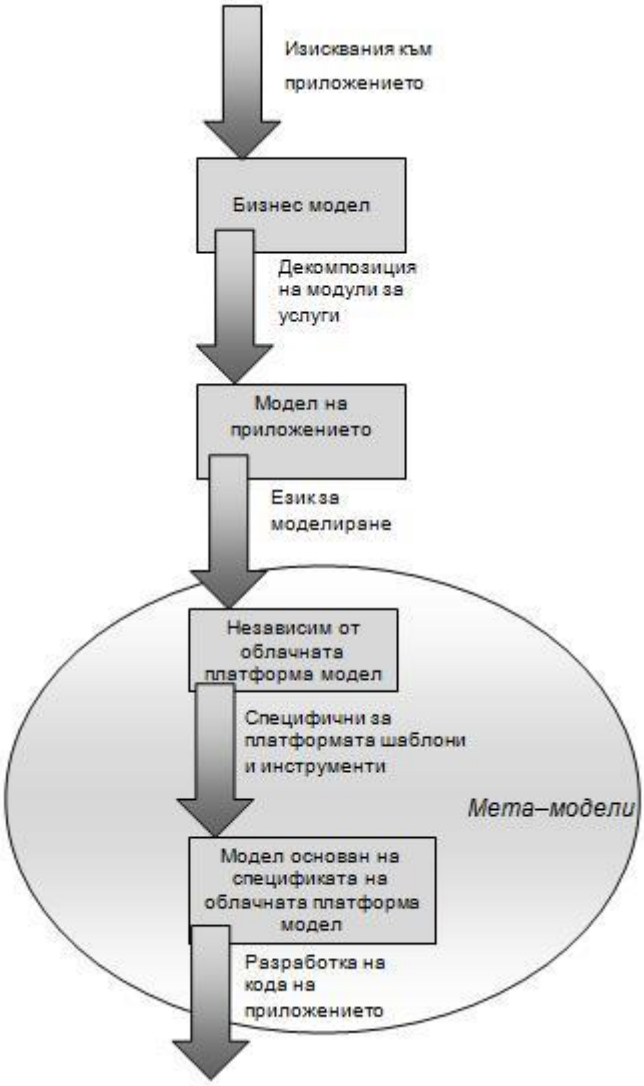
Разработката на облачно приложение се реализира посредством подходи базирани на модели. Правилното проектиране на приложения за платформата SaaS основано на MDA (Model Driven Architecture) [26], разделя архитектурата на приложението на два слоя. Първият слой описва данните, бизнес процесите и потребителския интерфейс. Във втория слой се описва модела на инструментите, отговорни за реализацията на персонализацията на приложението. Чрез персонализацията всеки потребител може да получи услугата в уникален, подходящ за него вид. Така услугата може да се промени според изискванията на клиента. Облачната платформата SaaS и MDA представят процеса на разработка на софтуер, като акцентират върху връзката между проблемна област и област на решение и по този начин се увеличава управляемост, производителността и скоростта на приложението. SOA декомпозира приложението на услуги. Това позволява повторната употреба на софтуера за различните услуги, автономността на развитието и използването им, абстракция им, балансирането им, преносимостта им, откриваемостта и свободно свързване на услуги. Авторите Шарма и Суд [27], предлагат моделиране на системата на по-високо ниво на абстракция за предотвратяване на зависимостта на модела на приложението от използваната технология. Техният подход пренебрегва проблемите на персонализацията на приложенията. Персонализацията се описва от Янг, Джан и Лю [28]. Моделът базиран на адаптирането е предложен от Лайтнер и се фокусира върху поддръжката на приложението от страна на доставчика на облачни услуги [29]. При този подход се използва информацията от доставчика на услуги, за да се адаптира приложението, съгласно изискванията му. Освен това, подходът предполага използване на информацията, генерирана по време на изпълнение, за да се определи поведението на приложението и да се осигури ефективно управление на софтуера. Управлението на приложението включва подсигуряване на нужната инфраструктура (ресурси), поддръжката на приложението и следователно адаптацията му по време на изпълнение. Таблица 2 показва сравнението между различните подходи за разработка на облачни приложения.

Таблица 2. Подходи за разработка.

Архитектури	Недостатъци
DSL (Domain Specific Language) подход [29]	Подходът е зависим от облачната платформа и изисква използването на специфичен език за всяка облачна платформа, където ще се разполага приложението.
Подход за мнократно използване на софтуера [27]	Прави се компромис с качеството на софтуера, тъй като някои от необходимите на приложението компоненти може да не са налични в хранилището на облака.
MDA подход за SaaS и EMAD (Enterprise Mashup Application Development) [31]	Този подход не представя персонализацията на услугите, затова не показва как се използва от множество потребители едновременно.
Обектно-ориентиран подход [32]	Платформо зависим подход, съобразен с определена среда за разработка.
MDA подход за SaaS [28]	Пак се игнорира персонализацията, съобразно изискванията на потребителите.

Предложените подходи имат някои недостатъци – процесът на разработка е съобразен с облачната платформа; персонализацията на услугите не е предвидена; не е явно как се осигуряват услугите за множество потребители; не е представена последователността на работа, при липсата на някой готов компонент за приложението.

Подходът може да комбинира горните подходи, за да се преодолеят недостатъците на всеки от тях. Подходът за разработка на облачно приложение се основава на следната последователност показана на фиг. 2:



Фигура 2. Последователност на моделите при разработка на приложение с облачни шаблони.

Разработката на приложението започва със създаването на бизнес модел, който е независим от програмната реализация. Той описва единствено бизнес логиката на приложението. Разработеният бизнес модел съответства на BpaaS слоя на архитектурата на облака. Специалистите на IBM въвеждат модел BpaaS „Бизнес процес като услуга“ и го обособяват, като четвърти слой в класификацията на облачните услуги и базовата архитектура на NIST[3]. От технологична гледна точка дейностите на бизнес процеса като услуга се реализират чрез облачни услуги [30]. Важно е да се уточни, че за целта се ползват услуги от трите по-долни слоя на архитектурата на облачните технологии – софтуера като

[3] National Institute of Standards and Technology

услуга (SaaS), инфрастуктурата като услуга (IaaS) и платформата като услуга (PaaS). По-конкретно може да се ползват SaaS приложения за реализиране на целите на потребителите, средства за изпълнение и управление на приложения от PaaS, както и инфраструктурни услуги за изпълнение на приложенията и съхраняване на данните. Затова и някои автори определят „Бизнес процес като услуга“ не като самостоятелен слой в архитектурата на облачните услуги, а като „обвивка“, която интегрира трите основни слоя – IaaS, PaaS, SaaS.

За да се детайлизира разработеният бизнес модел, първо се определят основните услуги, които трябва да реализира разработваното приложение. Те съответстват на SaaS слоя на облачната архитектура. На тази база приложението може да декомпозира на отделни модули за услуги и да се създаде функционално пълен модел на приложението. След това за всеки модул, реализиращ дадена услуга, се разработва модел, базиран на определен език за моделиране. По принцип е препоръчително да е UML, тъй като документацията на повечето шаблони е разработена чрез него. Ако моделите са разработят чрез UML е много по-лесно да се направи съпоставка между наличните шаблони и нуждите на приложението, за да се избере най-подходящият шаблон за дадена разработка. Ако такъв не е наличен, то съответната функционалност се проектира чрез UML модели. По-този начин се създава модел на приложението, който е независим от облачната платформа. Това е метамодел, който включва множество модели за отделните услуги.

Нужно е моделът да се съобрази със спецификите на облачната среда, в която ще работи приложението. Така чрез избор на подходящи инструменти и конкретни шаблони, които ще се използват в бъдещата разработка, могат пълно да се използват функционалните възможности, предлагани от облачния доставчик. Този последен модел е основан на спецификата на определена облачна платформа. След като се разработи модел независим от облачната среда може да се използва и автоматични средства за трансформация и да се генерира модел, основан на спецификата на облачната платформа. Такива инструменти има разработени за трасформацията на един модел в друг или пък за генериране на програмен код от модел. Подобни инструменти се основават на дефиниции за трансформация, определени от правилата за трансформации. По време на прехода от независимия към определния от средата модел, всеки отделен модел на модул за услуга се преобразува самостоятелно. Така отделната услуга е насочена към облачната среда, в която ще работи и има възможност отделните услуги да се изпълняват в различна среда и накрая да се разработи приложение, работещо едновременно с няколко доставчика на облачни услуги. Накрая отделните модели се интегрират в пълен модел на приложението, основан на облачната среда. След всяка трансформация на модел на услуга, трябва да се следи за качеството ѝ. За целта трябва да се измери качеството на услугата и да се провери дали отговаря на първоначалните изисквания. Могат да се разработят различен брой модели, ако се използват различни инструменти. И накрая трябва да се генерира кодът на приложението съобразно разработения пълен модел на приложението, основан на облачната среда. Например, при преобразуване на модела в платформа на java, всеки клас и атрибут от модела съответства на клас и атрибут в java платформата. Типът на атрибута е специфициран на java и е private. Всяка операция на класа се реализира от private член-функция на класа, която връща резултат съобразен с типовете в java.

Генерирането на кода става полуавтоматично, като се използва кодът на подбраните подходящи шаблони, които най-пълно съответстват на отделните модели на услуги. Макар за част от дейностите да има средства за разработка, е необходимо дейно участие на разработчика, за да реализира специфичните изискавания към кода на шаблона и да интегрира отделните услуги, реализирани от различни шаблони в едно цялостното приложение. Може да е необходимо да се разработи кодът на част от услугите, ако няма подходящ шаблон за тях.

След като приложението е разработено следващата фаза е разполагането му в облака.

Доставчикът на облачните услуги изисква необходимите за разполагането на приложението данни. Те включват модели и т.нар. битове на приложението. Необходимите за доставчика модели са комбинация от модели на приложението, а битовете на приложението представят поведението и реалната функционалност на различните компоненти. За да са достъпни за доставчика тези битове се представят като пакет или URL, от където могат да се изтеглят. Моделлите и битовете се комбинират в съответствие с изискванията в пакет и се въвеждат в облака. Облачната платформа анализира пакета и разработва план за разпределение на компонентите на приложението между различните сървъри. Целта е да се постигне необходимото качество на облачната услуга. При това разработеният план трябва да гарантира динамика на осигуряваната услуга, така, че когато нараства търсенето на дадена услуга, осигурявана от приложението, да могат да се създадат нови екземпляри на приложението. Аналогично при намаляване на търсенето на услугата, част от тези екземпляри трябва да могат да се изтрият. Когато приложението се качи в облака и работи, е необходимо да се предостави интерфейс, така че и отдалечените клиенти да получат достъп до тези услуги. Този интерфейс се предоставя от уеб услугите [27]. Уеб услугата има няколко стандарта WSDL, UDDI, SOAP. WSDL[4] описва услугите, които са достъпни в рамките на приложението и пояснява начина на достъп до тези услуги. SOAP[5] уточнява чрез комуникационен протокол, формата на съобщенията за обмен на информация между услугите. UDDI[6] поддържа регистъра на различните типове услуги, чрез които ги локализира. Всеки потребител има уникален идентификатор, чрез който може да достъпва услугата. Потребителите използват различно множество от опции, според работната си среда. Това произтича от необходимостта за персонализация на услугата [26] и се постига чрез изменение на моделите. Пакетът на приложението се състои от шаблони на модела, файлове с описанието на модела и инструментите на модела. Шаблоните на модела са реалните модели, но представени чрез определен стандарт. Файловете с описанието на модела са файловете, получени след персонализирането или модифицирането на шаблоните на модела. Те са специфични за всеки потребител и се съхраняват в отделна директория, предназначена за този потребител. Така всеки потребител може да използва услугите независимо от останалите. Инструментите на модела помагат при персонализацията на файловете с шаблоните и създават файловете с описанието на модела.

Основна характеристика на облачните приложения е мащабираемостта. Когато търсенето на услугите надвишава прага на обслужване, се създават нови екземпляри на услугата. Тези екземпляри трябва да се изтрият, когато търсенето спадне под определено ниво. За да се автоматизира създаването и изтриването на екземпляри, е необходимо да се включат подходящи показатели в модела, уточняващи кога да се създаде и унищожи екземпляр. Освен това към модела може да се добави информация, показваща начините за адаптация на приложението в облачната среда [27]. Различните доставчици на услуги изискват приложенията им да се държат по различен начин. Някои изискват приложението им винаги да е достъпно, други – приложението им да има малко време за реакция дори в пиковете часове, а някои предпочитат по-ниските разходи за използването на облачната среда. Тази информация помага на доставчика на облачните услуги да осигури управление, съобразено с изискванията на приложението. За да управляват работата на различни приложения, облачните доставчиви предсказват работа и диманиката на разпределение на ресурсите на приложенията. За целта използват данните за работата на подобни приложения.

Разработването на модел, който е независим от облачната платформа, позволява приложението да се пренася на различни облачни платформи. Освен това трябва да се опишат начините за персонализация на услугите, за да може приложението да обслужва

[4] Web Service Definition Language – език за дефиниране на уеб услуги
5 Simple Object Access Protocol – протокол за достъп
[6] Universal Description, Discovery Integration – универсален регистър

едновременно голямо разнообразие от клиенти. Клиентите на услугата могат да я персонализират според изискванията си, като данните за всеки клиент се пазят в отделна директория, свързана с идентификаторите им. Така те получават достъп до независима от останалите услуга и един екземпляр на софтуера може да обслужи множество клиенти. Подходът за разработка се съобразява с различни въпроси на облачната архитектурата и осигурява рамка за по-добро развитие на облачния софтуер.

Създаването на модели, независими от облачната платформа, осигурява на софтуера преносимост и приспособимост към промените в средата. Един такъв модел може да се трансформира в няколко различни модели, съобразени с облачните платформи. Така се избягва нуждата от препроектиране на приложението за различните платформи и се удължава жизненият му цикъл. Приложението е описано чрез модели и трансформациите между тях, което осигурява по-доброто им възприемане от заинтересованите страни. Декомпозицията на услуги за SOA, позволява услугите да се разработват относително самостоятелно. Така по-лесно се открива подходящ шаблон, реализиращ услугата. Разработените модули за услуги могат да се ползват повторно, като шаблони при усъвършенстване на приложението или в други приложения. Съхраняването на данните на клиента в отделна директория, освен че позволява персонализация на услугите, гарантира и някакво ниво на сигурност и поверителност на клиентските данни. Използването на шаблоните намалява нуждите от тестване и документиране на част от софтуера, което намалява разходите за разработка и гарантира определено ниво на качество на приложението. Подходът обаче има и недостатъци, като например големият брой модели, които се създават, затруднява управлението им. Моделите не са готово приложение, затова трябва да се преобразуват в програмен код, преди да се изпълнят. Това води до допълнителни разходи.

Заключение

Докладът представя шаблоните, които могат да се ползват за реализиране на облачните изчисления и гарантиране на сигурността на облачните услуги. Те могат да се ползват за самостоятелни приложения, мрежови приложения, както и за базирани на съобщения приложения, като при това се набляга конкретно на облачната среда.

Шаблоните за облачни изчисления е рационално да се класифицират според облачните приложения, за които се използват. Следователно всеки шаблон описва някои аспекти на потребителската група, на облачната среда по време на работа или на облачното приложение. Това помага на потребителите да открият подходящите шаблони за нуждите на облачното приложение, което ще разработват или усъвършенстват.

Предложеният подход показва последователността на разработване на различни модели на облачното приложение, като осигурява създаване на пълна документация на приложението и реализация на изискванията към приложението. Използването на шаблоните намалява разходите, времето и трудността при разработката на приложението. Така разработените приложения са с гарантирано ниво на качество, по-дълъг жизнен цикъл, гъвкави, преносими и могат динамично да се мащабират и персонализират според нуждите на потребителите.

Използвана литература

1. Gamma, E. Helm, R. and Johnson, R. E. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

2. Buschmann, F. Meunier, R. Rohnert, H. Sommerlad, P. and Stal, M. Pattern-Oriented Software Architecture. Wiley, 1996.

3. Hohpe, G.and Woolf, B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley: Reading, MA, USA, 2004.

4. Hanmer, R. Patterns for Fault Tolerant Software. Wiley, 2007.

5. Eloranta, V.-P. Koskinen, J. Leppnen, M. and Reijonen, V. Designing Distributed Control Systems: A Pattern Language Approach. Wiley Publishing, 2014.

6. Fehling, C. Leymann, F. Retter, R. Schupeck, W. and Arbitter, P. Cloud Computing Patterns. Springer, 2014.

7. Schumacher, M. Fernandez, E. B. Hybertson, D. Buschmann, F. and Sommerlad, P. Security Patterns: Integrating Security and Systems Engineering. Wiley, 2006.

8. Fowler, M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.

9. Hashizume, K. Yoshioka, N. and Fernandez. E. B. "Misuse Patterns for Cloud Computing." In: Proceedings of the Asian Conference on Pattern Languages of Programs (AsianPLoP). ACM. 2011.

10. Hashizume, K. Fernandez, E. B. and Larrondo-Petrie, M. M. "Cloud Service Model Patterns." In: Proceedings of the Conference on Pattern Languages of Programs (PLoP), 2012.

11. Encina, O. Fernandez, E. B. and Monge, R. "A Misuse Pattern for Denial-of-Service in Federated Inter-Clouds." In: Proceedings of the Asian Conference on Pattern Languages of Programs (AsianPLoP). 2014.

12. Fernandez, E. B. Yoshioka, N. and Washizaki, H. "Patterns for Cloud Firewalls." In: Proceedings of the Asian Conference on Pattern Languages of Programs (AsianPLoP). 2014.

13. Dara, S. "Privacy Patterns in Public Clouds." In: Proceedings of the Indian Conference on Pattern Languages of Programs (GuruPLoP). 2014.

14. Hanmer, R. "Patterns for Fault Tolerant Cloud Software."In: Proceedings of the Conference on Pattern Languages of Programs (PLoP). 2014.

15. Homer, A. Sharp, J. Brader, L. Narumoto, M. Swanson, Tr. Cloud Design Patterns, Microsoft Press, 2014.

16. Silberschatz, A. Korth, H. F. and Sudarshan, S. Database System Concepts. Mcgraw-Hill Professional, 2010.

17. Erl, T. Cope, R. Naserpour, A.Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, 2015.

18. Wilder, B. Cloud Architecture Patterns Develop cloud-native applications, O'Reilly Media, 2012.

19. Fehling, C.; Leymann, F.; Retter, R.; Schumm, D.; Schupeck, W. An Architectural Pattern Language of Cloud-based Applications. In Proceedings of the 18th Conference on Pattern Languages of Programs (PLoP 2011), 21–23 October 2011.

20. Malone, T.; Blokdijk, G.; Wedemeyer, M. ITIL V3 Foundation Complete Certification Kit; Emereo Pty Ltd.: Brisbane, Australia, 2008.

21. Varia, J. Architecting for the Cloud: Best Practices. Technical Report, Amazon, 2010.

22. Lagar-Cavilla, H.A.; Whitney, J.A.; Scannell, A.M.; Patchin, P.; Rumble, S.M.; De Lara, E.; Brudno, M.; Satyanarayanan, M. SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing. In Proceedings of the 4th ACM European Conference on Computer Systems, Nuremberg, Germany, April 2009.

23. Somorovsky, J.; Heiderich, M.; Jensen, M.; Schwenk, J.; Gruschka, N.; Lo Iacono, L. All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces. In Proceedings of the 3rd ACM workshop on Cloud computing security workshop (CCSW), Chicago, IL, USA, 17–21 October 2011.

24. Storage Networking Industry Association (SNIA): Cloud Data Management Interface (CDMI) Whitepaper, 2010.

25. Mell, P. and Grance, T. The NIST Definition of Cloud Computing. National Institute of Standards and Technology (NIST), Sept. 2011.

26. Esparza-Peidro. J., Mu˜noz-Escoi., F., "Towards the Next Generation of Model-Driven Cloud Platforms", Institut Universitari Mixt Tecnologic d'Informatica`Universitat Politecnica`

(SPAIN), Technical Report TR-ITI-SIDI-2011/001.

27. Sharma, R., Sood, M., "Enhancing Cloud Saas Development with Model Driven Architecture", International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.1, No.3, November 2011, DOI: 10.5121/ijccsa.2011.1307.

28. Jiang, X., Zhang, Y., Liu, S., "A Well-designed SaaS Application Platform Based on Model-driven Approach",.Ninth International Conference on Grid and Cloud Computing .978-0-7695-4313-0/10 © 2010 IEEE DOI 10.1109/GCC.2010.62.

29. Inzinger, C., Satzger, B., Leitner, F., Hummer, W., Dustdar, S., "Model-based Adaption of Cloud Computing Applications.", Modelsward, 2013,- Internatioal Conference on Model-Driven Engineering and Software Development.

30. Филипова, Н., „Фактори за успеха на модела „Бизнес процес като услуга"", списание „Бизнес управление", Академично издателство „Ценов", Свищов, Година XXV, кн. 4, 2015.

31. Sledziewski, K., Bordbar, B., Anane, R., "A DSL-based Approach to Software Development and Deployment on Cloud.", 24th IEEE International Conference on Advanced Information Networking and Applications 2010.

32. Singh, S., Singh, R., "Reusability Framework for Cloud Computing.", International Journal Of Computational Engineering Research Vol. 2, 2012.

За контакти
ас. Мария Армянова
ИУ-Варна
armianova@ue-varna.bg