Eastern Academic Journal *ISSN: 2367–738*
Issue 3, pp. 70-89, September, 2018

КРИТЕРИИ И СХЕМА ЗА КЛАСИФИКАЦИЯ НА ШАБЛОНИТЕ

Мария Р. Армянова

Мария Р. Армянова – ИУ-Варна
гр. Варна, България
armianova@ue-varna.bg

РЕЗЮМЕ *— Шаблоните подпомагат разработчиците на софтуер, като позволяват многократната употреба на полезни софтуерни решения. За да се използват успешно, е необходимо да се откриват бързо и лесно подходящите шаблони за различните задачи при създаването на проекто-програмното решение. Голямото разнообразие на съвременните шаблони затруднява откриването и прилагането им. Възможно решение на проблема е дефиниране на класификационна схема, която да е приложима за различните видове шаблони. В статията са дефинирани критерии и обобщена класификационна схема. Целта ѝ е да класифицира, известните на автора шаблони, да позволи бъдещо развитие и лесно добавяна на ново дефинираните шаблони.*

Ключови думи: шаблони за проектиране, класификация на шаблоните, критерии, схема за класификация

PATTERNS CLASSIFICATION CRITERIA AND SCHEME

Mariya R. Armyanova

Mariya R. Armyanova
Varna, Bulgaria
armianova@ue-varna.bg

ABSTRACT— *The patterns support software developers by allowing repeated reuse of proven software solutions. It is necessary for successful creating design and program solutions to find the appropriate pattern for the different problems fast and simply. The patterns wide variety makes difficult their discovering and appling. The possible solution of the problem is definition of a classification scheme which is applicable to different patterns types. The article defines criteria and a summarized classification scheme. Its purpose is to classify the known author patterns and to allow future development and its easily addition with the newly patterns.*

Keywords: design patterns, patterns classification, classification criteria, classification scheme

1. ВЪВЕДЕНИЕ

Литературата непрекъснато описва нови шаблони, които решават възникналите проблеми свързани с техническото развитие. Всеки автор на шаблони за определена

предметна област предлага собствен каталог. Например шаблоните на GoF (Гама, 2004) подпомагат прилагането на обектноориентираните технологии, на Филинг (Fehling, 2014) – облачните технологии, а на Рейнфърт (Reinfurt, 2017, с. 117-126) – IoT. Всеки автор на шаблони за определена технология предлага собствен каталог. Критериите за класификация в него се различават според спецификата им. Наличието на много шаблони обърква разработчиците при избора на подходящия за даден проект. Затова цел на изследването е да се открие обща класификационна схема, която да ги подпомогне в търсенето, да е отворена за разрастване и да се специфицира според особеностите на конкретния вид.

За да идентифицираме някои общи критерии за класификация на шаблоните, които са полезни за разработчиците при откриването, прилагането и интегрирането на шаблоните, е необходимо да се анализират различни класификации. В тях могат да се открият различни критерии, които да се включат към общата схема. Нужно е да се оценят така определените характеристики според полезността и лекотата на разбиране и прилагане от разработчиците. Трябва да се анализират и като признак за класификация – универсалност, изброимост и др.

2. МАТЕРИАЛИ И МЕТОДИ

За да се дефинира схема за класификация на шаблоните са направени проучвания на съществуващите класификации на различните автори. Използвани са методи на системен, сравнителен и исторически анализ. Използвани са мисловни карти за представяне на независимите критерии за класификация.

3. РЕЗУЛТАТИ

Преглед на някои популярни класификации

При проучването е установено, че са създадени множество класификации на шаблоните, обаче в тях обикновена са класифицирани различни видове шаблони. Например Шумахер (Schumacher, 2006) предлага шаблони за сигурност, Филинг (Fehling, 2014) за облачните системи, Рейнфърт за IoT (Reinfurt, 2017, с. 117-126), а Хамнер (Hanmer, 2014) за устойчивия на сривове софтуер. След анализ на избраните от авторите критерии стигаме до извода, че те са специфични, тъй като са ориентирани към конкретен вид шаблони. Целта е откриване на обща схема за класификация, за която са ни необходими критерии, които не отразяват специфичните особености, а са по-обхватни. Подходящо е да се анализират няколко класификации за една и съща група от шаблони.

На база на изследване на различни класификации на шаблоните, обособяваме по-обхватни критерии, които не са специализирани за определен вид, а са приложими за цялото множество от шаблони. Нужно е да ги оценим според полезността и лекотата на разбиране и прилагане от разработчиците, както и като признак за класификация – приложимост към цялото множество, изброимост, разширяемост. Целесъобразно е при анализа да се използват няколко класификации за една и съща група от шаблони – например шаблоните на GoF (Гама, 2004). Те са класифицирани от Гама, Бушман (Buschmann, 1995, с. 325-343; Buschmann, 2001, с 362-368), Зимър (Zimmer, 1995), Код (Coad, 1995), При (Pree, 1995) и други от основоположниците на шаблоните. Те са малко на брой са и добре известни са, което ни позволява да се съсредоточим върху критериите за класификацията им.

Подходът на Гама за класификация се основава на два критерия – цел и обхват. Класификацията му е представена в таблица 1.

Таблица 1. *Класификация на шаблоните според Гама*

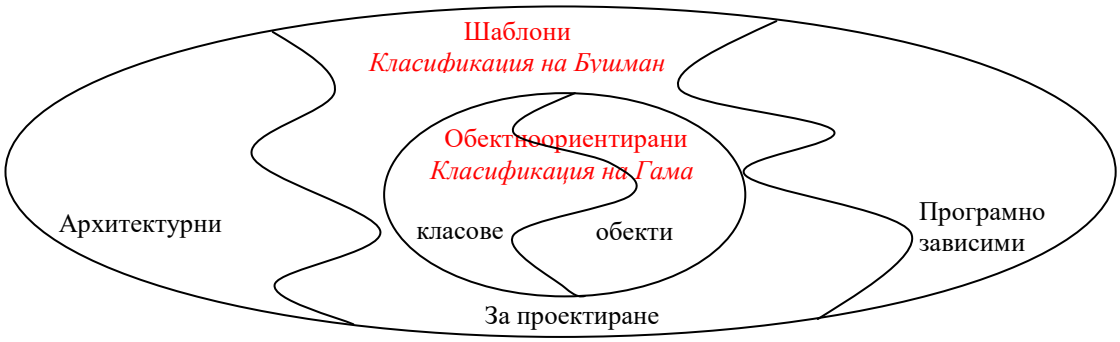
Обхват	**Цел**		
	За създаване	Структурни	Поведенчески
Класове	Фабрика (Factory Method)	Адаптер (Adapter)	Шаблонният метод (Template Method)
			Интерпретатор (Interpreter)
Обекти	Абстрактна фабрика (Abstract Factory)	Адаптер (Adapter)	Верига от отговорности (Chain of responsibility)
	Прототип (Prototype)	Декоратор (Decorator)	Итератор (Iterator)
	Единствен обект (Singleton)	Конфигурация (Composite)	Команда (Command)
	Строител (Builder)	Малък обект (Flyweight)	Наблюдател (Observer)
		Мост (Bridge)	Посетител (Visitor)
		Заместник (Proxy)	Посредник (Mediator)
		Фасада (Facade)	Възстановяване (Memento)
			Стратегия (Strategy)
			Състояние (State)

Целта позволява на разработчика да определи за какво служи шаблона. Съгласно този критерий Гама (Гама, 2004) определя три категории – шаблони за създаване, структурни и поведенчески. Шаблоните за създаване се концентрират върху процеса на създаване на обекти. Структурните шаблони са насочени към композицията на класовете и обектите. А Поведенческите шаблони характеризират начина на взаимодействие между класовете или обектите и разпределянето на отговорностите им.

Шаблоните на GoF се прилагат за класовете или за обектите. На тази база според критерия обхват има две категории шаблони – за класове и за обекти. Шаблоните за обектите реализират връзките между тях, които се променят по време на изпълнение и затова са по-динамични. Шаблоните за класовете третират връзките между класове и подкласове. Те се реализират чрез наследяване и затова са статични, т.е. създават се по време на компилация. Всеки шаблон може да принадлежи само към една от категориите за обхват, въпреки, че шаблонът Адаптер може да се използва и за класове и за обекти. Комбинирането на двата критерия създава класификацията на шаблоните на голямата четворка според Гама.

Критериите, предложени от Гама, са подходящи като основа за общата схема за класификация, тъй като са лесни за разбиране и много полезни, когато разработчикът се опитва да открие шаблон за даден проблем. Шаблоните на GoF са обектноориентирани, затова възможните категории според определените от него признаци са ограничени. Когато критериите цел и обхват се отнесат към цялото множество от съществуващи шаблони, те го разделят на други групи.

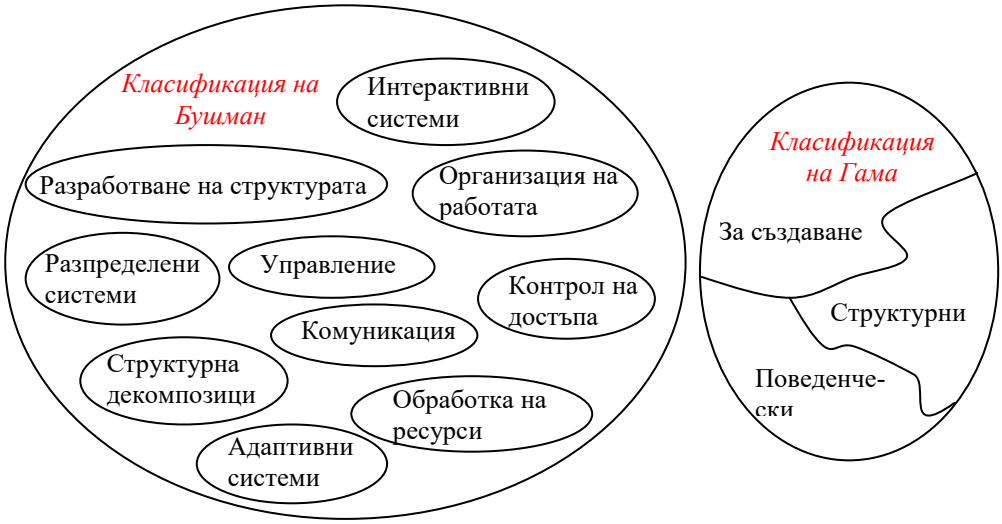
Друга класификация е предложена от Бушман (Buschmann, 1995, с. 325-343), (Buschmann, 2001, с 362-368), (Buschmann, 2005). Той класифицира собствените си шаблони според мащаба им и проблема, който решават. Класификацията му е много близка до тази на Гама. На фиг. 1 са показани множествата, на които се разделят шаблоните, според двете класификации.



Фигура 1. Множества на шаблоните според мащаба на Бушман и обхвата на Гама

Критериите мащаб на Бушман и обхват на Гама се отнасят до един и същ признак – архитектурни особености, но шаблоните, които класифицират са различни множества. Бушман се опитва да класифицира цялото множество (което му е познато по това време), докато Гама се съсредоточа само върху GoF шаблоните, а те са подмножество на шаблоните за проектиране. Считаме, че критерият мащаб е изключително подходящ за общата схема на класификация, тъй като е полезен, лесно приложим и универсален.

И в двете класификации шаблоните са разпределени по два независими критерия, което позволява класификациите да се представят в таблична форма. Вторите критерии – проблем и цел, също са близки. Съпоставката им е представена на фиг. 2.



Фигура 2. Сравнение на класификациите на Бушман и Гама по критерия проблем и цел

Критерият проблем съответства директно на решаваната от шаблона ситуация. Считаме, че той най-точно характеризира проблемната ситуация, за която се търси решение и е важен за търсенето на шаблон. Обаче за критерий в общата схема на класификация, е неподходящ, защото проблемите, които се решават са прекалено много и броят на категориите е неясен. Бушман добавя нови категории, когато класифицира и другите шаблони. Докато критерият на Гама се отнася до цялото множество, само, че множеството е ограничено до обектноориентираните шаблони за проектиране.

Ето защо е подходящо да бъде добавен като критерий за ограничено множество, т.е. като съподчинен критерий в схемата за класификация на група шаблони, например интерфейсни, за сигурност, облачни и т.н.

Таблица 2. *Класификация на шаблоните на Бушман* (Buschmann, 2001, с.366)

Проблем	**Архитектурни**	**За проектиране**	**Програмни**
Разработване на структурата	Layers Pipes and Filters Blackboard		
Разпределени системи	Broker Pipes and Filters Microkernel		
Интерактивни системи	MVC PAC		
Адаптивни системи	Microkernel Reflection		
Структурна декомпозиция		Whole-Part	
Организация на работата		Master –Slave	
Контрол на достъпа		Proxy	
Управление		Command Processor View Handler	
Комуникация		Publisher -Subscriber Forwarder -Receiver Client-Dispatcher-Sewer	
Обработка на ресурси			Counted Pointer

Бушман предлага и обединена класификация с шаблоните на Гама, като увеличава броя на категориите за проблема. Към критерия проблем се добавят категории съгласно два критерия: функционалност и структурни принципи.

Функционалността съответства на целта на шаблона, т.е. каква функционалност обслужва. Бушман определя четири категории: за създаване, за комуникация, за достъп и организиране на работата. Шаблоните за създаване дефинират как се създава конкретен екземпляр на сложна рекурсивна[1] или агрегираща[2] структура от обекти. Шаблоните за комуникация определят начина за организация на комуникацията между множество сътрудничещи си обекти, които могат да се отдалечени или да се създават и разрушават независимо един от друг. Шаблоните за достъп описват как се достъпват услуги и стойности/състояния на споделени или отдалечени обекти по безопасен начин, без да се нарушава капсулирането на състоянието и поведението им. Шаблоните за организиране на работата, организират поделянето на отговорностите между взаимодействащите си обекти, за да извършат по сложна функция или задача.

Другият критерий – структурни принципи, отразява базовите архитектурни принципи на шаблоните. Те са важни при създаването на архитектурата на системата, но не и при откриването на шаблон за нуждите на конкретен системен компонент. Според Бушман са четири категории: абстракция, капсулиране, разделяна на работата, свързване и съгласуване. Шаблоните принадлежащи към категорията абстракция осигуряват обобщение или абстракция на изгледа към определена същност или задача на софтуерната система. Шаблоните за капсулиране капсулират елементи за отделни обекти, компоненти или услуги, за да премахнат зависимостта на клиентите от тях или за да защитят достъпа до

[1] всеки обект се обръща към обект от същия тип – struct bintree {
struct bintree *left, *right;
....;};

[2] Агрегацията се различава от композицията, защото не включва собственост. При композицията при разрушаване на собственика се разрушават и принадлежащите му класове, което не е така при агрегация.

ISSN: 2367–738
Issue 3, pp. 70-89, September, 2018

тези елементи. Шаблоните за разделяна на работата определят специфични отговорности за отделни обекти или компоненти, с цел да се реши определена задача или да се предостави определена услуга. Шаблоните за свързване и съгласуване премахват или отслабват структурните и комуникационните връзки и зависимости между силно свързани обекти.

За разлика от класификацията на Гама, шаблоните могат да принадлежат на повече от една категория по съответния критерий. Схемата е представена в (Buschmann, 1995, 325-343) труда на Бушман, но не всички шаблони са описани подробно.

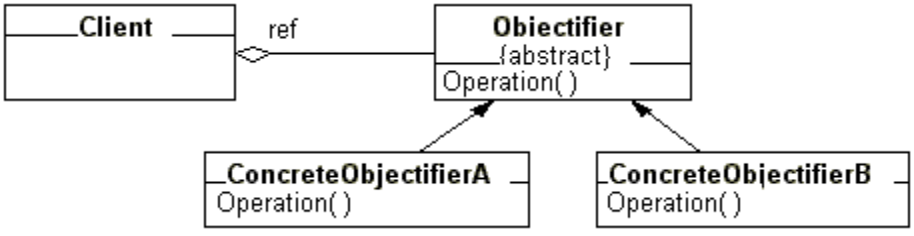
Друга класификация е тази на Зимер (Zimmer, 1995). Той класифицира връзките между шаблоните на Голямата четворка. Като база за разработката си Зимер ползва описаните от Гама връзки между шаблоните в секция Свързани шаблони. Взаимоотношенията засягат много типове връзки, например „шаблонът използва друг шаблон за реализацията си" или „шаблонът конструира обектите си по подобие на друг шаблон".

Зимер разглежда връзките от гл.т. на проблема и решението на шаблона. Разделя връзките в три категории:

- Шаблонът използва друг шаблон в решението си – такава ситуация възниква, ако решението на първия шаблон, води до друг проблем, решаван от втория шаблон. В този случай решението на втория шаблон е част от решението на първия;
- Вариант на шаблона използва друг шаблон в решението си – разликата с предната категория е в разделението, т.е. някои варианти на шаблона се нуждаят от другия шаблон, а някои не;
- Шаблонът е подобен на друг – двата шаблона се отнасят за един тип проблеми.

Подреждайки шаблоните според тези категории взаимоотношения Зимер открива три различни нива на шаблоните: базови шаблони за проектиране на техники; шаблони за проектиране на типични софтуерни проблеми; шаблони за проектиране, специфични за определена приложна област.

Класификацията на Зимер (Zimmer, 1995) е за шаблоните на Гама и дефинираните от него шаблони Конкретизатор (Objectifier), Лепило (Glue), Пасианс (Solitaire). Чрез абстрактен клас шаблонът Конкретизатор (фиг. 3) обобщава класовете, които конкретизират поведението. Лепило капсулира подсистеми, а Пасианс осигурява уникален достъп до услуги или променливи. Таблица 3 показва класификацията на шаблоните според Зимър.



Фигура 3. Диаграма на класовете за шаблона Конкретизатор на Зимер.

Таблица 3. *Класификация на шаблоните според Зимер*

Базови шаблони и техники	**Шаблони за типични софтуерни проблеми**	**Шаблони, специфични за определена приложна област**
Единствен обект (Singleton) Малък обект (Flyweight) Фасада (Facade) Шаблонният метод (Template Method) Посредник (Mediator) Итератор (Iterator) Възстановяване (Memento) Конкретизатор (Objectifier) Лепило (Glue) Пасианс (Solitaire)	Адаптер (Adapter) Строител (Builder) Прототип (Prototype) Декоратор (Decorator) Конфигурация (Composite) Мост (Bridge) Заместник (Proxy) Фабрика (Factory Method) Абстрактна фабрика (Abstract Factory) Верига от отговорности (Chain of responsibility) Команда (Command) Наблюдател (Observer) Посетител (Visitor) Стратегия (Strategy) Състояние (State)	Интерпретатор (Interpreter)

Определеният от Зимър критерий е изключително полезен за интегрирането на шаблоните, но е труден за определяне, тъй като всеки автор дефинира връзките на шаблона си с малко подмножество от шаблони, често ограничено само до шаблоните от авторовия каталог.

Код предлага подход за класификация, която също има само един критерий. Той (Coad, 1995) не придава голямо значение на класификацията си и не я описва подробно, затова и я разглеждаме само за пълнота. Код определя критерий според функционалността решавана от шаблоните в каталога. Организира шаблоните си около един шаблон, които е модел за останалите. Освен шаблони описва и стратегии (strategies), както ги нарича. Те са планове за действия при изграждане на обектноориентирани модели. Те се отнасят до етапа на анализ на обектноориентирана система и са тясно обвързани с прилагането на GoF шаблоните. Категориите на шаблоните му не са типични за цялото множество от шаблони, а са строго специфични.

Шаблоните са разделени на фамилии: Фундаментален шаблон – основният модел за всички; шаблони за транзакции – шаблоните, реализират транзакции или дейността им изисква транзакция; шаблони за агрегация – взаимодействат с други шаблони; шаблони за планиране – участват пряко или косвено в планиране; шаблони за взаимодействие – описват взаимодействието между обектите. Те се използват за поддръжка на другите шаблони.

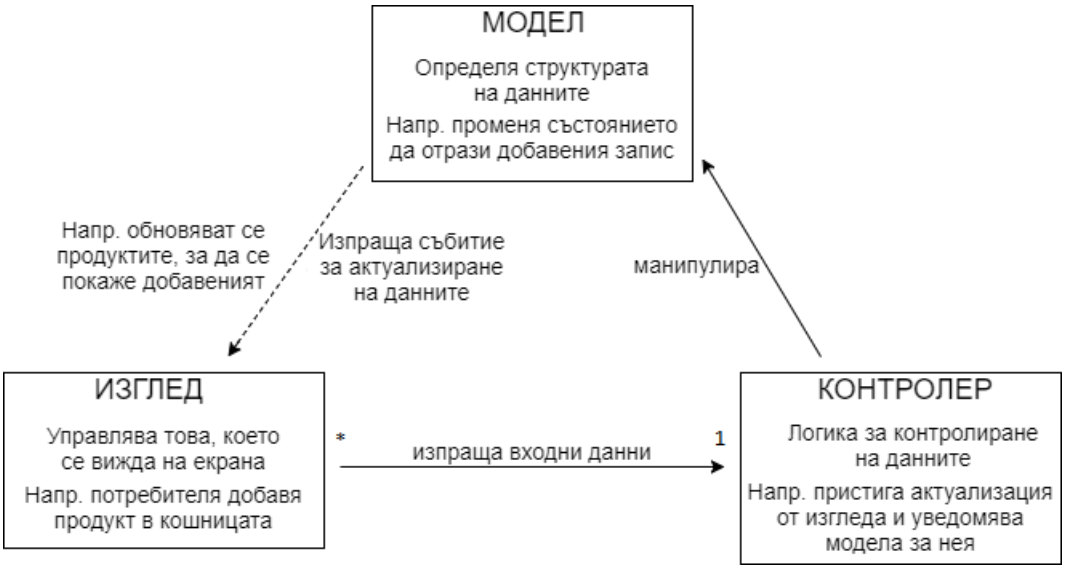
Поради специфичността си тези категории не са подходящи за общата схема за класификация.

При (Pree, 1995) класифицира шаблоните според структурата им. Той класифицира шаблоните от различни каталози, като включва и софтуерните рамки. Разделя ги в следните категории:

- базово наследяване и взаимодействие между шаблоните – основно обхващат базовите възможности за моделиране в обектноориентираните езици;
- шаблони за структуриране на обектноориентирани системи – описват начините, по който група от класове поддържат структурирането на софтуерните системи;
- шаблони, базирани на рекурсивни структури – позволяват рекурсивно построяване на йерархията от класове. Подкласовете имат препратка към себе си

или към главния (родителския) клас;

- шаблони, разчитащи на абстрактно свързване – базират се на абстрактното свързване на класовете. Абстрактното свързване е, когато обект се позовава на абстрактен клас;
- шаблони, свързани с MVC софтуерна рамка – Model-View-Controller (фиг. 4).



Фигура 4. MVC шаблон на Бушман

Класификацията на При е изключително полезна за разработчиците, когато трябва да решат обектноориентиран проблем. Тя подрежда шаблоните според модела на структурата на елементите им и позволява на потребителите след създаване на модел на системата да открият подходящ шаблон за реализацията му. Обаче, тя се отнася само до обектноориентираните шаблони и няма как да използваме критериите на При в общата схема на класификация.

Полезно би било да обобщим използваните критерии от различните класификации и на тази основа да се опитаме да ги свържем в обща схема. Предложените критерии са описани в таблица 4.

Таблица 4. *Използване на различни критерии в класификациите*

Критерии	Гама	Бушман	Зимър	Код	При
цел/функционалност/	+	+		+	
обхват	+				
мащаб		+			
проблем		+			
архитектурата на системата		+			
взаимоотношения между шаблоните			+		
според структурата си		+			+

Можем да забележим, че най-популярният критерий, който е и полезен при разработката е целта на шаблона, като всеки автор дефинира различни категории. Обаче има и критерии, които е трудно да се включат в класификацията, като взаимоотношения между шаблоните. Както споменахме почти всички автори ги определят само в рамките на създавания от тях каталог. Критерият структура е полезен, ако шаблоните се познават много добре от разработчиците, но тъй като те са много и постоянно се измислят нови, които да улеснят

въвеждането на новите технологии, няма как разработчиците да ги познават детайлно. Проблем е неизброимият брой категории по този критерий, затова избираме по-общ критерий, като предметна област. Още повече, че отделните каталози от шаблони се създават от логически обвързани шаблони в една област. Критерият архитектура на системата е подходящ само за подмножеството на архитектурните шаблони, а не за всички шаблони, т.е той определя целта на архитектурния шаблон.

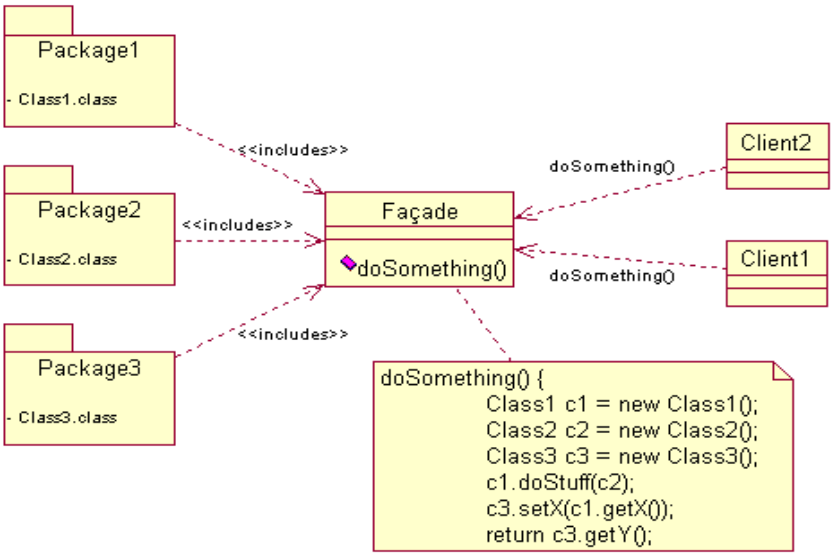
Можем да определим общите критерии и да дефинираме категориите им в схемата за класификация на база на анализа им.

Критерии за класификация на шаблоните

От таблица 4 се забелязва, че най-популярният критерий, който е и полезен при разработката, е целта на шаблона, като всеки автор дефинира различни категории. Обаче има и критерии, които е трудно да се включат в класификацията, като взаимоотношения между шаблоните, структура, проблем или архитектура на системата. Затова избираме следните основни критерии за класификация на шаблоните, идентифицирани на база анализа на публикациите на различни автори:

- **предназначение**
- **програмна парадигма (подход за разработка)**
- **слой на архитектурата или ниво на абстракция**
- **цел**
- **предметна област**

Важно е да се отбележи, че критериите не са йерархично съподчинени. Шаблоните принадлежат на няколко класификационни групи, определени по различните критерии. Например шаблонът „фасада” (фиг. 5) според целта си се определя като интерфейсен, според програмната парадигма като обектноориентиран, а според слоя на архитектурата като шаблон за проектиране. По същия начин и шаблонът „Pipes and Filters” (Тръби и Филтри) на Хомер (Homer, 2014) според слоя на архитектурата е шаблон за проектиране, според предназначението си за облачни системи, а според целта си – за осигуряване на достъп до бази от данни.



Фигура 5. Модел на шаблона „фасада” с описание на JAVA (Гама, 2004)

Предназначението е важно за шаблоните. Стремежът на създателите им е, те да имат широко приложение при разработването на софтуер. Затова предназначението им е независимо от конкретната предметна област, в която се използват, като например финансов сектор, здравна система, е-търговия. То се отнася до тясната им специализация към средата на работа – за нуждите на конкретна технология или специфичен софтуерен проблем. Шаблоните обаче все пак са създадени за клас системи с общо предназначение. Според този критерий се обособяват на шаблони за: системи, работещи в реално време; комуникационни системи; уеб приложения; разпределени системи или системи конкуриращи се за ресурси; облачни системи, IoT.

Важен критерий за разделяне на шаблоните е ***програмната парадигма***. Приемаме, че под нея се разбира парадигма на ниво подход, а не на методология за разработка. Затова тя може да бъде обектноориентирана, структурна или функционална. Всяка парадигма има собствени шаблони, които я поддържат. Например обектноориентираните шаблони разчитат на особеностите на езиците за програмиране, като наследяване и капсулиране и затова не са приложими за другите парадигми.

Според някои класификации, сред които класификацията на Бушман (Buschmann, 2001, с. 362-368), критерият слой на архитектурата разграничава шаблони от различен слой на архитектурата на системата. Има три нива на абстракция – концептуален, логически и физически модел на архитектурата на системата. Шаблоните участват в създаването на различните модели, като нивото на абстракция влияе, както върху засегнатите от тях проблеми, така и върху предложените от тях решения. Шаблоните подпомагат разработването на всеки един архитектурен модел на системата и по този начин поддържат цялостно, комплексно процеса на изграждане на информационна система.

Според Дъглас (Douglass, 2002) шаблоните се разделят в три нива – разработване на архитектурата (на концептуалния ѝ модел), разработване на механизмите (логически модел) и детайлно разработване (физически). Концептуалното ниво на архитектурата е свързано с вземане на първоначалните по-общи решения, отнасящи се до цялата софтуерната система. Разработването на логическото ниво се състои в създаването на механизми, състоящи се от елементи (например групи от класове и обекти), сътрудничещи

ISSN: 2367–738
Issue 3, pp. 70-89, September, 2018

си за постигането на цели от среден мащаб. Разработването на физическия модел се съсредоточава върху физическите особеностите, като езици за програмиране и условията на средата, в която се реализира системата. Съответно според Кардел (Kardell, 1997) шаблоните се делят на архитектурни шаблони (architectural patterns), шаблони за проектиране (design patterns) и идиоми или програмнозависими шаблони (idioms).

Шаблоните, използвани при разработването на концептуалния модел на архитектурата (архитектурни шаблони) (Buschmann, 2001, с. 25-221), определят начина на организация и взаимодействие на елементите на системата. Те поддържат цялостно структурните принципите на софтуерната архитектура. Според Ричардс (Richards, 2015) архитектурните шаблони подпомагат дефинирането на базовите характеристики и поведение на системата. Те участват в организирането на архитектурата на големи, многослойни или гъвкави системи. Пряко свързани са с новите технологии и насърчават въвеждането им в системите, например клиент-сървър или облачна архитектура. Те въвеждат базовите архитектурни принципи в системата и в повечето случаи не се влияят от програмната парадигма. Те са независими от предметната област, в която се прилагат.

Типичен представители са Service Oriented Architecture (SOA), REST (REpresentational State Transfer) и други (Bernus, 2003, с. 211-220). Към тях също така се отнася и MVC (Model-View-Controler) за уеб системите (фиг. 4).

Фаулър (Fowler, Rice, 2002) също дефинира архитектурни шаблони. Обаче, те са силно специализирани към обектноориентираната парадигма и типа система, за която са предназначени, като решенията, представени от тях, са за определени корпоративни системи. Те могат да се определят, като шаблони за различни бизнес области, т.е. бизнес отрасли, като производство, финанси, здравна система и др, които Фаулър разглежда. Съсредоточават се върху дадена бизнес структура и подпомагат въвеждането на принципите на обектноориентираната парадигма в системите. Представят компонентите на системата, като съвкупност от взаимодействащи си класове и обекти. Затова определяме тези шаблони не като архитектурни, а като обектноориентирани архитектурни шаблони. Шаблоните на Фаулър често са съчетание от няколко GoF шаблона или са развитие на някои от тях, така, че да отразяват специфичните бизнес процеси за предметната област. Силно специализирани са към типа система, за която са предназначени и решенията, представени от тях, са на ниво подсистеми. Те се свързват с разработването на логическия модел на архитектурата на системата, тъй като определят конкретните класове и обекти, които я изграждат. Независимо от името си те участват в създаването на логическия модел, като шаблони за проектиране, но са от по-голям мащаб – подпомагат създаването на подсистеми, а не на отделни модули. Друга разновидност на архитектурните шаблони са и шаблоните за анализ (Fowler, 1996), които представят определена корпоративна архитектура чрез бизнес модели. Те подпомагат създаването на бизнес моделите, но не се фокусират върху реализацията им в архитектурата на системата. Различните видове шаблони се използват при разработването на различни модели на системата. Схема за използването им е показана на фиг. 6.

Шаблоните, участващи в разработването на логическия слой на архитектурата или шаблоните за проектиране, имат по-малко влияние – само върху отделен елемент, а не върху цялостното функциониране на системата, както е при архитектурните шаблони. Подпомагат решаването на конкретен проблем на разработката на системата, например за създаване на модул или за реализиране на взаимодействието между тях. Типични представителите са шаблоните на Гама (Гама, 2004).

Шаблоните, които подпомагат създаването на технологичния (физическия) архитектурен модел, се наричат и програмнозависими. Те позволяват разработването на

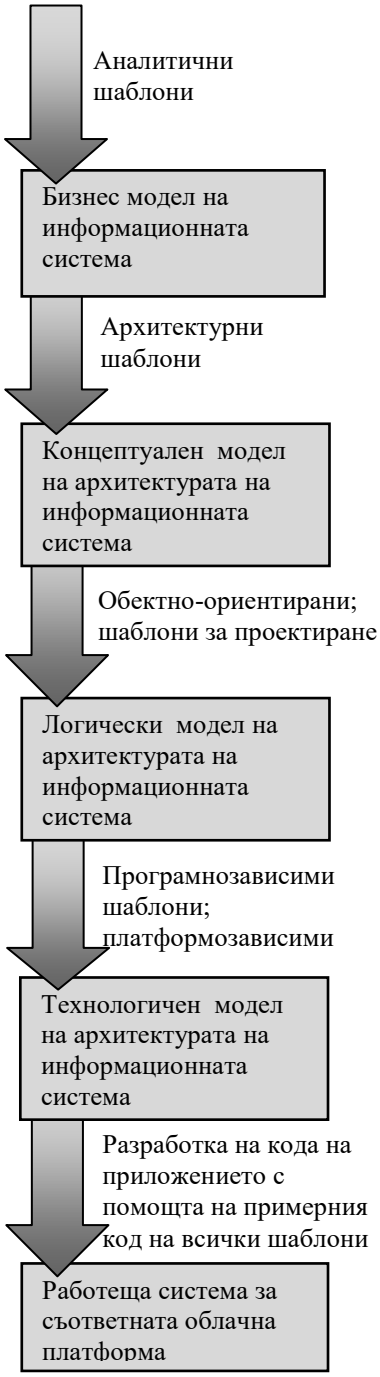
ISSN: 2367–738
Issue 3, pp. 70-89, September, 2018

архитектурата на системата от най-ниско ниво и представят модели, които отразяват особеностите на езиците за програмиране или средата на работа на системата, например особеностите на определена облачна платформа. Коплин (Coplien, 1991) предлага подобни шаблони за C++, свързани с управление на паметта и работа със стрингове, а Кент Бек (Beck, 1996) за Smalltalk, а Купър (Cooper, 1998) за JAVA.

Голяма част от програмнозависимите шаблони трудно се приспособяват към различен език от този, за който са дефинирани. Например шаблон, който се отнася до управлението на жизнения цикъл на динамичните обекти, е необходим за езици, които не поддържат автоматично освобождаване на паметта (garbage-collection), но е излишен, ако се поддържа.

Целите им са да демонстрират полезни начини за комбиниране на базови езикови концепции; да формират основа за стандартизация на използваните в програмния код структури и имена; да избегнат недостатъците за конкретен език.

ISSN: 2367–738
Issue 3, pp. 70-89, September, 2018



Фигура 6. Схема на приложение на различните шаблони при разработката на моделите на информационната система

Важен критерий е и ***целта***. Според Кардел (Kardell, 1997) тя се определя от елементите на системата, на които влияе шаблонът: интерфейс, функционалност, състояние, комуникация, достъп, физическа реализация, създаване на екземпляри, структура. Това разделение се отнася не само до шаблоните за проектиране, а и до архитектурните шаблони, които също могат да добавят цели, като например независимост на компонентите. Проблемът с този критерий е, че за разлика от другите, определените от него категории са неизброими. С развитието на информационните технологии (ИТ) те постоянно се увеличават.

Като критерий предлагаме да се добави и ***предметна област***. Въпреки, че повечето шаблони са независими от конкретната предметна област, то за част от архитектурните шаблони (обектноориентирани архитектурни шаблони и шаблоните за анализ) предметната област е ключов критерий. И тъй като те са от най-горния слой на архитектурата, изборът им оказва влияние върху всички елементи на информационна система, което определя важността на критерия. Като водещи представители са избрани шаблоните на Фаулер (Fowler, Rice, 2002; Fowler, 1996) за областта на производството, финансите, здравната система, търговията. Те съдържат шаблони за корпоративни клиенти, които решават проблеми в следните предметни области: заплати, документация за пациентите, проследяване на стоки, анализ на разходите, кредитно отчитане, застраховане, верига за доставки, счетоводство, обслужване на клиенти и търговия с валута.

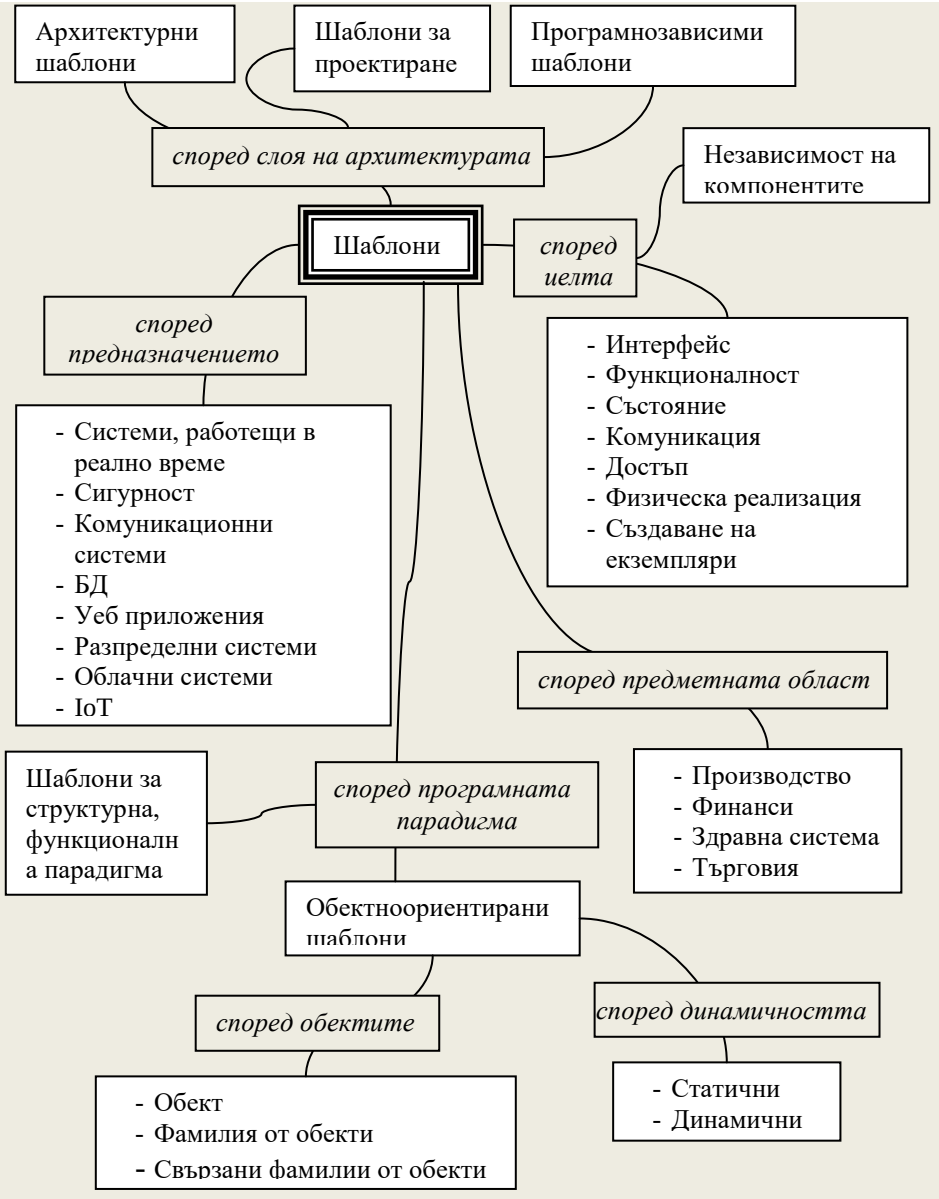
До момента са направени редица класификации на шаблоните. Част от шаблони са дефинирани за нуждите на определени фирми, разработващи софтуер и не са публикувани. Според направено от нас проучване 30% от разработчиците в проучените фирми използват подобни частни шаблони, създадени специално за предметната област на фирмата. Ето защо фокусът на изследването е да представи нова отворена за разширение схема за класификация на шаблоните според определените критерии. Те разделят шаблоните на няколко групи. Зависимостите между тези групи са показани на мисловната карта от фиг. 7.

Всички шаблони се създават с определена цел и броят на приложенията им постоянно нараства, което води и до постоянно увеличаване на категориите по този критерий. Чрез архитектурните шаблони се постигат по-крупни цели, като независимост на компонентите, което е първата категория шаблони според критерия „цел". Типичен представител е шаблонът „микро услуги" (Microservices (Lewis, 2012)).

Шаблоните за интерфейс (UI-Design Patterns (Toxboe, 2007)) са друг вид по този критерий. Те са концентрирани върху интерфейса на елементите. Основно се ползват при разработването на сайтове и мобилни приложения. Разделени са в групи според предназначението си на: социални (например шаблон за chat), за въвеждане на данни, за организиране на продуктовите каталози и продажбите и други. Но и някои обектноориентирани шаблони се причисляват към тази група, например „фасада" (фиг. 5).

Признакът „цел" може да се ползва в съчетание с признака „предназначение". Определянето на двата признака характеризира в голяма степен проблема, пред който се изправя разработчикът.

Признакът „предназначение" е основен критерий за класификация на шаблоните. Всеки каталог от шаблони обединява шаблоните с общо предназначение, но те могат да са от различен слой на архитектурата. Пример за шаблони с определено предназначение са шаблоните на Йодер и Баркалоу (Yoder, 1998), представящи различни аспекти на сигурността. Фернандес и колегите му (Fernandez, 2005; Fernandez, 1993) представят обектноориентирани модели на системи за сигурност без да ги определят, като шаблони. По-късно се появяват два нови шаблона за криптография (Braga, 1998) и за контрол на достъпа (Das Neves, 1998). В съвременните условия вече има представени цели колекции от шаблони за сигурност (Schumacher, 2006). Те се базират на йерархичната архитектура, чиито пластове определят обхвата на всеки механизъм за сигурност. Засега най-пълният каталог на шаблоните за сигурност е на Хафиз (Hafiz, 2013).



Фигура 7. Обща схема за класификация на шаблоните

Освен тях има шаблони за бази от данни (БД) (Stathopoulou, 2013; Heer, 2006), разпределени системи и системи работещи в реално време с многонишкови процеси (Sandén, 2003), комуникационни системи и уеб приложения (Lea, 1999; Schmidt, 2000; Volter, 2002), облачни шаблони (Dara, 2014; Fehling, 2014; Wilder, 2012; Somorovsky, 2011; SNIA, 2010).

Понякога за шаблоните, създадени с едно предназначение може да се открие и друго. За облачните изчисления могат да се използват шаблоните на Хоф и Уолф (Hohpe, 2004) за интегриране на компоненти посредством размяна на съобщения или тези на Хамнер (Hanmer, 2007) за устойчив на сривове софтуер. Могат да се ползват и шаблоните за сигурност на Шумахер (Schumacher, 2006).

В различни класификации, могат да се намерят и други признаци, обаче те не класифицират цялото множество от шаблони, а се отнасят за определен тип. Например обектноориентираните шаблони могат да се разделят и според обектите си. Това

разделение отразява видовете същности, за които се прилага шаблонът – обект, фамилия от обекти или свързани фамилии от обекти. Шаблоните с предназначение обект са подходящи за един обект. За фамилия от обекти реализират функционалност за група обекти, които не е необходимо да са обвързани с наследяване, а за свързани фамилии от обекти се прилагат за група от обекти, част от които са обвързани с наследяване.

Критерият „динамичност" показва възможността на шаблона за поддържане на промяна на решението по време на изпълнение. Определят се две категории – статични и динамични. Критерият се специализира според парадигмата на шаблоните. Например при обектноориентираните шаблони Гама предлага разделение на шаблоните според динамичността им на шаблони за класове и за обекти. Представители на статичните шаблони за класовете са „метод фабрика" (Factory Method), „шаблонен метод" (Template Method) и „интерпретатор" (Interpreter), а на динамичните шаблони за обекти са почти всички други, като „наблюдател" (Observer), „фасада" (Facade), „стратегия" (Strategy).

Може да се дефинират и други категории, като подходящи структури от данни – списъци, дървета, но тъй като шаблоните са абстрактни, използването на конкретна структура, ограничава приложението им.

Предложената схема на класификация е абстрактна. За да изпълни целта си – да подпомогне разработчиците при търсенето на шаблоните за определена технология, тя може да се специализира. Когато разработчиците се опитват да открият подходящи шаблони, те правят предварително проучване и могат да открият специфични категории. Например шаблоните за облачните изчисления според предназначението си могат да се разделят на шаблони за облачната среда и за приложенията, работещи в нея. Класификация на примерни шаблони, познати на автора, според дефинираната схема на фиг. 7 е представена на фиг. 8.

Фигура 8. Примерна класификация на шаблоните

Има една особена група шаблони, която оставяме извън класификацията. Това са т.нар. анти-шаблони (Koenig, 1995), които са разпространени практики. Важна разновидност на интерфейсните шаблони и анти-шаблоните са "Dark Pattern" (Brignull, 2017; Jenkins, 2013; Roseblade, 2014) или тъмни, подмолни шаблони, чиято цел е да подведат крайния потребител да извърши действия, противоречащи на интересите му. Те са създадени с участието на психолози и се използват съзнателно от разработчиците. Идеята е потребителят лесно да приеме нещо нежелано, а после много трудно да го откаже. Популярен представител е шаблонът „мотел за шарани" (Roach Motel, Snyder, 2012). Общият им формат позволява на разработчика бързо да се ориентира и определи, подходящите за дадения проблем. Така се подпомага получаването на качествен резултат от процеса на прилагането им. Качествената документация на шаблоните намалява разходите, времето и трудността при разработката на системите. Създадените с тях приложения са с по-лесни за съпровождане, т.е. с по-дълъг жизнен цикъл, гъвкави и преносими.

4. СЪТРУДНИЧЕСТВО

Използвани са класификациите на шаблоните на различни учени, но те са цитирани в текста.

5. ЛИТЕРАТУРА

Статия

Bernus, P., (2003). *Enterprise models for enterprise architecture and ISO 9000:2000.* н.м.: Annual Reviews in Control 27 (2003), p. 211-220.

Braga & Rubira & Dahab & Tropyc, (1998). *A pattern language for cryptographic object-oriented software.* н.м.:C16 in Pattern Languages of Program Design 4, Procs. of PLoP'98.

Brignull, H., (2017). *Dark Patterns.* [Онлайн] Available at: darkpatterns.org (08.08.2017)

Buschmann, F. & Meunier, R., (1995). *A System of Patterns, In Coplien, J. O., Schmidt, D. C., Pattern Languages of Program Design*. н.м.: Addison-Wesley, p. 325-343.

Dara, S., (2014). *Privacy Patterns in Public Clouds.* н.м.: Proceedings of the Indian Conference on Pattern Languages of Programs (GuruPLoP).

Fernandez & Larrondo-Petrie & Gudes, (1993). *A method-based authorization model for objectoriented databases.* н.м.: Proc. of the OOPSLA 1993., Workshop on Security in Object-oriented Systems , 70-79.

Fernandez & Larrondo-Petrie, (2005). *Teaching a course on data and network security using UML and patterns.* н.м.: Procs. of the Educators Symposium of MoDELS/UML 2005, Montego Bay, Jamaica.

Hanmer, R., (2014). *Patterns for Fault Tolerant Cloud Software*. н.м.: Proceedings of the Conference on Pattern Languages of Programs (PLoP).

Heer, J., Agrawala, M., (2006). *Software Design Patterns for Information Visualization.* н.м.: IEEE Transaction on visualization and computer graphics, vol. 12, No. 5.

Jenkins, S., (2013). *Dark Patterns: The questionable art of boosting conversion rates* [Онлайн] Available at: https://www.gadgetdaily.xyz/dark-patterns-the-questionable-art-of-boosting-conversion-rates/ (09.09.2017)

Kardell, M., (1997). *A Classification of Object-oriented Design Patterns. Master's thesis, Department of Computing Science.* н.м.: Umeå University. Available at: http://www8.cs.umu.se/~jubo/ExJobbs/MK/patterns.htm (19.07.2017)

Koenig, A., (1995). *Patterns and Antipatterns*,. н.м.: JOOP 8(1): p. 46-48.

Lewis, J., (2012). *Micro services - Java, the Unix Way.* н.м.: conference 33rd Degree conference for Java masters, March 2012, Krakow. Available at: http://2012.33degree.org/talk/show/67

Reinfurt, L. & Breitenbücher, U. & Falkenthal, M. & Leymann, F. & Riegg, A., (2017), *Internet of Things Patterns for Devices.* н.м.: Ninth international Conferences on Pervasive Patterns and Applications (PATTERNS), XPS, p. 117-126.

Roseblade & Owen, (2014). *Dark Patterns are Everywhere: how to Stay Informed and Keep them Out of your Nonprofit Website*. [Онлайн] Available at: http://whitefusemedia.com/blog/dark-patterns-are-everywhere-how-stay-informed-and-keep-them-out-your-nonprofit-website (19.05.2017)

Sandén, B. I., (2003). *Entity-life modeling: modeling a thread architecture on the problem environment.* н.м.: IEEE Software.

Snyder & Jesse, (2012). *Dark Patterns in UI and Website Design*, 10 Sep 2012, [Онлайн] Available at: http://webdesign.tutsplus.com/articles/dark-patterns-in-ui-and-website-design--webdesign-8506 (19.08.2017)
Somorovsky, J. & Heiderich, M. & Jensen, M. & Schwenk, J. & Gruschka, N. & Lo Iacono, L., (2011). *All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces.* н.м.: Proceedings of the 3rd ACM workshop on Cloud computing security workshop (CCSW), Chicago, IL, USA, 17–21 October 2011.
Toxboe, (2007). *A., User Interface Design patterns.* Available at: ui-patterns.com (08.08.17)
Yoder, J. & Barcalow, J., (1998). *Architectural patterns for enabling application security.* н.м.: Procs. PLOP'97.
Zimmer, W., (1995). *Relationships between Design Patterns, A System of Patterns, In Coplien, J. O. & Schmidt, D. C., Pattern Languages of Program Design.* н.м.: Addison-Wesley, 345-364.

Книга

Beck, K., 1996. *Smalltalk Best Practices.* н.м.: Prentice-Hall.
Buschmann, F. & Meunier, R. & Rohnert, H. & Sommerlad, P. & Stal, M., (2001). *Pattern-Oriented Software Architecture – A System of Patterns.* н.м.: John Wiley & Sons, 2nd Edition.
Buschmann, F. & Henney, K., (2005). *Pattern-Oriented Software Architecture – On Patterns and Pattern Languages.* н.м.: John Wiley & Sons.
Coad, P. & North, D. & Mayfield, M., (1995). *Object Models: Strategies, Patterns, and Applications.* н.м.: Englewood Cliffs, Prentice Hall.
Coplien, J. O., (1991). *Advanced C++ – Programming Styles and Idioms.* н.м.: Addison-Wesley.
Cooper, J., (1998). *The Design Patterns, Java Companion.* н.м.: Addision-Wesley.
Das Neves, Garrido, (1998). *Pattern Languages of Program Design 3.* н.м.: Addison-Wesley.
Douglass, (2002). *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems.* н.м.: Addison-Wesley, Boston, MA, USA.
Fehling, C. & Leymann, F. & Retter, R. & Schupeck, W. & Arbitter, P., (2014). *Cloud computing patterns.* н.м.: Springer-Verlag Wien. ISBN 978-3-7091-1567-1
Fowler, M., (1996). *Analysis Patterns.* н.м.: Addison-Wesley Professional.
Fowler, M. & Rice, D. & Foemmel, M. & Hieatt, E. & Mee, R. & Stafford, R., (2002). *Patterns of Enterprise Application Architecture.* н.м.: Addison-Wesley.
Hafiz, M., (2013). *Security Pattern Catalog.* [Онлайн] Available at: http://www.munawarhafiz.com/securitypatterncatalog/index.php (09.08.17)
Hanmer, R., (2007). *Patterns for Fault Tolerant Software.* н.м.: Wiley.
Hohpe, G. & Woolf, B., (2004). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* н.м.: Addison-Wesley: Reading, MA, USA.
Homer, A. & Sharp, J. & Brader, L. & Narumoto, M. & Swanson, Tr., (2014). *Cloud Design Patterns.* н.м.: Microsoft. 978-1-62114-036-8.
Lea, D., (1999). *Concurrent Programming in Java, in Concurrent Programming in Java: Design Principles and Patterns, Second Edition.* н.м.: Addison-Wesley.
Pree, W., (1995). *Design Patterns for Object-Oriented Software Development*. н.м.: Addison-Wesley.
Richards, M., (2015). *Software Architecture Patterns.* н.м.: O'Reilly Media.
Schmidt, D. C. & Stal, M. & Rohnert, H. & Buschmann, F., (2000). *Pattern-Oriented Software Architecture – Patterns for Concurrent and Networked Objects.* н.м.: John Wiley & Sons.
Schumacher, M. & Fernandez, E. B. & Hybertson, D. & Buschmann, F. & Sommerlad, P., (2006). *Security Patterns: Integrating Security and Systems Engineering.* н.м.: John Wiley & Sons.
Stathopoulou, E. & Vassiliadis,P., (2013). *Design Patterns for Relational Databases, University of Ioannina.* [Онлайн] Available at: http://www.odbms.org/wp-content/uploads/2013/11/PP2.pdf (09.06.17)

Storage Networking Industry Association (SNIA), 2017. *Cloud Data Management Interface (CDMI)*. [Онлайн] Available at: https://www.snia.org/cdmi (06.09.2017)
Volter, M. & Schmid, A. & Wolff, E., (2002). *Server Component Patterns – Component Infrastructures Illustrated with EJB.* н.м.: John Wiley & Sons.
Wilder, B. (2012). *Cloud Architecture Patterns Develop cloud-native applications.* н.м.: O'Reilly Media.
Гама, Е. & Хелм. Р. & Джонсън, Р. & Влисидес, Дж., (2004). *Шаблони за дизайн.* н.м.: СофтПрес.

6. СЪКРАЩЕНИЯ

1. GoF – Gang of Four, Голямата четворка. Това е групата на Гама, Хелм, Джонсън и Влисидес (Гама и др., 2004). Те въвеждат съвременното разбиране за шаблоните за проектиране и предложени от тях 24 шаблона са класическите обектноориентирани представители.

2. MVC шаблон – Model-View-Controller е специфичен шаблон, получен като обединение на три шаблона. Той е представя модел на архитектурата на уеб приложенията.

3. SNIA – Storage Networking Industry Association е асоциация, която създава стандартите за областта на съхраняване на данни

4. SOA – Service-Oriented Architecture е архитектура, според която приложението се декомпозира на услуги. Тя има множество предимства. 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.

5. REST – REpresentational State Transfer е архитектурен модел за реализация на уеб услуги.

6. БД – Бази от Данни

7. ИТ – Информационните Технологии