Упражнения 2 – 3 - 4. Какво е Use Case. Използване на StarUML за създаване на UML Use Cases. Use Cases в „Спецификацията на Изискванията” към софтуера (SRS)

Цел: Запознаване с предназначението и процеса на построяване на UML Use Case диаграми.

1. **Какво е Use Case**

“Use cases” е техника за определяне на **функционалните изисквания** на една система.

“Use case” (Случай на употреба, Случай на използване, Вариант на използване, Прецедент и др.) е **независима** част от функционалността на моделираната система, притежаваща **резултантна ценност** за своите изпълнители (актьори).

- "Независимост" - ако един случай за използване винаги се изпълнява заедно с някой друг, то по всяка вероятност той трябва да бъде включен в другия.
- "Резултантна ценност" на случая за използване означава, че той трябва да носи за актьора някакъв завършен, ценен от гледна точка на бизнеса резултат, тоест да стане по-ефективен, по призводителен бизнеса на актьора.

Множество от „случаи на използване” описва особеностите на **поведението** на моделираната система, без да се разглежда нейната вътрешна структура.

Един “Use case” се изобразява като **елипса**, в която е поставено наименование, започващо с главна буква: във вид на **съществително** (а) или **глагол** (б) с **пояснителни думи**.

Когато говорим за случай на употреба, не може да не говорим и за сценарий на случая на употреба. Един “Use case” е **набор от сценарии**, свързани помежду си от обща потребителска цел. **Сценарият** (scenario) представлява серия от стъпки, описващи взаимодействието между потребител и система. Сценариите биват два вида:

- **основен сценарии на успех** (main success scenario)

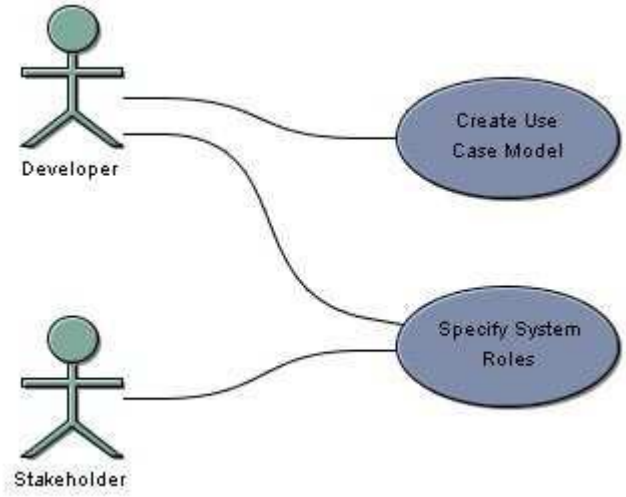
и

- **разширен сценарий** (extensions scenario).

2. **Какво е актьор?**

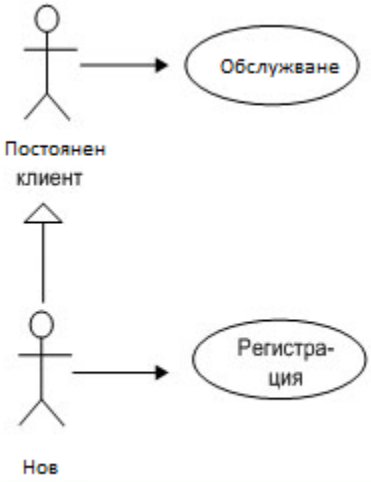
Актьорът (actor) е роля, която един потребител изпълнява по отношение на системата. Всеки use case има основен актьор, който изисква от системата предоставяне на услуга. Основният актьор е актьорът с целта, която use case предписанието трябва да задоволи, и обикновено, но не винаги, е инициатор на use case. Може да има и други актьори, с които системата комуникира в рамките на use case.

Актьорите изпълняват случаи на употреба. Когато един актьор изпълнява даден случай на употреба, то между актьора и случая на употреба има връзка от тип „Асоциация” („Association”), която графично (в диаграмите на потребителските случаи) се представя чрез плътна връзка без стрелка, най-често **по следния начин:**



В горната диаграма човечетата са актьорите, а елипсите – случаите на употреба. И двамата потребители разработчик и спонсор са отговорни за определяне на ролите на системата, но само разработчика създава модела.

Връзката **между актьор и случай на употреба може да е от тип «**Насочена асоциация» (т.е. със стрелка). Стрелката е към този, от когото се очаква да извърши услугата.
Един актьор може да изпълнява много случаи на употреба. Също – един случай на употреба може да се изпълнява от много актьори, но няма случай на употреба без актьор. Някои автори считат, че терминът актьор не е правилен, а трябва вместо него - да се използва роля.
Всяка стъпка от сценария, наречена още **use case предписание**, е елемент на взаимодействие между актьора и системата.
Единствената връзка между актьори е „обобщаване" (Generalization), което на практика означава, че актьорите могат да се наследяват един друг. При наследяването, потомъкът получава в наследство всички случаи на използване на своите предци, но има и собствени. Обозначава се с куха стрелка.

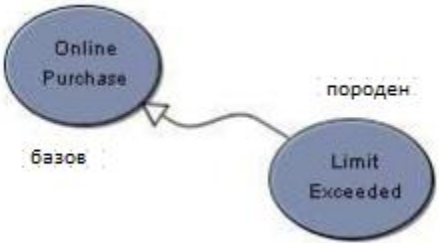


Има четири типа връзки между случаите на използване:

- **включва** (<<include>>) - когато един use case задължително използва функционалността на друг use case. Стрелката сочи включвания use case. Вие може да включите такава връзка във вашия модел за да покажете следните ситуации:
  - Поведението на включения use case е общо за 2 или повече use cases.
  - Резултатът от поведението на включения use case, а не поведението му, е важен за базовия use case.

- **разширява** (<<extend>>) - вие можете да използвате extend relationship за да специфицирате, че един use case (т.е extension use case) **може** (но не е задължително) да разшири поведението на друг use case (base use case). Тази връзка показва, че изпълнението на разширението (т.е на extension use case) зависи от това, което се случва, когато базовия use case се изпълнява, тоест разширението не е задължително да се изпълни, то е опционално и зависи от изпълнението на базовия клас. Вие може да специфицирате няколко разширения ("Extension points:") на един единствен базов use case. Стрелката сочи базовия use case.

- **обобщава** (Generalization) – Използвайте генерализация, когато описвате вариации на нормалното поведение, а нормалното поведение е - в базовия случай на употреба. Връзка от този тип се визуализира със плътна линия и куха стрелка, като линията тръгва от вариацията и сочи към базовия на употреба.



- **зависи** (Dependency): връзка, при която единият use case (да го наречем „**клиент**"), използва или **зависи** от друг елемент („**доставчик**"). Ако доставчикът бъде променен, тогава и клиентът **може** (но не е задължително) да бъде засегнат, но обратното не е в сила. Обикновено тези връзки нямат имена и се представят с пунктирана отворена стрелка.

„Клиент" зависи от „Доставчик"

Виж. „Програмни спецификации . Ръководство за лабораторни упражнения” тема III и тема IV.

Диаграми на случаите на употреба

Диаграмите на случаите на употреба са начин за визуализация на случаите на употреба в UML. Това са поведенчески диаграми. Всеки случай на употреба (use case) е описание на отделен аспект от **поведението** на системата.

Разработката на UML Use Case диаграми – това е първият етап от процеса на инженеринг на изискванията, който има за цел - да се извлече набор от функционални изисквания за поведението на проектирана система. Получените в резултат диаграми са средство за комуникация между експерти, потребители и изпълнители и основни артефакти на **Спецификацията на изискванията към софтуера** (SRS). Спецификацията на изискванията към бъдещата система под формата на диаграми на варианти на използване (на различни нива на абстракция, започвайки от концептуално ниво) и допълнителни сценарии може да бъде един самостоятелен модел на ПС, който в езика UML получи название <<useCaseModel>> („Модел на Случаи на Използване”).

Пример 1. Постъпково реализиране на Use case диаграма

Нека постъпково да реализираме и анализираме Use case диаграма за управление на „**Система за електронно обучение**”. В случая нашата основна цел е да определим различни нива на достъп до системата на електронно обучение.

Последователност на създаване на диаграмата на прецедентите :

- да бъдат анализирани потенциалните **актьори в системата**, които всъщност са участници в системата
- да се определят **Use case**, т.е. задачите които трябва да бъдат изпълнени, за реализацията на потребителската цел.
- да се определи кои актьори, какви **Use cases** изпълняват
- текстово описание на всеки **Use case**

1. Идентифициране на потенциалните елементи на системата.

И така нека да определим потенциалните актьори на системата. Важно е да се разбере, че под актьори имаме предвид различните участници в системата, като не е задължително това да са хора. Възможна е ситуация, в която един човек изпълнява две различни роли за системата, както и друга – актьорът е система, която взаимодейства с разработваната система.

На пръв поглед се различават следните **основни елементи (обекти) на разработваната система**:

- Курсове (т.е учебни дисциплини) и Теми, които представляват елементи на курса;
- Лектор, който води курса;
- Администратор на системата, който може да редактира курса, да създава нов курс и да го управлява;
- Календарен график на курса;
- Студенти, които ръководят календарния график на курса, в който участват.

2. Идентифициране на актьорите в системата за електронно обучение.

Сега трябва да определим от горе описания списък само значещите **актьори**. Те са:

- Администратор на системата;
- Лектор;
- Студент.

Главният актьор за нашата диаграма е администраторът на системата, а лекторът и студентът са допълнителни актьори.

3. Идентифициране на Use case.

Тук трябва да се опитаме да определим потенциалните **бизнес процеси**, т.е. процесите, които трябва системата да изпълнява за да може да реализира различните нива на достъп до нея, т.е. да удовлетвори поставената цел.

И така **основните задачи** за системата са:

- Управление на курс;
- Управление на назначение на курс.

Като анализираме по-нататъшното развитие на системата, може да определим някои подпроцеси, без които основните процеси не могат да съществуват.

За актьор – „Администратор на системата": за да управлява един курс, той трябва да може да разгледа съществуващите вече курсове, както и съдържаната в тях информация, лекторите, които ги ръководят, както и тяхната продължителност. Освен това„Администратор на системата" трябва да има възможността да редактира информацията, съдържаща се в даден курс, да може да създава курс и да назначава лектор за курса. Също така, той трябва да може да редактира служебната информация за самите лектори, както и да проверява дали вече някой лектор е въведен в системата, и ако не е въведен, да може да направи това. Ако някой от лекторите вече не води никой от курсовете, то тогава - да може да изтрива профила му от системата.

И така след тези разсъждения достигнахме до следните use case:

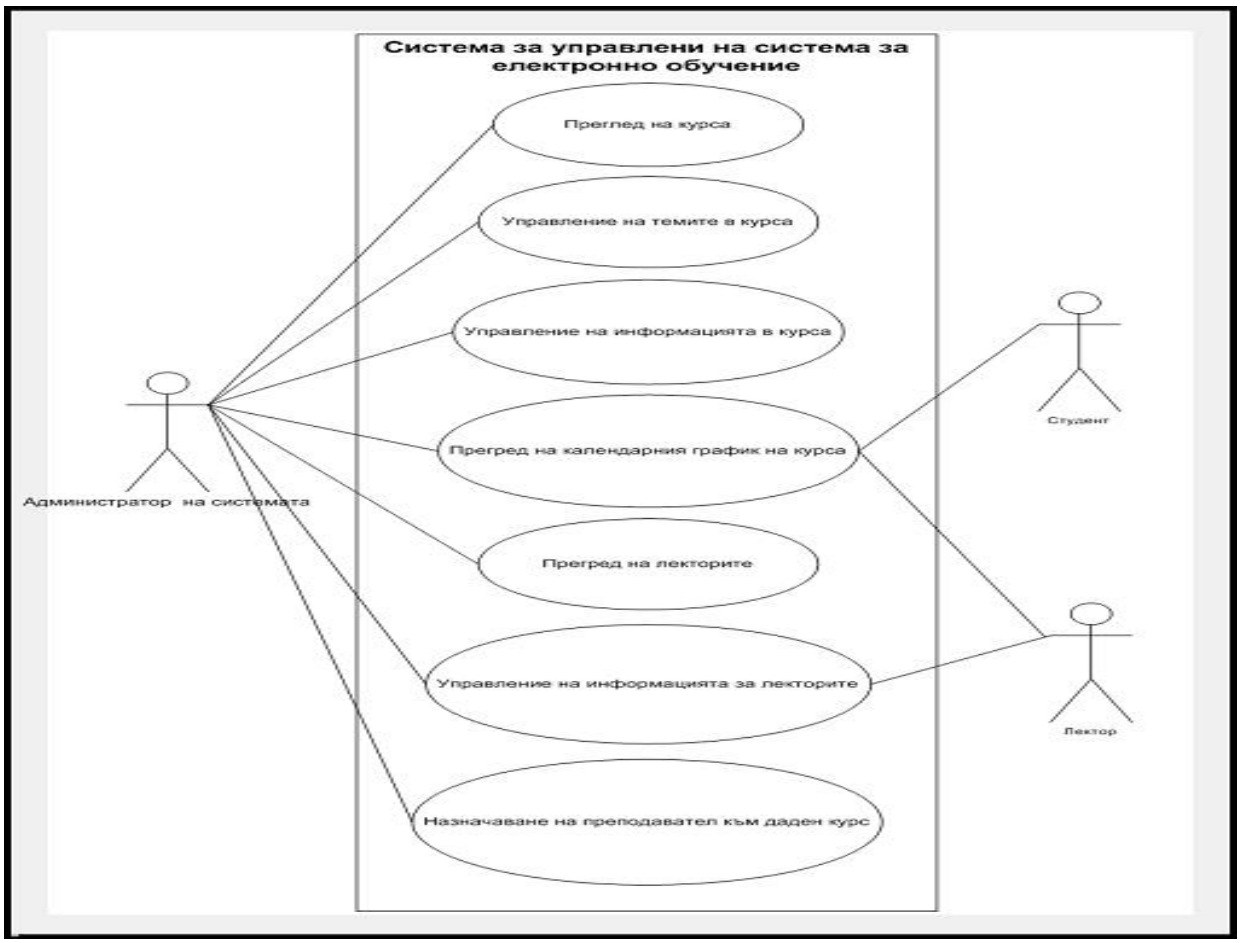
- Преглед на курсовете;
- Управление на тематичното съдържание в курса;
- Управление на информацията за курса;
- Преглед на календарния график на курса;
- Управление на информацията за лекторите;
- Назначаване на преподаватели към даден курс.

Ако сега се направи анализ на морфологичния състав на горе изложените изречения, много лесно подлогът може да се възприеме като актьор, а сказуемите и техните описателни допълнения да се разглеждат като use case. Често това е едно добро начало за разработване на use case диаграми.

Добра практика е оформянето на таблица с 2 колони: 1-та - **use case**, а във 2-та - актьорите, участващи в него.

use case	**Актьори**
Преглед на курсовете	Администратор на системата
Управление на тематичното съдържание в курса	Администратор на системата
Управление на информацията за курса	Администратор на системата
Преглед на календарния график на курса	Администратор на системата, Лектор, Студент
Управление на информацията за лекторите	Администратор на системата, Лектор
Назначаване на преподаватели към даден курс.	Администратор на системата

Въз основа на таблицата лесно се създава диаграмата. Ето и как ще изглежда нашата use case диаграма:



Правоъгълникът обозначава граница на системата (или на подсистема). В случая – може да се пропусне, т.к. се подразбира интуитивно. Актьорите са представени без техните атрибути, които при програмирането ще бъдат обособени в клас. Връзките между актьори и прецеденти винаги са от тип **1:1**, но 1 актьор може да се свърже с няколко прецедента с отделни връзки.

Пример 5. „Обслужване на клиенти по телефонни заявки”

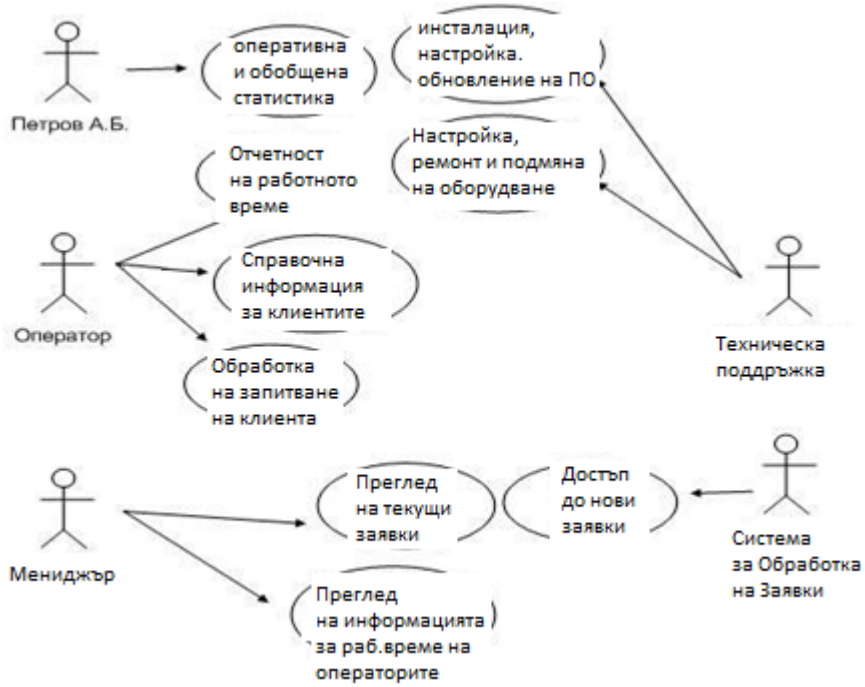
В качеството на пример при разглеждане на различните UML диаграми по време на лекции, често ще бъде използван един пример: система "**Обслужване на клиенти по телефонни заявки**"

Възложител на дадената система - това е компания, владееща мрежа от продуктови магазини. Възложителят изисква да се разработи система, предоставяща услугата „**обслужване на клиенти по телефонни заявки**” (ОКТЗ). Всеки клиент трябва да може да се регистрира в компанията, след което да може по телефона, в удобно за себе си време да прави поръчка на стоки, които да му се носят у дома, където той ще се разплаща. За тази цел, в компанията ще се организира локален телефонен център, състоящ се от многоканална АТЦ, щатни оператори и съответното ПО (което е обект на възлагане). При това, компанията вече разполага с **информационна система за Системата за Обработка на Заявките (СОЗ)** и новата поръчкова система (ОКТЗ) трябва да бъде интегрирана с нея.

Преди да се захване с **реализация на описаната по-горе задача, всяка сериозна софтуерна компания се заема с уточняване на изискванията. Резултатът е - Спецификацията на Изискванията” към софтуера (SRS).**

Дейността по анализ на изискванита включва **опит за преценка какво точно потребителите очакват от разработения софтуер и какво самите разработчици трябва да направят**.

Тук може да се използват **Use Cases, които да опишат на най-високо ниво целите на потребителите на системата, цели, които разработваната система трябва да реализира**. Тези цели не е необходимо да са задачи или действия, а могат да са по общи изисквания към **функционалността на системата**.
За конкретния пример, функционалните изисквания на ПС могат да се опишат със следната UML диаграма на случаи на използване:



Пример на диаграма на случаи на използване

На тази диаграма са обозначени **следните видове потребители**: оператор, мениджър и представители на техническата поддръжка.
Системата трябва също да поддържа външен интерфейс със Системата за Обработка на Заявките (СОЗ). Това е - четвърти потребител.
Има и още един потребител на системата, а именно Петров А.Б. - директор на департамента за продажба на стоки, който желае периодично да следи дейността на телефонната служба за приемане на заявки. За него е създадено специално работно място с екранни форми и статистики.

Препоръки при разработка на use case диаграмите

Както беше отбелязано по-горе, **едно от основните предназначения на тази диаграма е за определяне и формализация на функционалните изисквания на системата**. Тази диаграма може да служи като основа за съгласуване на функционалните изисквания на системата с клиента, при ранните етапи на проектиране на системата. Всеки **use case** на даден етап на разработка може да бъде подложен на декомпозиция при следващите етапи. Препоръчително е, общия брой актьори да не превишава **20**, а броят на случаите на използване - да не надвишава **50**. В противен случай моделът губи своята яснота.
За разработка на **use case д**иаграми се препоръчва следната последователност от действия:

- Да се определят главните или първичните и второстепенни актьори
- Да се определят целите на главните актьори по отношение на системата
- Да се формират основни варианти на използване, които специфицират функционалните изисквания към системата
- Да се подредят вариантите на използване в низходящ ред на риска за тяхната тяхната реализация
- Да се разгледат вариантите на използване в низходящ ред на риска за тяхната тяхната реализация

- Да се определят участниците, интересите, предусловията и постусловията при изпълнение избран вариант на използване
- Да се напише успешния сценарий за реализация на избрания вариант за използване
- Да се определят «Изключения» или «неуспех» при изпълнение на *сценария на* варианта на използване
- Да се напише сценарий за изпълнение на всички изключения
- Да се определят общите варианти на използване и да се изобразят техните взаимовръзки с базовите варианти (със стереотип <<include>>)
- Да се определят варианти на използване, свързани с изключенията и да се изобразят тяхните взаимовръзки с базовите, като стереотип <<extend>>
- Да се провери диаграмата за отсъствие на дублиране на варианти на използване и актьори

Кога да използваме случаи на употреба?

- Те са ценен инструмент за разбиране функционалните изисквания на системата.
- Те са външен изглед на поведението на системата, така че не очаквайте корелация между тях и класовете на системата.
- Те са основен инструмент за „уточняване" на изискванията при планирането и контролирането на итеративно разработвани софтуерни проекти. „**Улавянето**" на случаите на употреба е една от основните задачи през фазата на уточняване и специфициране на изискванията към софтуера.
- Повечето от случаите на използване ще се генерират по време на тази фаза на проекта, но някои ще бъдат открити и по късно, през следващите фази на разработка на проекта. Така, че: мислете за тях, през цялото време, т.е през целия жизнен цикъл на ПО. Всеки случай на използване е едно потенциално изискване, и докато не сте „разбрали изискването", не можете да планирате да се справите с него. Обсъждането с потребителите е от голяма полза за тяйното уточняване!
- При разработка на случаи на употреба се концентрирайте върху **текстовото описание**, а не върху **диаграмата**!

Теми за реферат (презентация):

Software Cost Extimation (методи)
Software Quality (метрики)
Agile Software Development

Виж. „Програмни спецификации . Ръководство за лабораторни упражнения" тема III и тема IV.

Използвана Литература:

Мартин Фаулър „UML основи"
http://staruml.sourceforge.net/docs/user-guide(en)/toc.html
http://194.141.86.222/lecture/rkraleva/LetenSem/SoftTech/Ypr_SoftTech/st10.htm
http://cs.calvin.edu/curriculum/cs/262/kvlinden/04analysis/lab.html
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML
http://www.umljokes.com