Упражнение 1. Унифицираният език за моделиране (UML) - въведение. Изграждащи елементи на UML, UML перспективи и инструментални средства и StarUML. Разглеждане на съществуващ UML проект.

Цел: запознаване с UML – предназначение, изграждащи елементи. Средства за създаване на UML диаграми. StarUML – запознаване и работа с готов UML проект.

1.1. UML - Въведение. Изграждащи елементи на UML. UML перспективи.

UML - Unified Modeling Language (**унифициран език за моделиране**) представлява фамилия от графични нотации (на Grady Booch, Ivar Jacobson и Jim Rumbaugh), обединени от общ мета-модел, които помагат при специфицирането (описанието), визуализирането и документирането на изискванията и архитектурата на софтуерни системи, най-вече **обектно-ориентирани**.
Създаден е през 1997г. (UML 1.0), UML е сравнително нов **стандарт за моделиране на софтуер**, контролиран от консорциума от компании Object Management Group (OMG). Текущата версия на UML е 2.4.1 (2011).
UML е „**универсален” език за моделиране** на ПО, той не е ориентиран към конкретна методология на разработка, нито към конкретен език за програмиране.
Модел – опростено представяне на обекта за моделиране. Моделът ни позволява да се абстрахираме от някои свойства на обекта и да се съсредоточим само върху някои от тях. Например, за да се създаде информационна система за автоматизация на предприятие се строи модел на предприятието, който се фокусира върху бизнес процесите, потоците на данните и бизнес роли. Този модел не включва такава информация като междуличностните отношения на работниците и служителите, подробности за планиране офис пространство, график на работа (работа, време обяд, събота и неделя), и т.н. При компютърна симулация, чрез „задействане на модела” се придобиват знания за характеристиките на обекта.
UML е език **за „визуално” моделиране**. Визуалното моделиране (visual modeling) се явява метод, който:

- Използва графичен (много често базиран на графи) модел за визуализация на ПО;
- Предлага моделиране на предметната област от различни гледни точки (перспективи);
- Може да се използва както при разработката, така и при еволюция на ПО, а така също при различните дейности на създаването на ПО.

UML дефинира нотация и мета-модел:
- Нотацията (notation) – това са изграждащите елементи в UML моделите, т.е. това е графичният синтаксис на езика за моделиране.
- Мета-модел (meta-model) - Няма точна дефиниция за това какво е мета-модел. **Мета-моделът** на UML е нещо като граматиката на UML, това са правилата за свързване на изграждащите елементи. По същия начин, както например граматиката на Java определя структурата на програмите на Java, по подобен начин и UML мета-моделът дефинира правилата за изграждане на всички видове UML диаграми.
UML **не е език за програмиране**, но инструментални средства (CASE-средства) могат да се използват за генериране на код на различни езици от UML диаграма на класове.
Основното предназначение на повечето CASE-средства обаче е: поддръжка и повишаване ефективността на процеса на моделиране на ПО. Налице е изобилие от CASE-средства (има и безплатни), поддържащи нотация на езика UML 1.4-1.5 и UML 2.x и обезпечаващи интеграция, включително права и обратна генерация на код на програми, с най разпространените езици и среди за програмиране, като MS Visual C++, Java, Object Pascal/Delphi, Power Builder, MS Visual Basic, Forte, Ada, Smalltalk, C#.

1.2. Изграждащите елементи на UML

Изграждащите елементи на UML могат да бъдат класифицирани на:

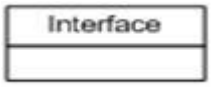
- Things (инструменти),
- Relationships (връзки) и
- Diagrams (диаграми)

(1) Things (инструменти). Това са най-важните изграждащи елементи на UML. Те могат да бъдат:

- **Структурни (Structural)** – дефинират статичната част на модела. Те представляват физически и концептуални елементи, например:

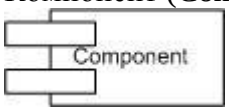
Class - представлява набор от обекти с подобни отговорности:

Interface - определя набор от операции, които определят отговорността на класа.

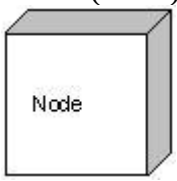


Случай на използване (Use case): Представлява множество от действия (actions), които системата изпълнява със специфична цел.

Компонент (Component): Описва физическа част на системата



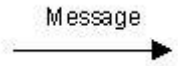
Възел (Node): Физически елемент, съществуващ повреме на изпълнение на системата (at run time).



и др.

- **Поведенчески (Behavioral) –** касаят динамиката в UML моделите, например:

Взаимодействие-то (Interaction) - дефинира се като поведение, което се състои от група обменени съобщения между елементите, за изпълнение на конкретна задача



Машина на състояние (State machine):

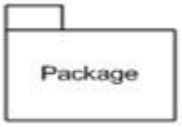
"Машина на състояние" – описва последователността на състоянията през които един обект минава в отговор на събития. Събитията са външни фактори, отговорни за промяната на състоянието

- **Групиращи (Grouping)**

Групиращите инструменти, могат да се определят като механизъм за групиране на елементи на UML модел в едно цяло. Налице е само един такъв инструмент за групиране:

Пакет (Package):

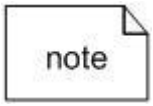
Пакетът е единственото нещо за групиране, което на разположение в UML, за събиране в едно, на структурни и поведенчески инструменти.



- **Коментарни (Annotational)**

Коментарните инструменти (Annotational) могат да се определят като механизъм за поставяне на бележки, описания и коментари на UML модели елементи. "Бележка" (Note:) е единственият такъв елемент в UML.

"Бележка" (Note): Използва се за да се направят коментари, да се зададат ограничения и т.н., на UML елемент .



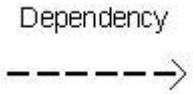
(2) **Взаимоотношения или връзки (Relationship):**

Relationship (Взаимоотношение или Връзка) е друг най-важен градивен елемент на UML. Показва как градивните елементи (инструменти) са свързани един с друг и тази асоциация описва функционалността на приложението.

Има четири вида взаимоотношения на разположение в UML:

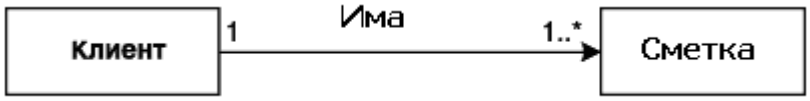
Зависимост (Dependency):

Зависимост (Dependency) е връзка между два инструмента, при която промяната на един от инструментите може да повлияе на друг.



Асоциация (Association):

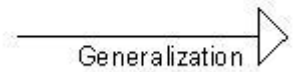
Асоциацията всъщност е множество от връзки, което свързва елементите на един UML модел. Тя описва колко обекта участват в това взаимоотношение.



Направлението на стрелката в еднопосочна асоциация показва кой инструмент (в случая клас) трябва да се указва първи при разглеждане на връзката и кой - втори, в случая имаме бинарната асоциация с име „Има". Клас „Клиент" трябва да се показва първи, а клас „сметка" – втори, при следния смисъл: „клиент" Има „сметка1", „сметка2" ....

Генерализация или Обобщение (Generalization):

Обобщението (Generalization) може да се дефинира като връзка, която свързва специализиран елемент с обобщен елемент. Тя всъщност описва „взаимоотношение на наследяване" в света на ООП.



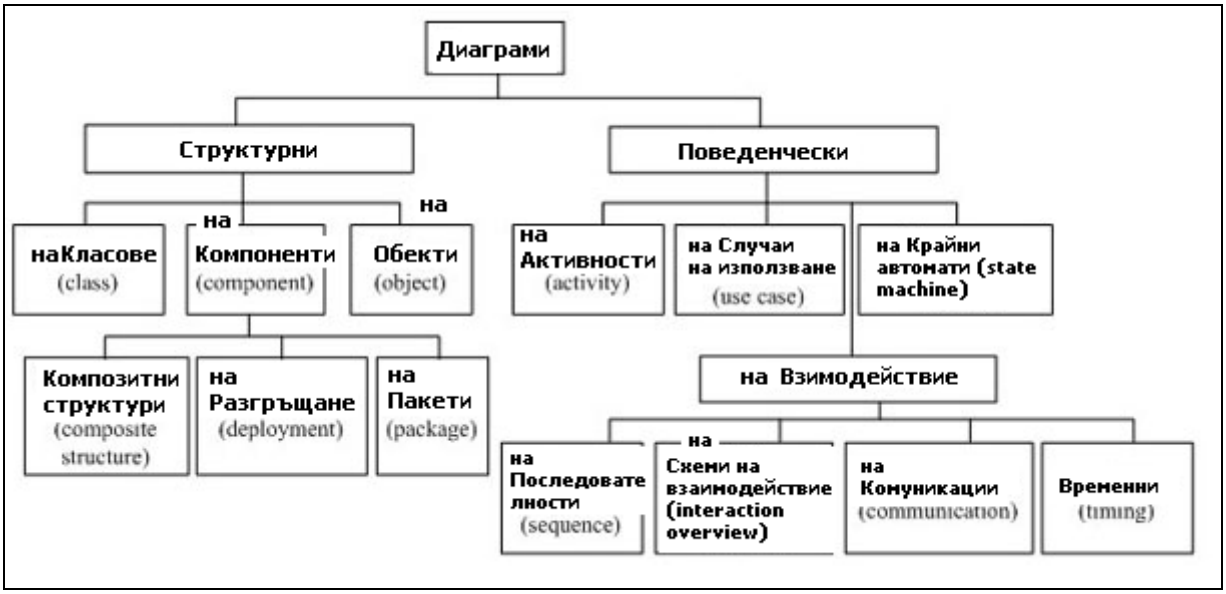
Реализация (Realization):

Реализация (Realization) може да бъде дефинирана като връзка при която 2 елемента са свързани. Единият от елементите описва някакви отговорности, които не са имплементирани, а другият - имплементира тези отговорности. Това е връзка между интерфейси.

(3) UML Diagrams:

UML диаграмите са най-важната част от на нашата работа по време на този курс. Нито една диаграма самостоятелно не моделира системата в нейната пълнота. Всички изброени по-горе изграждащи елементи се използват за създаване на колекция от UML диаграми имащи за цел цялостното моделиране на системата. Повечето UML диаграми **(**без „sequence diagrams" и „timing diagrams") се изобразяват като графи, при съответна интерпретация на върховете и дъгите на графа.

UML включва следните 13 диаграми, повечето от които ще разглеждаме както по време на лекции, така и по време на лабораторни упражнения:



Предназначение:

- За спецификация на изискванията и архитектурата на системата
- За комуникация между участниците в проекта
- За документиране на системата
- За симулация на модела (с автоматично генерирания код)
- За тестване и верификация на модела

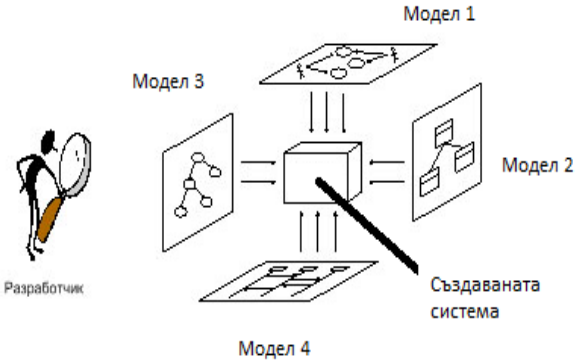
Понятието „**спецификация**" има двойствено значение:

- **„описание"** като **документ** (т.е. спецификация на изискванията към софтуера или на проекта),
- **самия процес на описване** в документа на изискванията (Software Requirements Specification) или в спецификацията за проекта.

Най-общо, спецификацията на изискванията" описва **какво системата трябва да прави**, а тази на "проекта" ("спецификация за реализация") - как ще го прави.

1.3 UML перспективи (гледни точки)

Всяка реално съществуваща система се използва от различни потребители. Потребителите могат да бъдат разработчици, тестери, бизнесмени, анализатори и много други. На всички тях е необходим модел на системата, но **от различна гледна точка** (или перспектива). Моделите **се множат преди всичко заради различни видове дейности в процеса на разработка на ПО.** UML дава възможност за моделиране на системата от различна гледна точка или перспектива. Ще спомена основните:



- **Проектна (Design)** – състои се от класове (classes), интефейси (interfaces) и сътрудничество (collaboration). UML осигурява class diagram и object diagram за поддръжката на тази перспектива.
- **На разработка (Implementation)** - дефинира компонентите, изграждащи пълна физическа система. UML component диаграма се използва за изпълнението перспектива.
- **На процес (Process)** – дефинират се работните потоци в системата. Така че, същите елементи, като при Design се използват за подкрепа на тази гледна точка.
- **На разгръщане или разполагане (Deployment)** - представя физическите възли на системата, които формират хардуера. UML диаграма на разгръщане се използва за подкрепа на тази гледна точка.

И в центъра е **„Use Case view"** (Изглед „Случай на използване"), който свързва всички тези 4 перспективи. Един случай на използване (use case - случай на употреба, вариант на използване и т.н)

представя функционалността на системата. Така, другите перспективи са всъщност свързани със случая на използване.

1.4. UML инструментални средства и StarUML.

UML е език моделиране, който се поддържа от множество инструментални средства. Едно такова „open source" средство е StarUML (http://sourceforge.net/projects/staruml/files/staruml/5.0/), твърде подобно на утвърдени комерсиални UML tools (н.р, IBM Rational Rose, iLogix Rhapsody, SPARX Enterprise Architect). Има и други „open source" инструментални средства, които вие може да използвате вместо StarUML, например ArgoUML, Dia, Visual Paradigm Community Edition, Violet и др. Тази секция съдържа кратки обяснения на StartUML, в случай че сте го избрали. Повече подробности може да намерите на **http://staruml.sourceforge.net/docs/user-guide(en)/toc.html.** За да се свали **продукта**: http://sourceforge.net/projects/staruml/files/staruml/5.0/

Базови понятия при използване на StarUML

- Model, View и Diagram (Модел, Изглед и Диаграма)
- Project и Unit (Проект и Проектна единица)

StarUML прави ясно разграничение между **модели (models)**, **изгледи (views)** и **диаграми (diagrams)**.

Виж. „Програмни спецификации . Ръководство за лабораторни упражнения" тема II.

Използвана литература:

http://www.tutorialspoint.com/uml/uml_overview.htm
http://staruml.sourceforge.net/docs/user-guide(en)/toc.html