

4. Модели на разпределена софтуерна архитектура

Васил Георгиев



ci.fmi.uni-sofia.bg/



v.georgiev@fmi.uni-sofia.bg

Съдържание

- Модели софтуерна архитектура
- Спецификации с **UML**
- Структурни и функционални диаграми
- Модели на изгледи



Модели софтуерна архитектура

- Софтуерната архитектура представя – т.е. моделира – програмния проект (процес на обслужване) като съставен т.е. разпределен процес от софтуерни компоненти
- моделирането на РСА е първата и най-важна фаза на проектиране, настройка, тестване, разгръщане и документация на разпределени среди за обслужване
- моделът на дадена софтуерна архитектура описва
 - декомпозицията на процеса на компоненти
 - функционалната им композиция
 - прилагания архитектурен стил – напр. процедурен, обектен, потоков (**data flow**), йерархичен или не-йерархичен, информационен (**data centric**), интерактивен (**interaction oriented**), базиран на изгледи (**views**) и др.
 - качественте (нефункционалните) атрибути на услугата – **QoS**

Фаза Анализ, предхождаща моделирането

- колаборацията,
- акцията и
- конкурентността между обектите
- UML диаграмите могат да се транслират до HLL с общо приложение

SPMD
MPMD (MPSD)

анализ → моделиране → проектиране . . .

- 1) дефиниране на образци (use cases: еволюция, развитие, гатане)
- 2) дефиниране на образци - SFS
- 3) Набор от параметри на образците за R-образци
 - функционални - реализирани алгоритми
 - нефункционални - security, QoS

изглед, (http: функционални зависимости)

образец	сервиз 1	сервиз 2	функционални зависимости
[1]	✓	не	ни
[2]			и
...			
процес			...

Представяне на софтуерните модели

- Използват се графи и техни разширения
- описанието е чрез диаграми или техни текстови еквиваленти
- цели на описанието са
 - визуализация
 - спецификация
 - конструиране
 - документация
- ☞ следователно обикновено моделът включва мн. повече от една диаграма
- описанието (моделирането) стартира от по-упростените концепции на бизнес-модела или потребителския сценарий
 - напр. едномерен модел с блокова диаграма (ненасочен граф) – 4.5
- за по пълно функционално и нефункционално описание на проекта се прилагат многомерни модели
 - напр. «4+1» модели, включващи
 - логически изглед
 - изглед процеси
 - изглед проектиране
 - физически изглед
 - потребителски интерфейсни изгледи

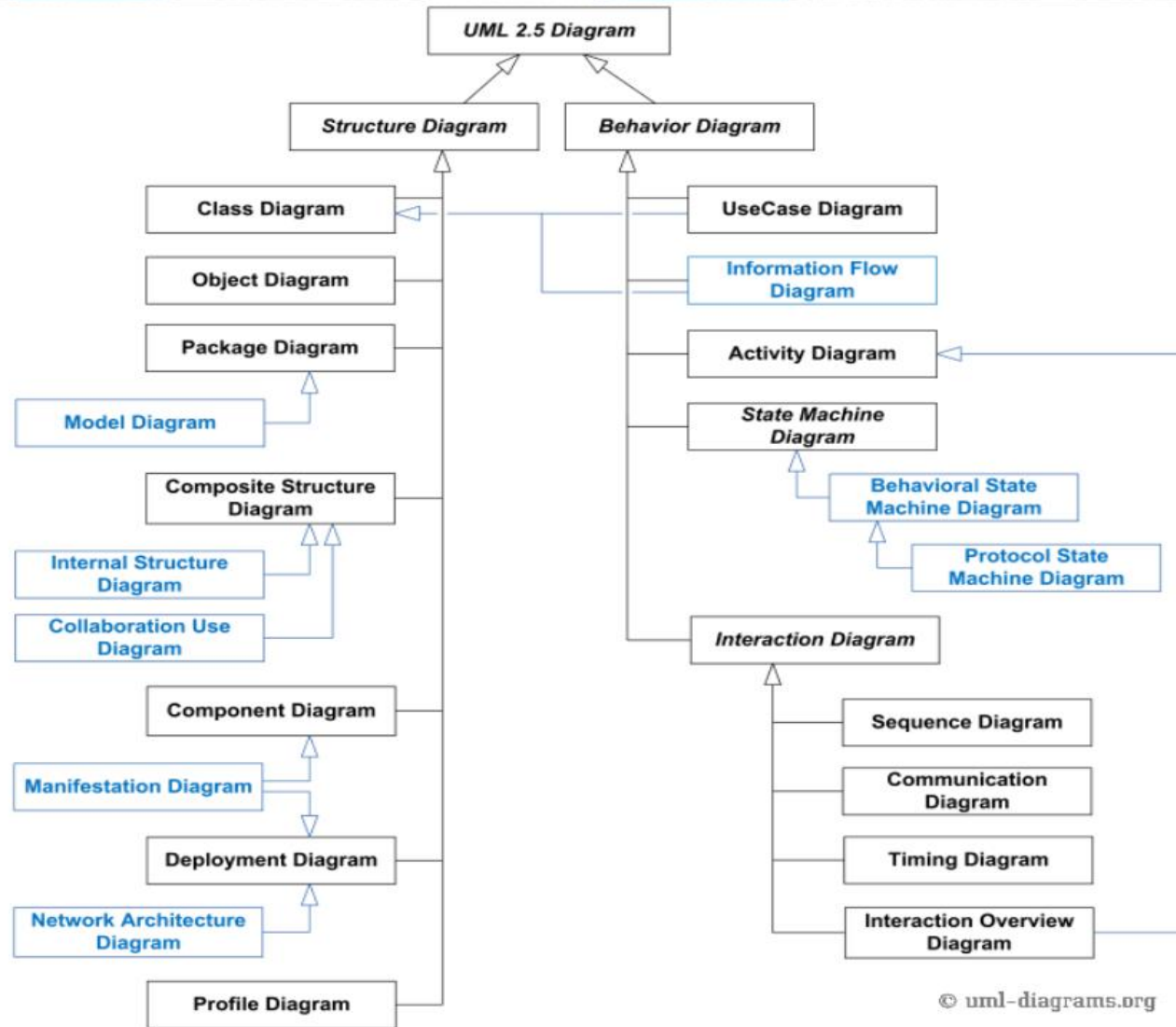
Блокова диаграма на система за електронна търговия



UML-модели на СА

- ✦ използва се за ОО-спецификация, анализ, проектиране и документиране на софтуерни проекти
- ✦ спецификациите са в две групи диаграми:
- ✦ структурни диаграми – **статично** описание (изреждане) на елементите в системата
 - ✦ йерархична библиотека класове
 - ✦ статични връзки между класовете
 - ✦ наследяване (“is a”)
 - ✦ асоциация (“uses a”)
 - ✦ агрегация (“has a”)
 - ✦ обмен (method invocation)
- ✦ функционални (**behavioral**) диаграми – **динамично** описание функциите (“поведението”) на инстанциите на класовете (т.е. обектите) с диаграми на
 - ✦ интеракцията,
 - ✦ колаборацията,
 - ✦ акцията и
 - ✦ конкурентността между обектите
- ✦ UML диаграмите могат да се транслират до HLL с общо приложение

UML-диаграми за СА – фиг. 4.8



© uml-diagrams.org

Структурни UML диаграми

Class	Изброяване и статични връзки между класовете (независещи от взаимодействието им по вр. на изпълнение)
Object	Извлечение от клас диаграмата за обектите и тяхното взаимодействие в определени специфични моменти от изпълнението на системата
Composite	Диаграма на съставни структури – описание на структурата на даден компонент като съставлящи го класове и компонентните интерфейси
Component	Описание на системата като структура от компоненти, интерфейсите между тях, и общите системни интерфейси
Package	Йерархична пакетна структура на организацията класовете в директории (т.е. групирани файлове) – пакети от класове и пакети от пакети
Deployment	Диаграма на разгръщането - описание на изпълнителната инфраструктура: сървери, изпълняващи компонентите , системно осигуряване и мидълуер, интерфейси и протоколи, вътрешна и външна мрежова свързаност

Функционални UML диаграми...

Use case	Диаграма на случай на употреба – потребителските сценарии на заявки към системата и техните реакции – за описание на функционалните и нефункционалните изисквания към системата
Activity	Диаграма на дейностите – описание на контролния и контекстния обмен между класовете като мрежа от акции, които системата изпълнява за да осъществи реакциите по потребителския сценарий – оркестрация на акциите
State Machine	Диаграма на машина на състоянията – описание на жизнения цикъл на обектите като машина на състоянията – диаграми на състоянията и преходите (активни вътрешно-обусловени и реактивни външнообусловени преходи)

... функционални UML диаграми

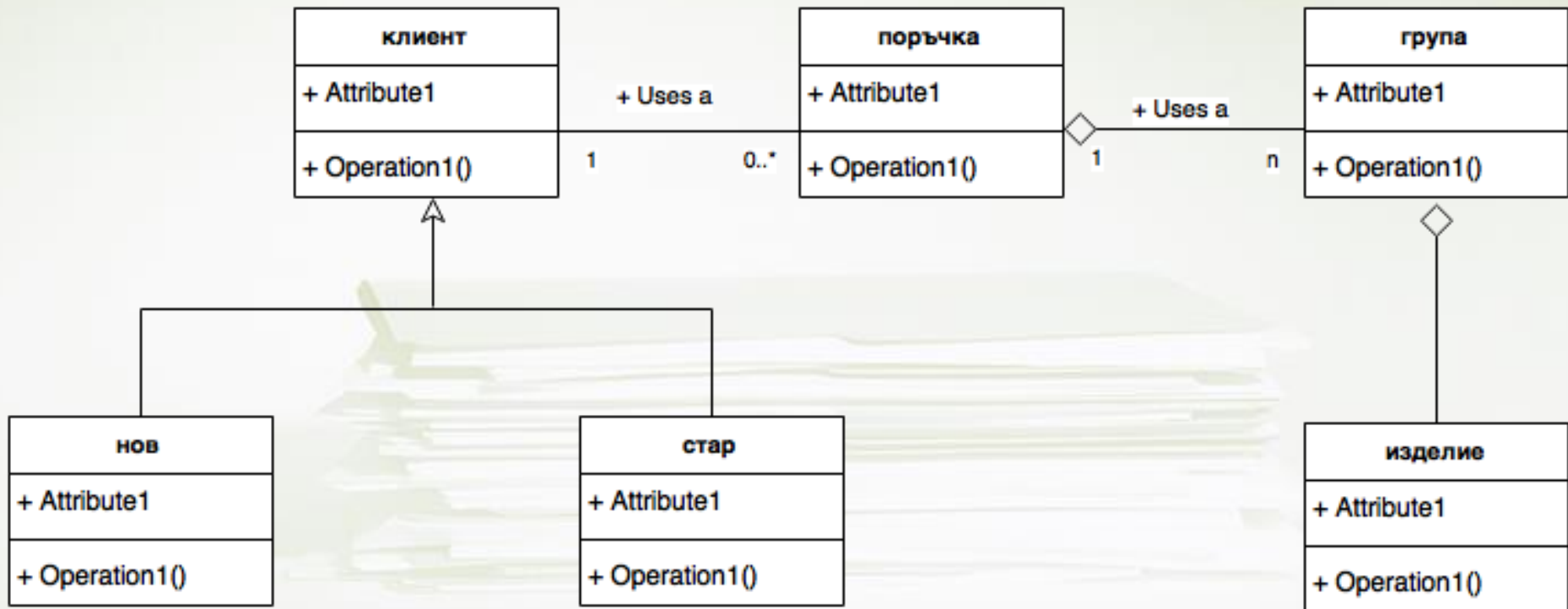
Inter-action Overview	Диаграма за преглед на взаимодействието – описва потока команди между обектите (control flow) и е комбинация от Action и Sequence диаграмите
Sequence	Диаграма на последователност - нареден (т.е. времеви) списък от съобщенията между обектите
Communication	Аналогично на Sequence диаграмата, но структурирана като като комуникационни канали , които съдържат определен брой последователности
Time Sequence	Времево описание на преходите между вътрешните състояния на обектите и на различимите външни събития (от потребителския сценарий) като последователност от съобщения

Клас диаграми

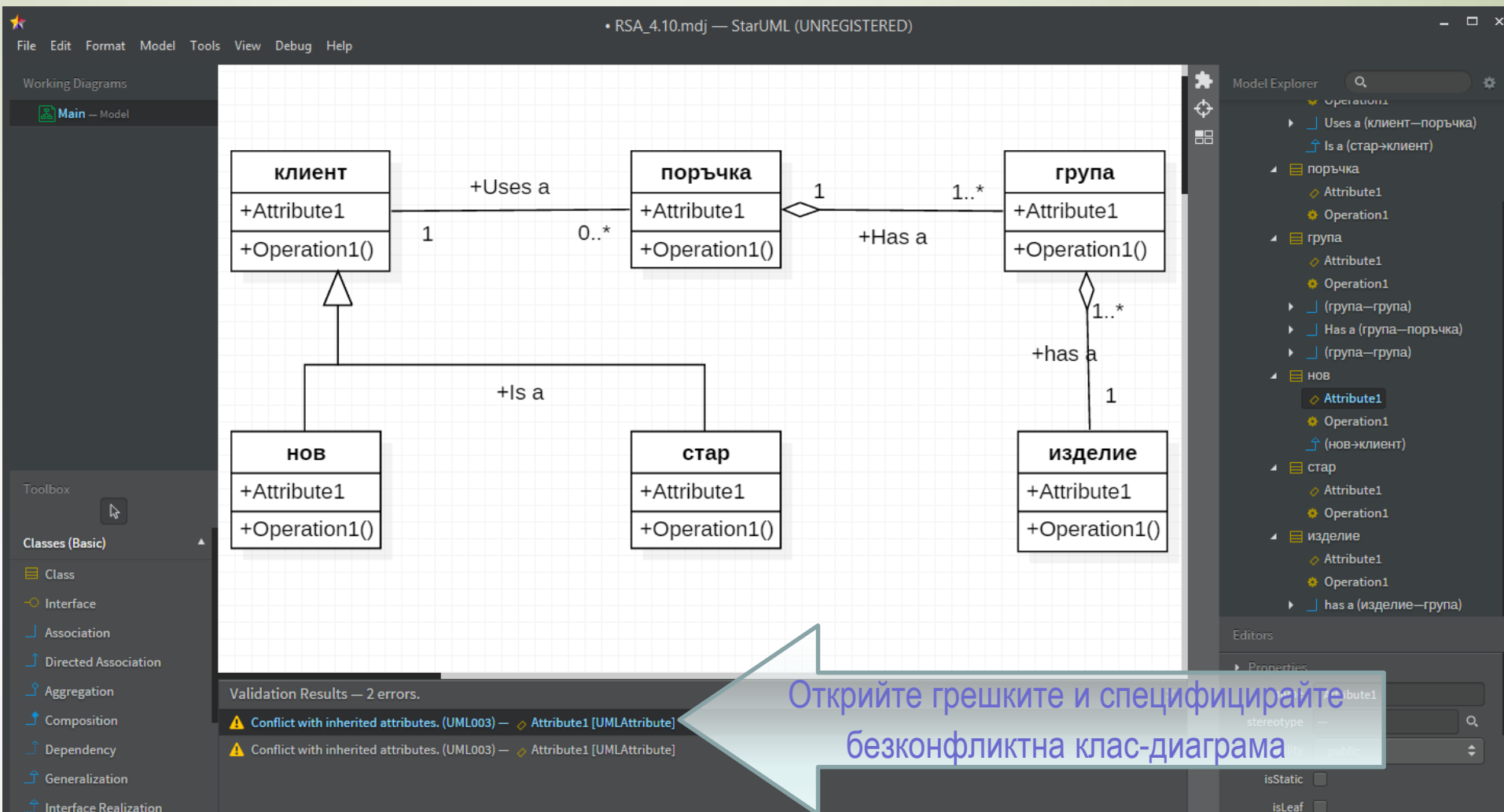
- най-разпространеното описание при всеки модел
- статично изброяване на съставните блокове на модела като **класове**
- задава «**речника**» на модела в съответствие с проблемната област
- класовете се описват с техните атрибути
 - тип
 - интерфейс
 - методи
 - свойства
- достъпността (видимостта) на атрибутите се описва като
 - **public**
 - **private**
 - **protected**
 - **default**
- описва се и отношенията между класовете – наследяване, асоциация, агрегация (чрез дъги)
 - а също и мощността на тези отношения – 1:1, 1:много и т.н. (чрез маркировки в края на дъгите)

Клас диаграма – фиг. 4.12

- система за потребителски заявки
- наследственост “is-a” (стрелка към родителя/базовия клас)
- агрегация “has-a” (ромб към корена)
- асоциация “uses-a” (нейерархична дъга)
- маркировка на мощността в двата края на дъгите

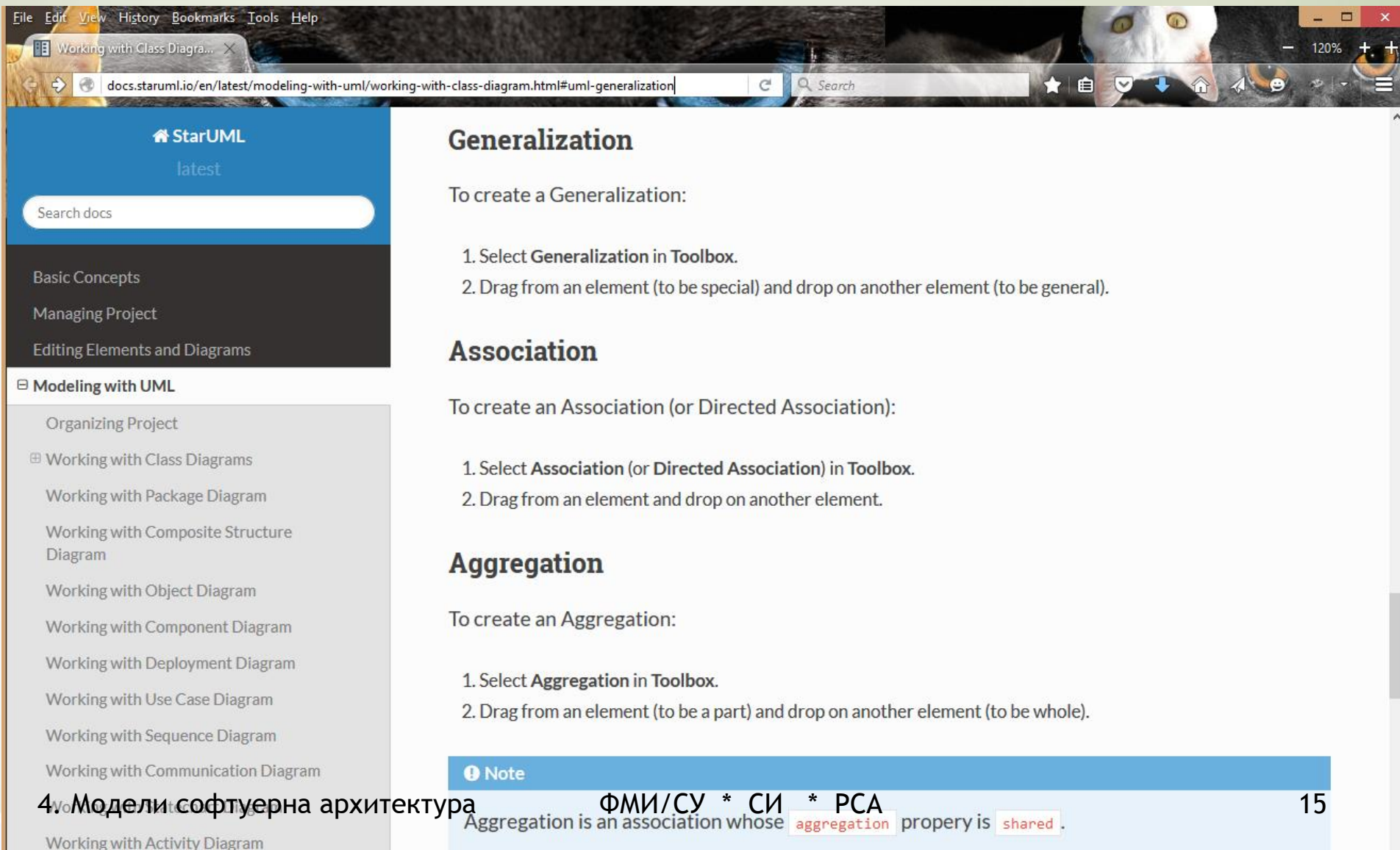


Моделиране със StarUML



РЪКОВОДСТВО по StarUML

- <http://docs.staruml.io/en/latest/modeling-with-uml/working-with-class-diagram.html#uml-generalization>



The screenshot shows a web browser window displaying the StarUML documentation. The browser's address bar shows the URL: `docs.staruml.io/en/latest/modeling-with-uml/working-with-class-diagram.html#uml-generalization`. The page content is as follows:

Generalization

To create a Generalization:

1. Select **Generalization** in **Toolbox**.
2. Drag from an element (to be special) and drop on another element (to be general).

Association

To create an Association (or Directed Association):

1. Select **Association** (or **Directed Association**) in **Toolbox**.
2. Drag from an element and drop on another element.

Aggregation

To create an Aggregation:

1. Select **Aggregation** in **Toolbox**.
2. Drag from an element (to be a part) and drop on another element (to be whole).

Note
Aggregation is an association whose `aggregation` property is `shared`.

4. Модели софтуерна архитектура ФМИ/СУ * СИ * PCA 15

Моделиране с draw.io

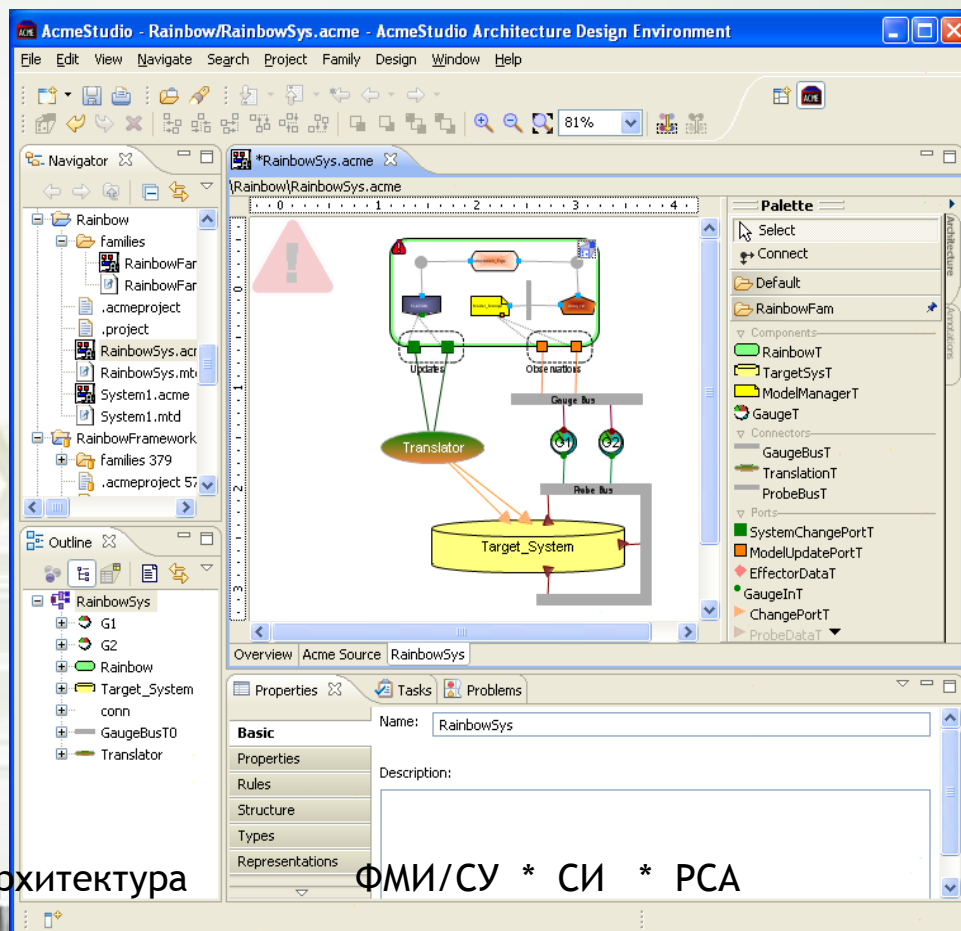
The screenshot displays the draw.io web interface for editing a UML class diagram. The diagram features the following elements:

- Classes:**
 - клиент**: + Attribute1, + Operation1()
 - поръчка**: + Attribute1, + Operation1()
 - група**: + Attribute1, + Operation1()
 - изделие**: + Attribute1, + Operation1()
 - стар**: + Attribute1, + Operation1()
- Relationships:**
 - Generalization:** 'стар' inherits from 'клиент'.
 - Association:** 'клиент' (multiplicity 1) uses 'поръчка' (multiplicity 0..*).
 - Association:** 'поръчка' (multiplicity 1) uses 'група' (multiplicity 1).
 - Association:** 'група' (multiplicity 1) uses 'изделие' (multiplicity n).

The interface includes a toolbar with various drawing tools, a left sidebar with UML shape libraries, and a right sidebar with view and paper size settings. The browser address bar shows the URL: <https://www.draw.io/#G0B76lPEAsVyyMMV90ZVppSWZnbkE>.

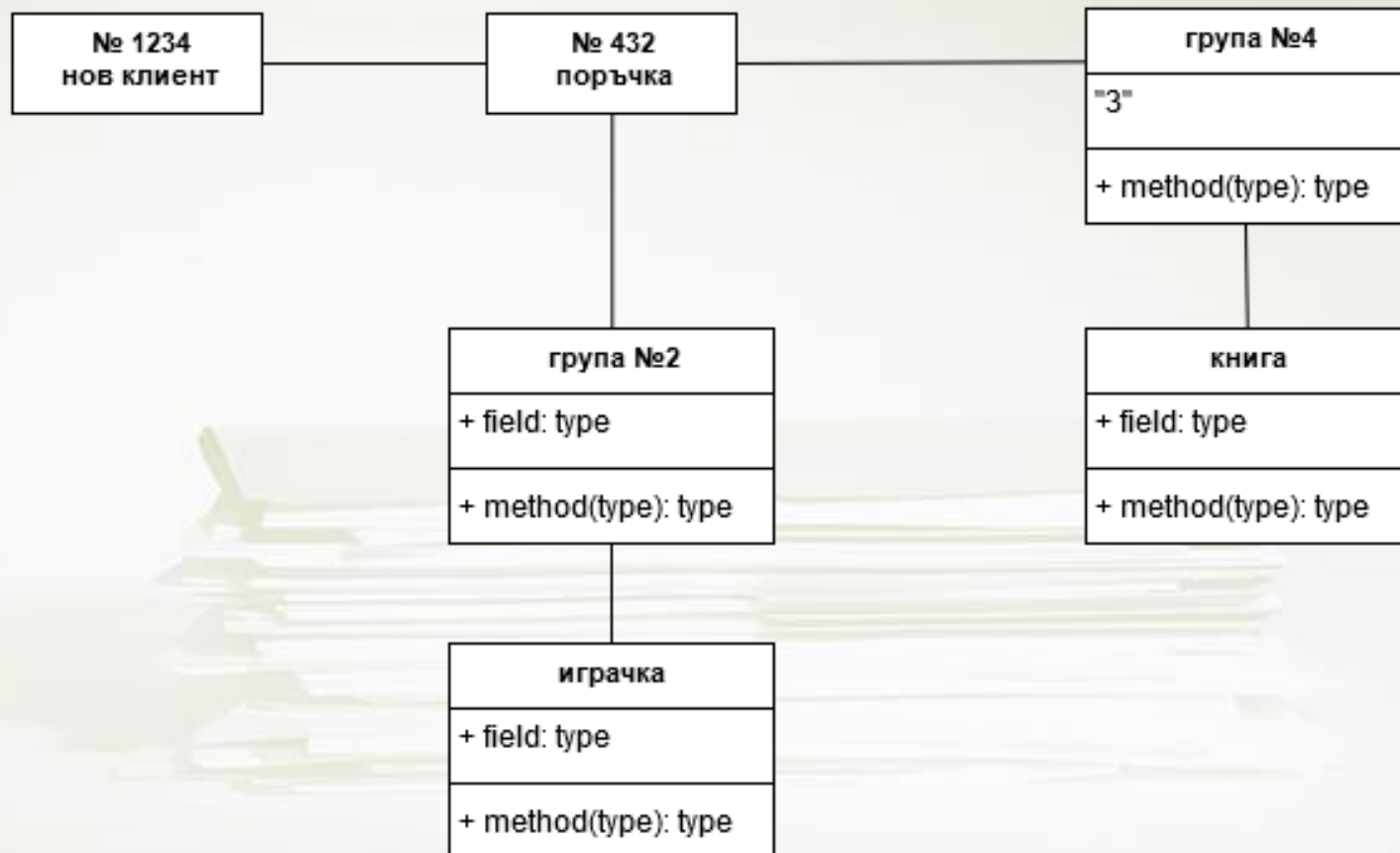
Моделиране с ADL

- Architectural Description Language - графична спецификация на модели на разпределена софтуерна архитектура
- свободна среда за спецификация на ADL- модели AcmeStudio (<http://www.cs.cmu.edu/~./acme/AcmeStudio/index.html>) с автоматична генерация на Java и C++



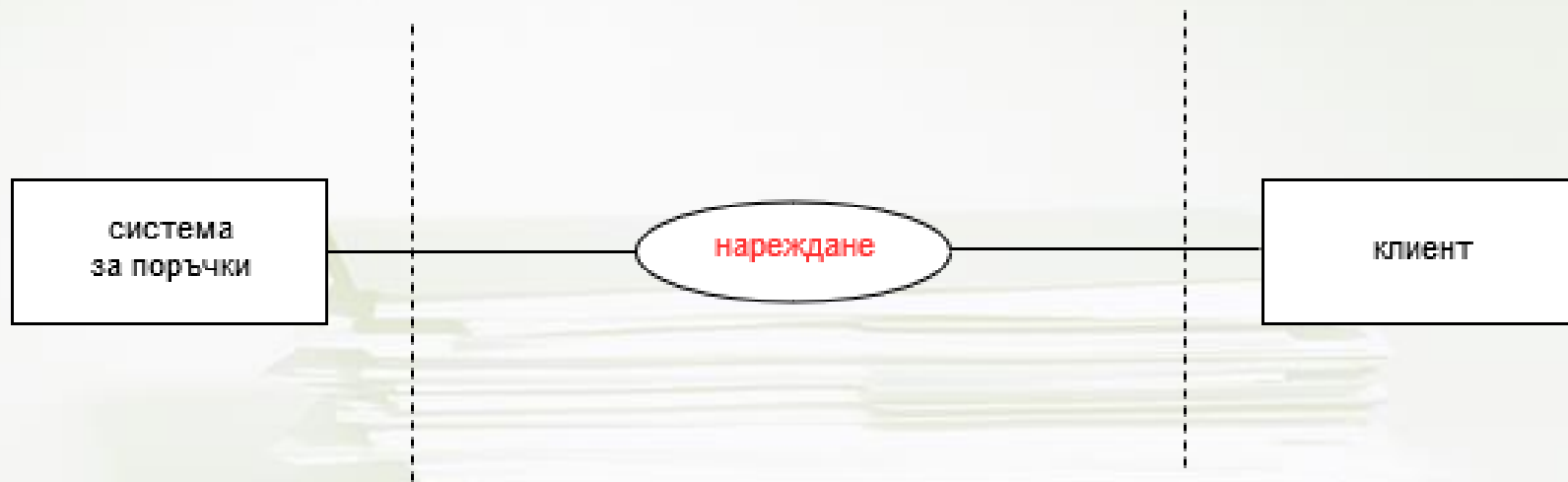
Обектни диаграми – фиг. 4.18

- ✦ извличат се от клас-диаграмата
- ✦ описва обектите като инстанции на класовете т.е. примерно подмножество обекти за дадена клас-диаграма конкретен момент на работа на системата



Диаграма на съставната структура– фиг. 4.19

- описва връзката между обектите (**runtime**), с което разширява “речника” на модела
- обектите и връзката се означават с **етикети** – съответно на ролята (бизнес- или функционална логика) и отношението им (“колаборацията”)



Компонентни диаграми – фиг. 4.20

- компонентите са изпълними **SW**-модули за многократно използване при проектиране, които се представят със своя интерфейс
- в **UML** те са със скрита структура (черна кутия) [но при различните технологии се прилагат и компоненти тип “сива” и “стъклена кутия”]
 - **jar** в компонентната библиотека **JavaBean**
 - **dll** в **.Net**
- компонентната диаграма представя съответствието между изискваните (полукръгче) и имплементираните (кръгче) интерфейси
- компонентите в даден проект може да са готови – **COTS** – и специфични

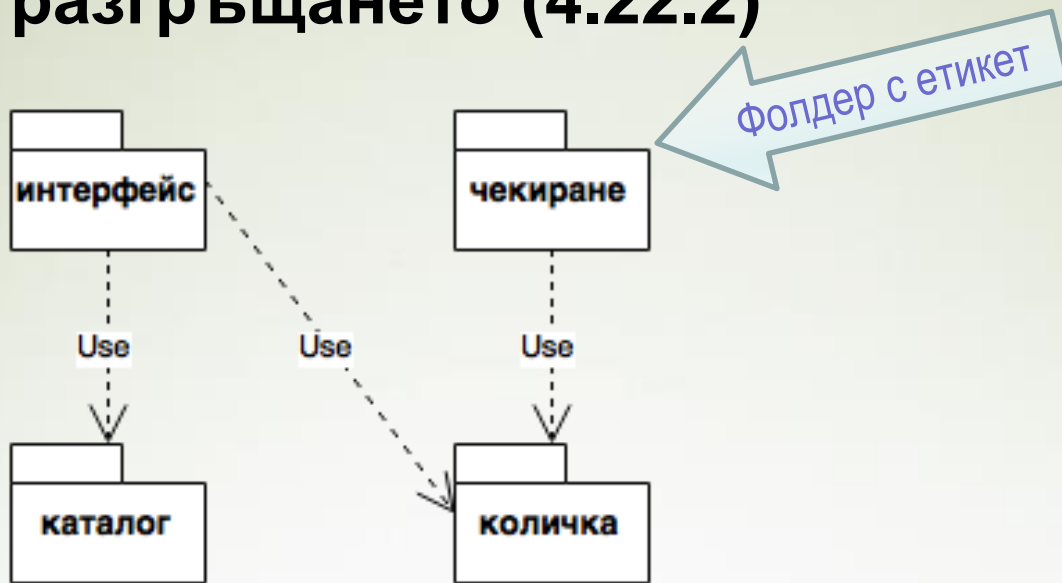


... Компонентни диаграми – фиг. 4.21

- компонентната диаграма представя съответствието между изискваните (полукръгче) и имплементираните (кръгче) интерфейси
- компонентите в даден проект може да са готови – COTS – и специфични



Пакетна диаграма (фиг. 4.22.1) и диаграма на разгръщането (4.22.2)



Диаграма на случаите на употреба

- ✦ описва потребителските сценарии на приложение на системата като граф от актори, случаи на употреба (потребителски функции) и връзките между тях
- ✦ акторите са крайни потребители или други системи, приложения и устройства
- ✦ случаите (**Use Cases**) са комплексни функционални модули от разпределеното приложение/проекта, които описват отделни стъпки от цялостната бизнес-логика
- ✦ описанието на случаите се допълва в други диаграми с пред- и след-условията на изпълнението им като последователности от стъпките на общото приложение при конкретно негово изпълнение
- ✦ връзките между сценариите се маркират с
 - ✦ <<**include**>> от случай, който използва друг случай за изпълнение на дадена функция (насочена дъга)
 - ✦ <<**extend**>> от случай, който извиква друг такъв за изпълнение на функция по изключение (т.е. като опция, която се изпълнява само по изключение)
- ✦ диаграмите на случаите на употреба са основа на описанието и [началните] им версии се използват за основа на структурните и **sequence** диаграмите

Диаграма на случаите на употреба (фиг. 4.24)

- описва потребителските сценарии на приложение на системата като граф от случаи на употреба (потребителски функции) и връзките между тях
- акторите са крайни потребители на системи, приложения или устройства
- случаите (Use Cases) са функционални модули на приложение/проекта, които описват отделни стъпки от цялостната бизнес-логика
- описанието на случаите може да бъде представено с други диаграми с помощта на условията за изпълнение им като последователности от стъпките на приложение при конкретно негово изпълнение
- връзките между сценарии са маркират с:
 - «include» от случай, който е необходим за изпълнение (насочена дъга)
 - «extend» от случай, който извиква друг такъв за изпълнение (т.е. като опция, която се изпълнява само по изключение)
- диаграмите на случаите на употреба са основа на описанието и [началните] им версии се използват за основа на структурните и sequence-диаграмите



Диаграма на дейностите

- ✦ описва проекта като **поток** (*workflow*) бизнес процес, състоящ се от дейности – **activities**
- ✦ дейностите капсулират
 - ✦ логиката на взимането на решение
 - ✦ конкурентното изпълнение на функции
 - ✦ обработката на изключения
 - ✦ прекратяването на процеса (**termination**)
- ✦ потоковата **activity** диаграма се състои от
 - ✦ една начална точка и поне една крайна точка (плътен кръг и ограден кръг)
 - ✦ точките на решаване (означават се с ромбче)
 - ✦ другите дейности (заоблен правоъгълник)
 - ✦ конкурентното разделяне и събиране на потоците (дебела черта); **N.B.** – събирането на два и повече потока се счита за синхронизатор (следващите го дейности не могат да стартират без завършване на *всички* предхождащи го)
 - ✦ събития (**events** - опция) – представят обмена на съобщения (**signals**) между конкурентните акции (насочени многоъгълници с етикети)

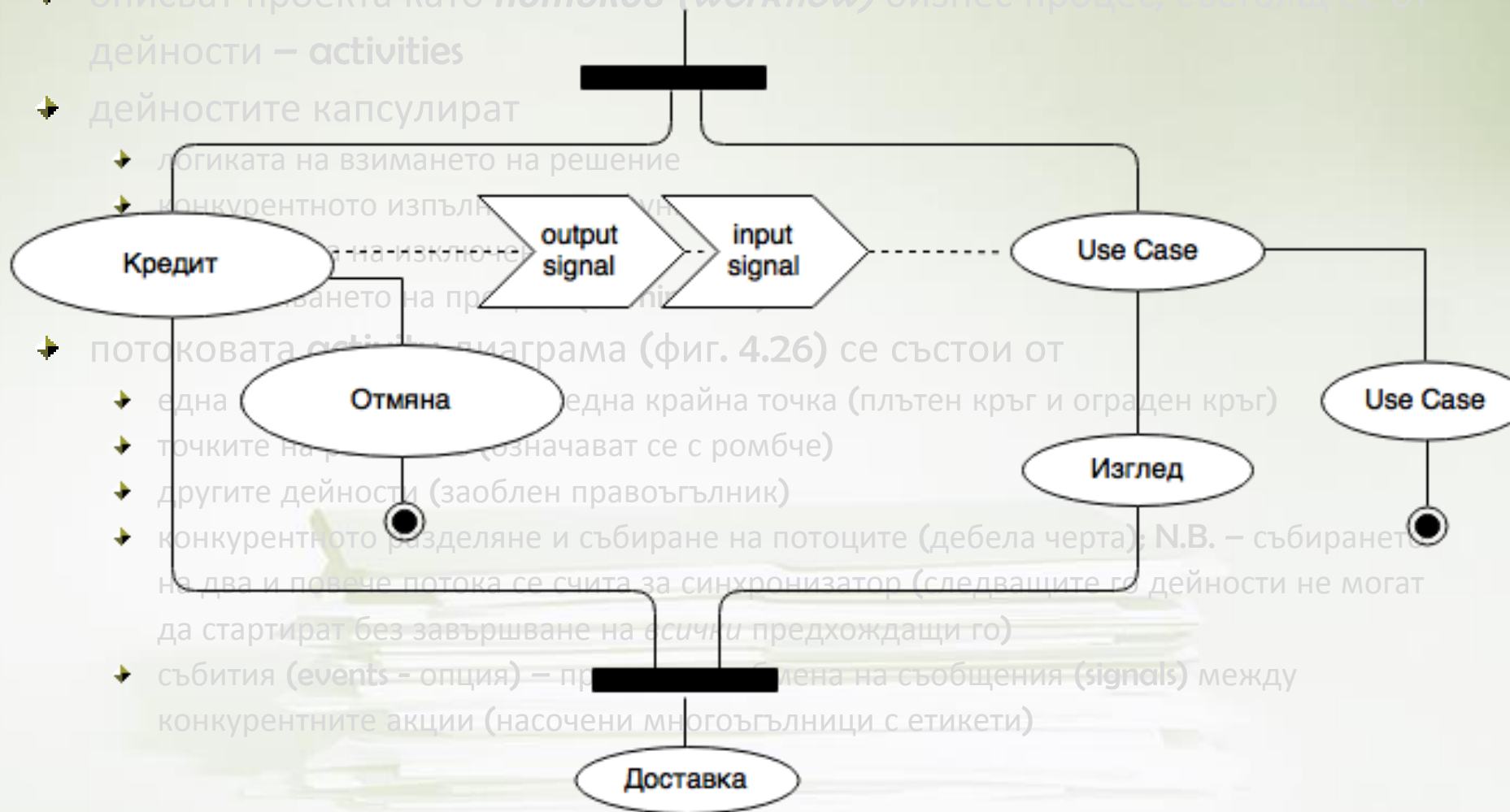
... Диаграма на дейностите (фиг. 4.26)

➤ описват проекта като *потоков (workflow) бизнес процес*, състоящ се от дейности – **activities**

➤ дейностите капсулират

➤ логиката на взимането на решение

➤ конкурентното изпълнение



➤ потоковата *activity* диаграма (фиг. 4.26) се състои от

➤ една крайна точка (плътен кръг и ограден кръг)

➤ точките на разклоняване (означават се с ромбче)

➤ другите дейности (заоблен правоъгълник)

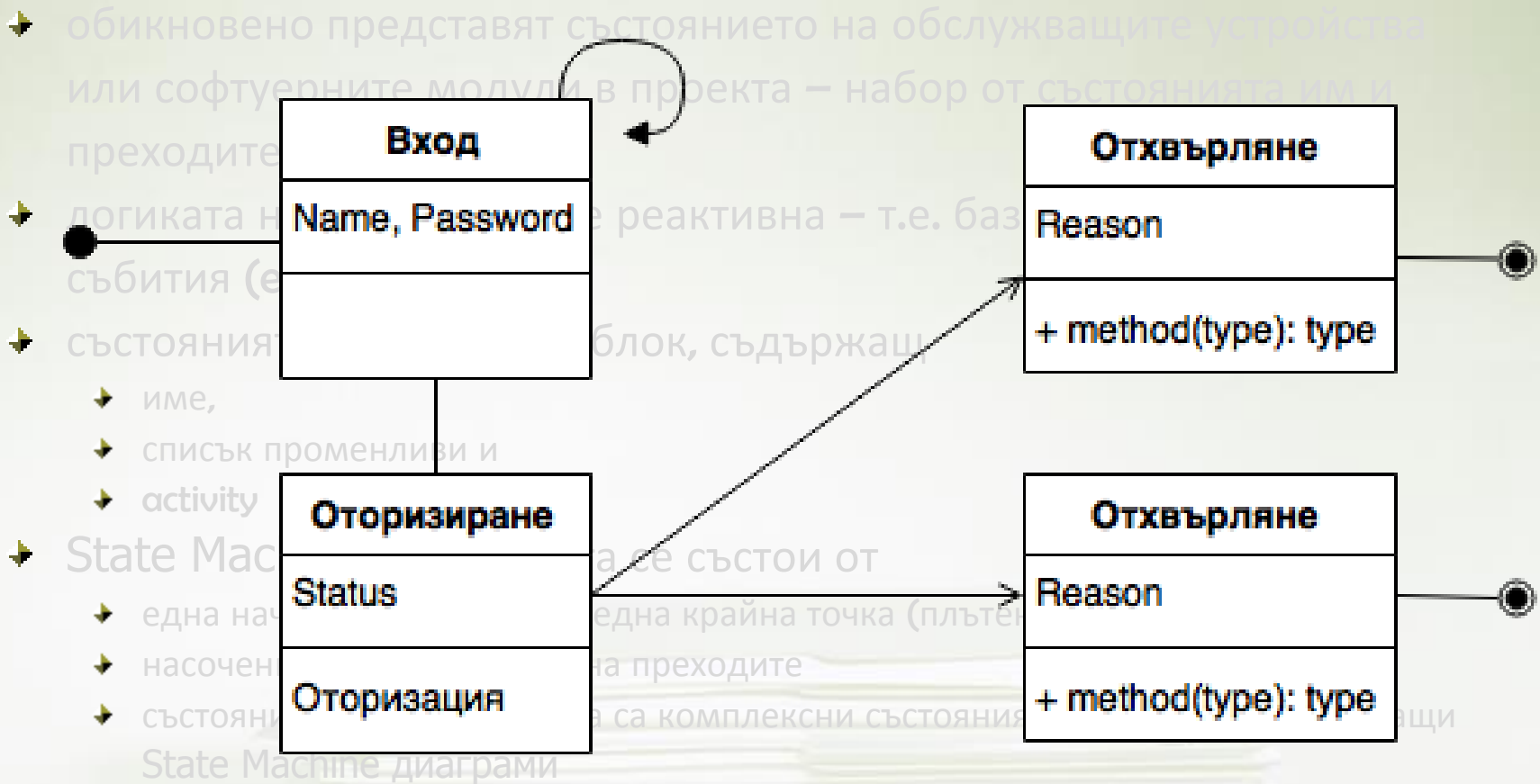
➤ конкурентното разделяне и събиране на потоците (дебела черта); **N.B.** – събирането на два и повече потока се счита за синхронизатор (следващите го дейности не могат да стартират без завършване на всички предхождащи го)

➤ събития (**events** - опция) – пренасочена на съобщения (**signals**) между конкурентните акции (насочени многоъгълници с етикети)

Диаграма Машина на състоянието

- обикновено представят състоянието на обслужващите устройства или софтуерните модули в проекта – набор от състоянията им и преходите между тях
- логиката на състоянията е реактивна – т.е. базира се на външни събития (**events**)
- състоянията се описват с блок, съдържащ
 - име,
 - списък променливи и
 - **activity**
- **State Machine** диаграмата се състои от
 - една начална точка и поне една крайна точка (плътен кръг и ограден кръг)
 - насочени маркирани дъги на преходите
 - състоянията, които може да са комплексни състояния, съставени от допълващи **State Machine** диаграми

... Диаграма на Машина на състоянието (фиг. 4.28)

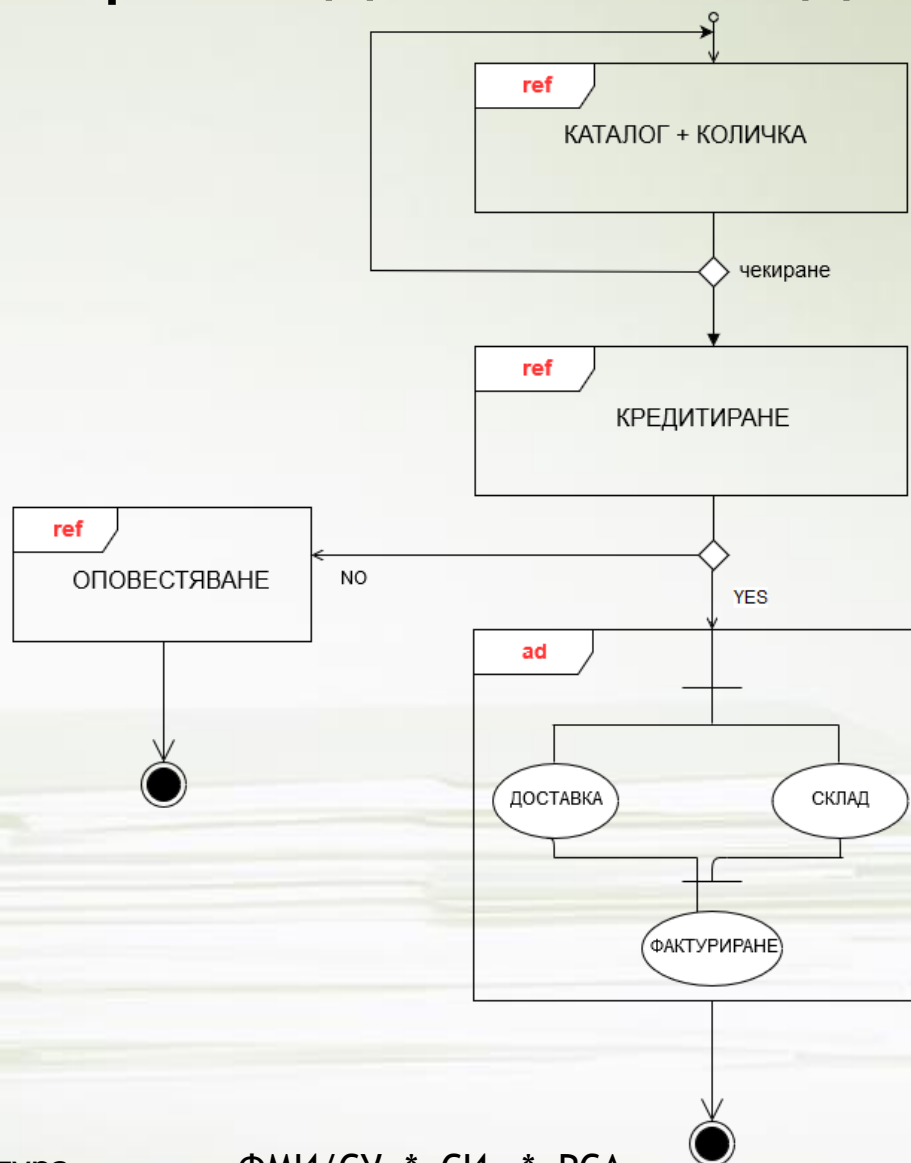


- обикновено представят състоянието на обслужващите устройства или софтуерните модули в проекта – набор от състоянията им и преходите
- догиката на реактивна – т.е. базирани на събития (event-driven)
- състоянията са блок, съдържащ
 - име,
 - списък променливи и
 - activity
- State Machine се състои от
 - една начална точка (плътен кръг)
 - насочен преходите
 - състоянията са комплексни състояния

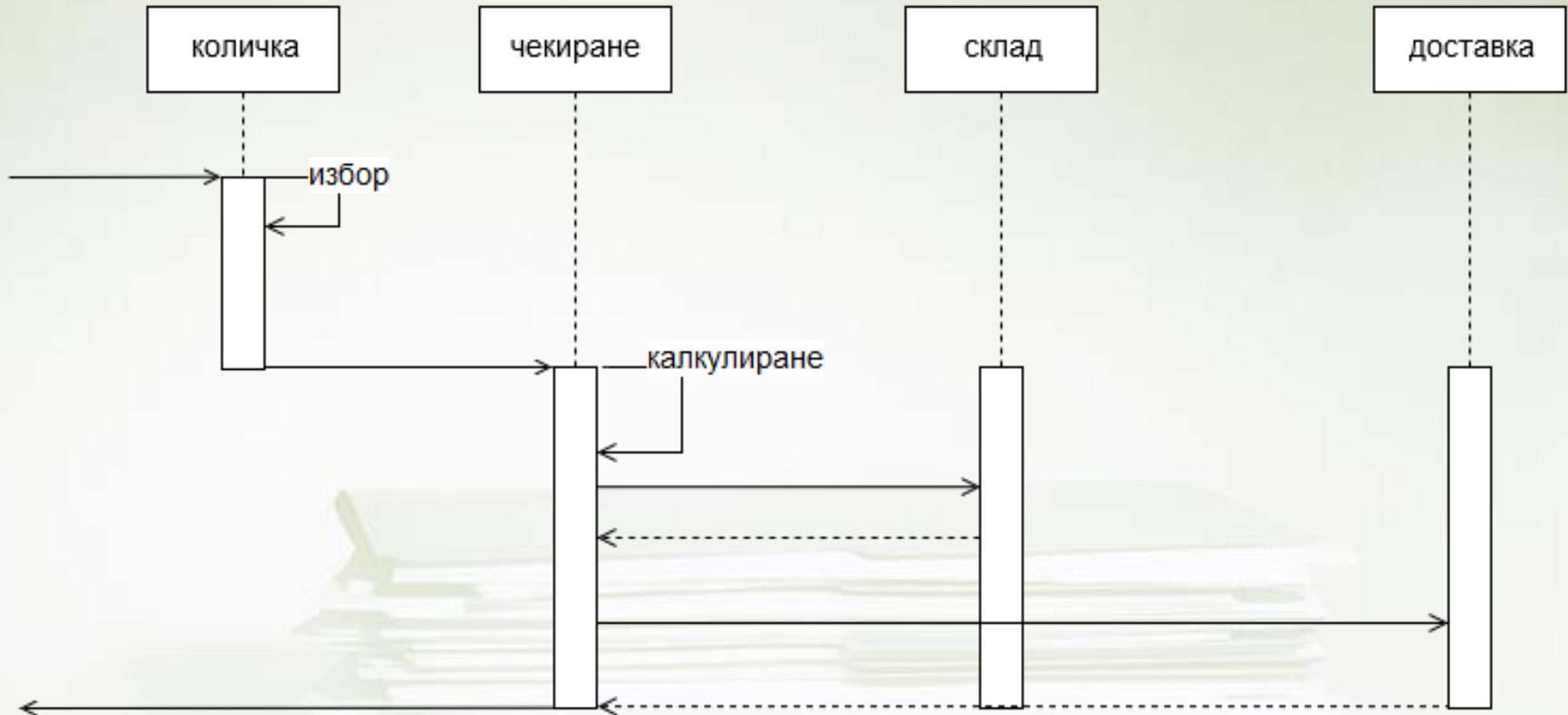
Диаграми за преглед на взаимодействието, последователности и времеви диаграми

- диаграмите за преглед на взаимодействието (Interaction Overview) се състоят от кадри (frames), които представляват други диаграми на проекта, маркирани с указател (reference) или със самите диаграми, маркирани с тип – напр. **sd**, **cd**, **ad**
 - дъгите отразяват контролния поток на взаимодействието
- последователностните (sequence) диаграмите отразяват относителната последователност от контролни съобщения между обектите
- времевата диаграма описва графика на състоянията от машината на състоянията - прилага се за RTприложения и системи – RTOS, ES

Диаграма за преглед на взаимодействието – фиг. 4.30.



Последователностна диаграма – фиг. 4.31.



Фиг. 4.18.2

Модел на изгледи

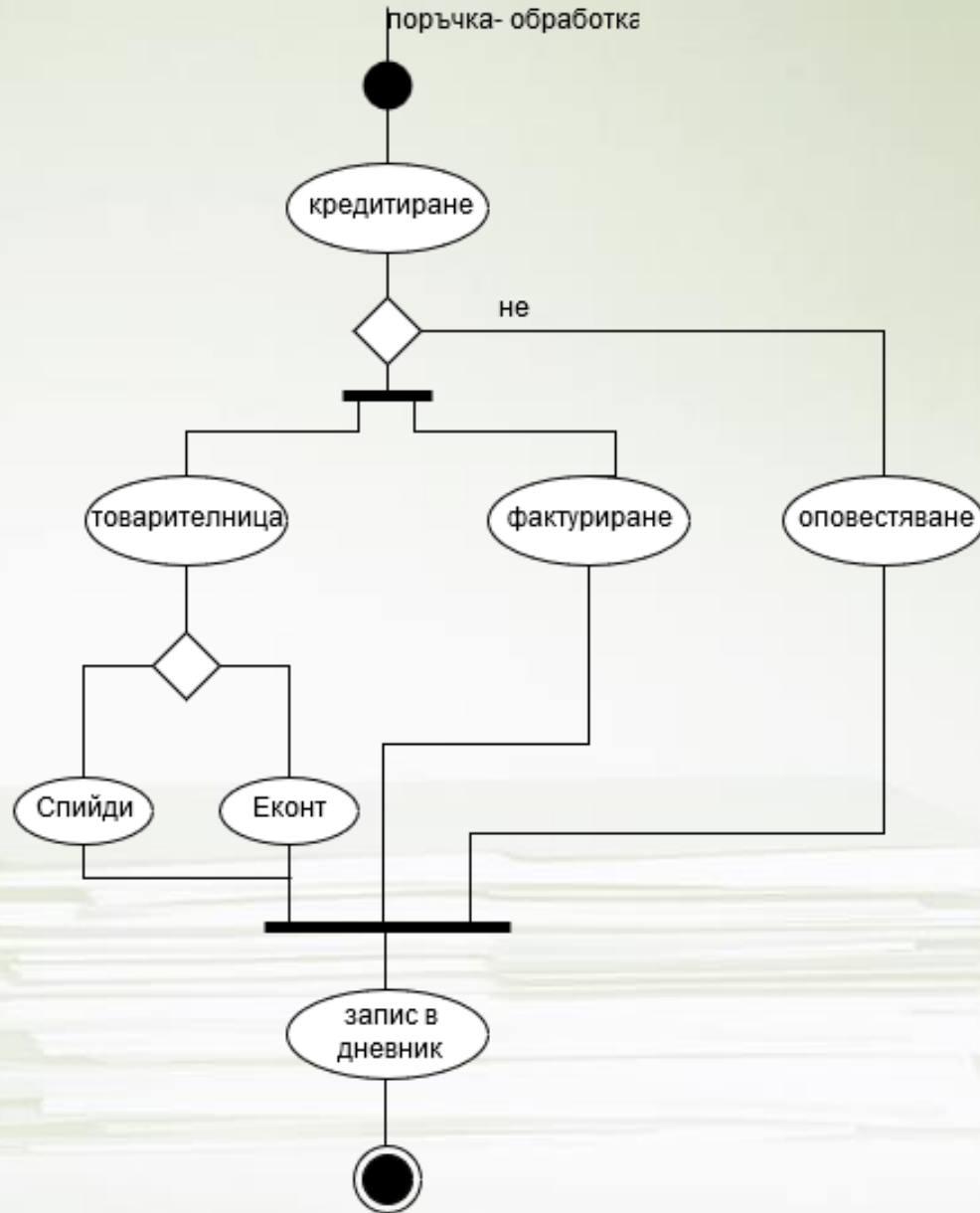


- 4+1 моделиране – представя PCA с 4 основни изгледа и един допълнителен – логически, развоен, процесен и физически + сценарий на приложни/функциониране, който често се придружава и от изглед на потребителските интерфейси Сценарният изглед и асоциираният с него интерфейсен изглед описват потребителските функции на приложението както и основните нефункционални изисквания
 - произтича от потребителското задание
 - в UML се специфицира с диаграма на потребителските случаи Логическият изглед описва декомпозицията на разпределеното приложение с оглед на реализираните функции
 - представя основните блокове или компоненти
 - в UML се специфицира с клас-диаграма (статична), допълнена с една или повече динамични диаграми – най-често последователности

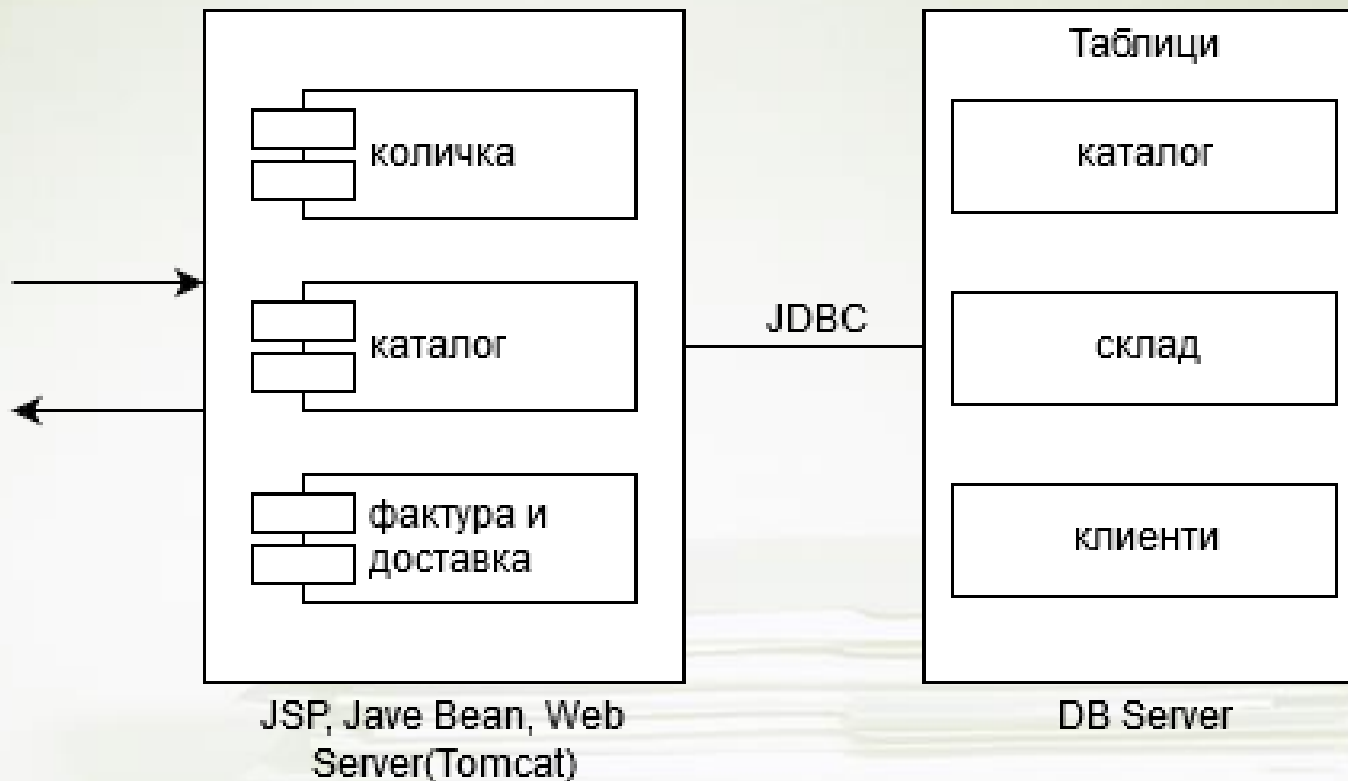
Развоен, процесен и физически изглед

- **Развойният изглед** и асоциираният с него интерфейсен изглед описват потребителските функции на приложението както и основните нефункционални изисквания
 - произтича от потребителското задание
 - в UML се специфицира с **диаграма на потребителските случаи**
- **Процесният изглед** описва декомпозицията на разпределеното приложение с оглед на реализираните функции
 - представя основните блокове или компоненти
 - в UML се специфицира с клас-диаграма (статична), допълнена с една или повече динамични диаграми – най-често **последователности** или **на дейностите**
- **Физическият изглед** описва цялата РСА на платформата + приложението – инсталация, конфигурация, разгръщане
 - компонентите са на ниво услуги/протоколи или процеси
 - връзките между тях са на ниво комуникационни канали
 - представя нанасянето (или картирането – mapping) на компонентите от развойния изглед върху инфраструктурните възли

Процесен изглед с диаграма на дейностите – фиг. 4.34.



Физически изглед – фиг. 4.35.



Фиг. 4.20.2

Потребителски интерфейс изглед

