Упражнение 5. Диаграми на активностите (activity diagrams)

1. **Същност и предназначение**

„**Диаграмата на активност**" (**Activity Diagram**) е **поведенческа** диаграма, тоест тя описва **динамичните** аспекти на системата (ПС).

По дефиниция, диаграмите на активност (activity diagrams) се използват за описване на:

- процедурна логика (алгоритъма, по който се изпълняват операциите на системата и на класовете),
- бизнес процеси (алгоритмите по които работят компаниите)

и

- работни потоци.

При моделиране на поведението на проектираната или анализирана ПС възниква необходимост не само да се представи процеса на изменение на нейното състояние (което става чрез диаграма на състояние), но и **да се разберат и опишат особеностите на алгоритмичната и процедурна реализация на изпълняваните от системата операции**.

a) Диаграми на активност и „блок-схеми"

За тази цел, често се използват „блок-схеми" или структурни схеми на алгоритмите. Всяка от тези схеми акцентира вниманието на последователността на изпълнение на определени процедури или елементарни операции, които в своята съвокупност водят до получаването на желан резултат. В много отношения диаграмите на активност приличат на блок-схеми, но основната разлика между тях и блок-схемите е, че диаграмите на активност могат да моделират „паралелно поведение" за разлика от блок-схемите. Тази диаграма дава възможност да се опишат разклонени и едновременни (паралелни) потоци на системата.

Диаграмата на активност представлява **граф**, чийто върхове са „**състояния",** на **дейност** или на **действие**, а дъгите представят „**преходите"** от едно състояние към друго.

Дейността (или активността) е съвкупност от **действия**, тоест, **действието** е конкретна стъпка от дейността.

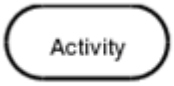
b) Диаграми на активност и диаграми на състояние

Диаграмите на активност се явяват считат за частен случай на диаграмите на състояние. Диаграмата на състояние е предназначена също за моделиране на поведението на системата. Използваната при диаграмите на активност нотация много прилича на тази на диаграмите на състояние, доколкото при тях също присъстват обозначения за „състояния" и „преходи". Разликата е в семантиката на „състояние" и „преход". Всяко „състояние" в диаграмата на активност съответства на изпълнение на някаква **операция** („дейност" или „действие'), а преходът в следващо състояние **става само след приключване на изпълнението на операцията**.

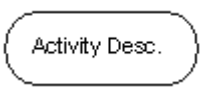
2. **Елементи на диаграмите на активност**

а) Състояние в диаграмите на активност:

Графически, състоянието на активност (Activity State) или на дейност и състояние на действие (Action State) се изобразяват по един и същи начин, чрез правоъгълник със заоблени краища. Вътре в правоъгълника се записва името на състоянието (дейност или действие), което трябва да е уникално в рамките на диаграмата.

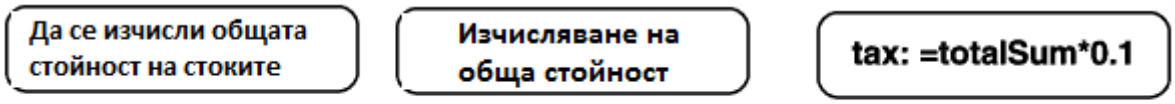


Състояние на Активност или Дейност (Activity State). Активността (Дейността) е множество от действия (actions).



Състояние на Действие (Action State) – това е отделна стъпка (операция) от Дейността (Activity).

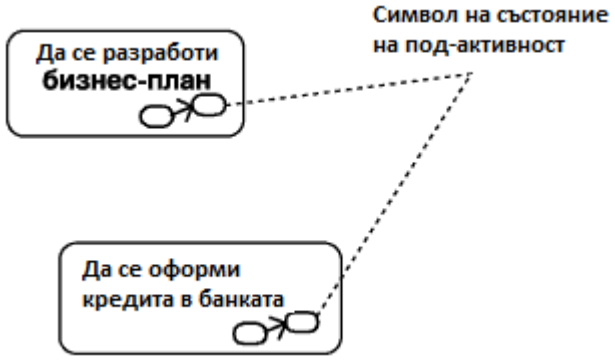
Пример:



(а) (б)

Ако това е състояние на действие, то вътре в правоъгълника се записва действието - на естествен език, на псевдокод или на език за програмиране. Няма ограничения. Препоръчва се (в качеството на име на състояние на елементарна дейност) да се използва **глагол (или съществително) с пояснителен текст** (Пример (а)). Ако действието може да се представи във формален вид – това е препоръчителния начин за запис (Пример (б)).

Състояние на „Под-активност” (Под-дейност)
Понякога, в диаграмата трябва да се представят сложни дейности (действия), тоест такива, които се състоят от няколко други (неатомарни).
В този случай се използва специален елемент – **subactivity state** (състояние на „под-активност”) - състояние в диаграмата на активност, чрез което се обозначава неатомарна последователност от стъпки.



Начално състояние

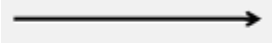

Initial Node (**Начален възел**): съответства на нотацията „**Начално Състояние**” („Initial State”). Отбелязва началното действие от множеството от действия.

Крайно състояние


Final Activity Node (Краен възел): съответства на нотацията **„Крайно Състояние”** („Final State”). Отбелязва завършително действие, което бележи края на всички потоци на управление и потоци на обекти в една диаграма на активност.

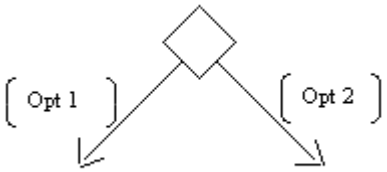
б) Преходи в Activity диаграмите

При построяване на диаграмата на активност се използват само „нетригерни” преходи т. е. такива, които се извършват веднага след завършване на дейността или съответното действие. Такъв преход предава управлението на следващото състояние изведнъж, веднага след завършване на дейността или на действието в предходното състояние. Този преход се изобразява с плътна линия, със стрелка („**Control Flow**”, т.е поток на управление).

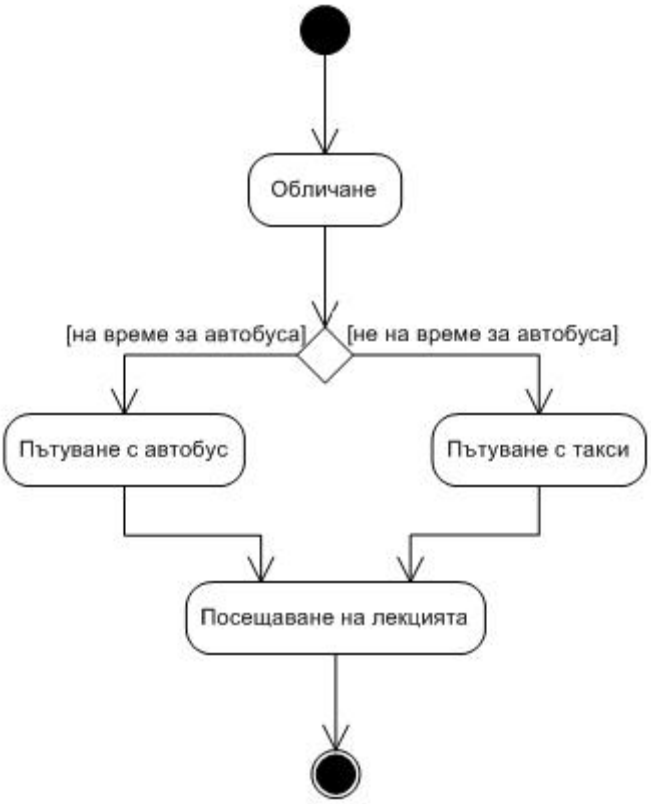

Показва последователността на изпълнение на операциите, на практика работата на системата може да се опише чрез операции.

в) Разклонение в диаграмата на активност (Decision Node)

Графически, разклонението в диаграмата на активност се изобразява чрез „**Decision Node**”, който се изобразява като ромб, вътре в който няма текст.

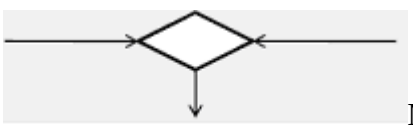


Decision Node – Има единствен входен поток и 2 или повече ограничени (guarded) изходни потока. Всеки изходящ поток има ограничение [guard] - булево условие, поставено в квадратни скоби. Тъй като всеки път, когато се достигне до Decision Node се тръгва само към един от изходните потоци, следва ограниченията да са взаимно изключващи се.
Пример 1: Нека да разгледаме следния пример: за отиването на лекция в 8 часа, сутринта:



Фиг. 1. – Пример за Activity диаграма (отиването на лекция в 8 часа, сутринта)
Както може да се види от фиг. 1, първото действие, което трябва да се извърши е да се облечем, за да може да се изпълни действието посещаване на лекцията. Решението, което трябва да се вземе, зависи от времето на започване на лекцията и времето за пристигане на автобуса до университета. И така, ако имаме достатъчно време, може да пътуваме с автобуса, ако обаче времето не е достатъчно и има вероятност да закъснеем за началото на учебното занятие, тогава ще изберем пътуването с такси, което определено ще е по-бързо. Т.е. двете действия са взаимно изключващи се. Последното действие е посещение на лекцията, след което диаграмата завършва.

г) Възел за Сливане в диаграмата на активност (Merge Node**)**


Merge Node – Свързва няколко клона на разклонение (2 и повече) в един (излиза само един поток).

Пример: На следващата рисунка (http://old.intuit.ru/department/pl/umlbasics/11/2.html#image.11.3) е показана диаграма на активност, която моделира ситуация, възникваща в супермаркетите при заплащане на стоките. Като правило, да се заплати за покупките може или кеш, или чрез кредитна карта. Ако сме избрали кредитна карта, то се проверява „баланса” по картата (наличната сумата в

картата, т.е „баланса” трябва да е по-голяма или равна на стойността на покупката). Заплащането се реализира, само ако стойността на покупките не надвивашава баланса по картата. В противен случай – стоката остава при продавача.

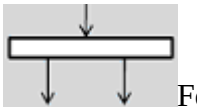
д) Fork Node и Join Node в Activity диаграмите

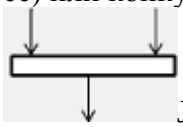
Има действия, които могат да се изпълняват едновременно или паралелно: например, слушане (на лектора) и гледане (на дъската) по време на лекция. Такива действия се наричат **паралелни.** Доколкото паралелните изчисления съществено повишават бързодействието на ПС, то са необходими графични примитиви за представяне на паралелни процеси. Важно е да се отбележи, че когато имаме паралелни операции е необходимо, те да бъдат синхронизирани.

Едни от най-големите недостатъци на блок-схемите или структурните схеми на алгоритмите е свързан с проблеми при изобразяване на паралелни действия и изчисления, тоест липсват графични примитиви за това.

За показване на **начало на паралелно** поведение в UML се използва **Fork Node**, а **за край - Join Node**.

В диаграмите на активностите за това се използва специален символ за показване на разделяне и сливане на паралелни изчисления или потоци за управление: това е хоризонтална черта, аналогична на тази при диаграмата на състояние. В инструменталните средства, тази черта е като куха греда. При това разделянето (fork) има един входен и няколко изходни потока. Сливането (join), обратно, има няколко входни и един изходен поток.


Fork Node – разделя поведение в множество от паралелни (едновременно изпълняващи се) или конкуриращи се потоци от действия.


Join Node - множество от паралелни (едновременно изпълняващи се) или конкуриращи се потоци от действия се събират.

Разгледаните досега преходи са почти достатъчно за моделиране на сложни ситуации.

е) Актьори и диаграми на активност

Activity диаграмите показват какво се случва, но не показват, кой актьор какво действие извършва. Често по съществено е да се наблегне върху това, какво се прави, а не кой го прави.

Ако обаче е необходимо в Activity диаграмите да се покаже, кой какво извършва, тогава може диаграмата да се раздели на дялове, показващи кое действие, от кой актьор се извършва. Този стил се нарича коридори за плуване (swimming pool). Това е така, защото всеки коридор или дял, се изпълнява само от един актьор. Колкото актьори ще има в една система, на толкова дялове трябва да се раздели диаграмата, и всеки дял носи наименованието на актьора.

3. **Декомпозиция на дейности**

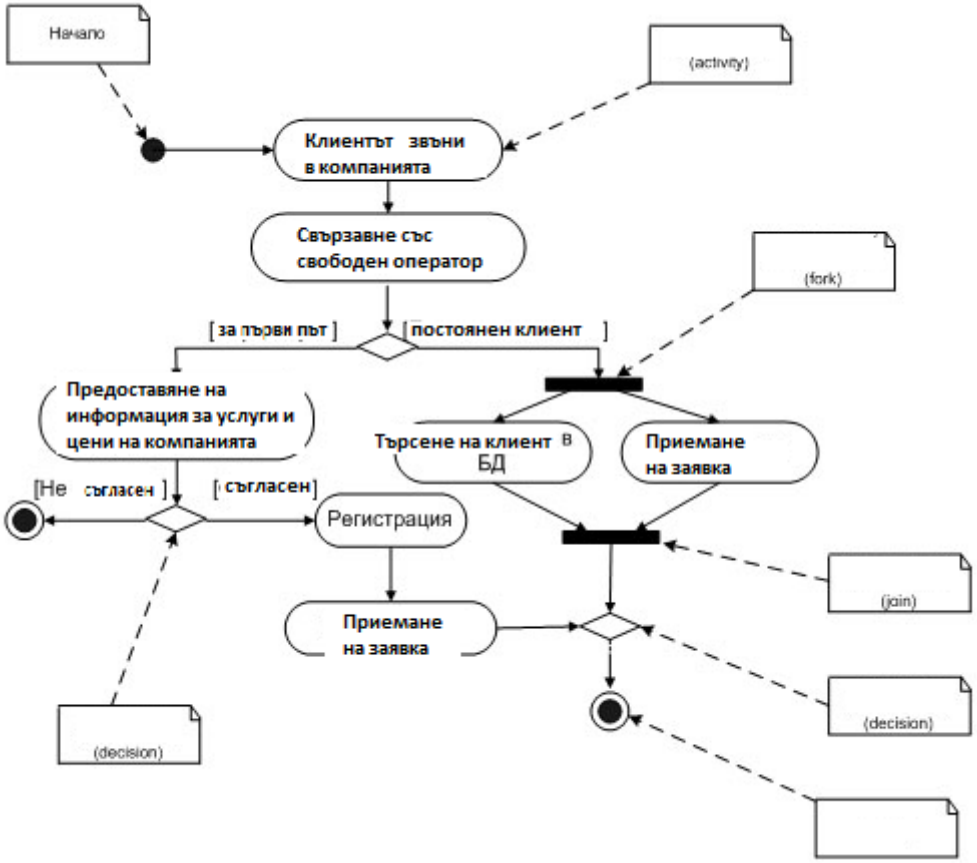
Една дейност може от своя страна да се представи като композиция от дейности, например логиката за доставка може да се представи като **под-дейност** „**Delivery”**, включена в родителската дейност. Идеята е, че дейност „**Delivery”** е съставна част от родителска диаграма, в която имаме действие с име Delivery.

Изобразяване на бизнес процеси

Според http://old.intuit.ru/department/se/vismodtp/3/vismodtp_3.html, с помощта на Actiivity Diagram удобно могат да се изобразят бизнес-процеси, тоест алгоритмите по които работи една компания. Именно, в името на бизнес-процесите, се взема решение за разработка на ИС, с идеята – да се вгради тази ИС в бизнес процеса, автоматизирайки някаква част от него.

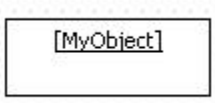
Да се върнем към примера от предни лекции (и упражнения), свързан със „Система за обслужване на клиенти по телефонни заявки”. За да се реализира тази система, в компанията трябва да бъде създаден нов бизнес-процес, свързан с „телефонна обработка на заявките”, който трябва да се вгради

в съществуващия. Възложителят си представя този бъдещ процес по някакъв начин, а разработчиците се опитват чрез диаграма на активност да го опишат (тоест да опишат алгоритъма на работа на новата служба, свързана с телефонна обработка на заявките), с цел той да бъде обсъден с възложителите и коректно разбран от всички заинтересовани...
На следващата рисунка (http://old.intuit.ru/department/se/vismodtp/3/vismodtp_3.html#image.3.4) е показана обща схема на работата на оператора с клиент, под формата на Activity Diagram:



Пример на диаграма на активностите на бизнес-процес, свързан с „телефонна обработка на заявките”
За програмистите (http://old.intuit.ru/department/se/vismodtp/3/vismodtp_3.html) е полезно да си представят ясно всички бизнес-процеси в компанията. В дадения случай, в компанията съществува бизнес процес по телефонна обработка на заявките, който трябва да бъде разбран от разработчиците, тоест – създаването на правилна диаграма на активност е особено важно. Иначе може да се окаже че е изпуснат важен детайл, който няма да позволи на новата система пълноценно да изпълнява своите функции. Например, може да се окаже, че подсистемата за обработка на завките, с която трябва да се интегрира новосъздаваната система е реализирана чрез макроси, на Word/Excel! Очевидно, е че да се интегрираш с такава система ще е трудно. На този и други факти е нужно да се обърне внимание, да се каже това на възложителя, тъй като иначе проектът може да се окаже неуспешен – възложителят да загуби пари без да получи нужните за своя бизнес услуги.

4. **Още елементи в Activity диаграмите:**
   a) **Обекти, в диаграмата на активност:**



Object Node – представя обект, който се свърза към състояние на дейност или действие с Object Flow

Object Flow.

Действията в диаграмата на активност могат да се извършат над едни или други обекти. Тези обекти или инициират изпълнението на действията или са резултат от тези действия.

Обектите се представят с правоъгълник, като името на обекта е подчертано. В диаграмата на активност, след името на обекта може, а понякога е задължително, да се укаже и състоянието на обекта (в квадратни скоби). Обектите се свързват (тоест присъединяват се) към състояние на дейност или действие чрез отношение на зависимост – пунктирана линия със стрелка. Съответстващата зависимост определя състоянието на конкретния обект след изпълнение на предшестващото действие.

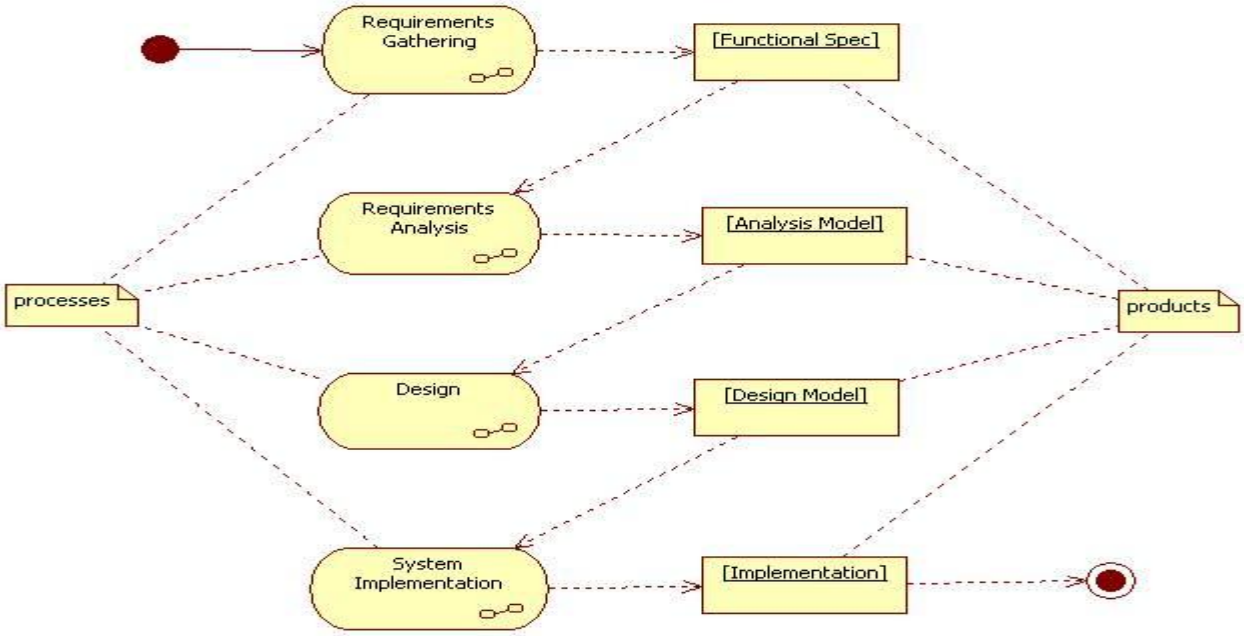
Ако една диаграма на активност е разделена на дялове, тоест има коридори (броя на коридорите зависи от броя на актьорите), то разположението на обекта може да има допълнителен смисъл, а именно: ако обектът е разположен на границата на 2 коридора, то това може да означава, че преходът в следващото състояние на действие в съседния коридор е асоцииран с намиране на обекта в някакво състояние (както в случая, преминаване към дейност „регистриране на поръчка" има само, ако е налице попълнена „поръчка" т.е. налице е „запис за поръчка в БД"). Ако обектът е вътре в коридора, то състоянието на този обект изцяло се определя от действията в този коридор.

Приложено към диаграмата на активност, обектите по правило се явяват екземпляри на класове същности, или на бизнес-класове. Още, един и същ обект, в тази диаграма може да се изобрази няколко пъти, като трябва обаче да се указват за обекта различни характеристики на състояние.

Пример: Процесът на разработка на софтуер може да се опише по следния начин:

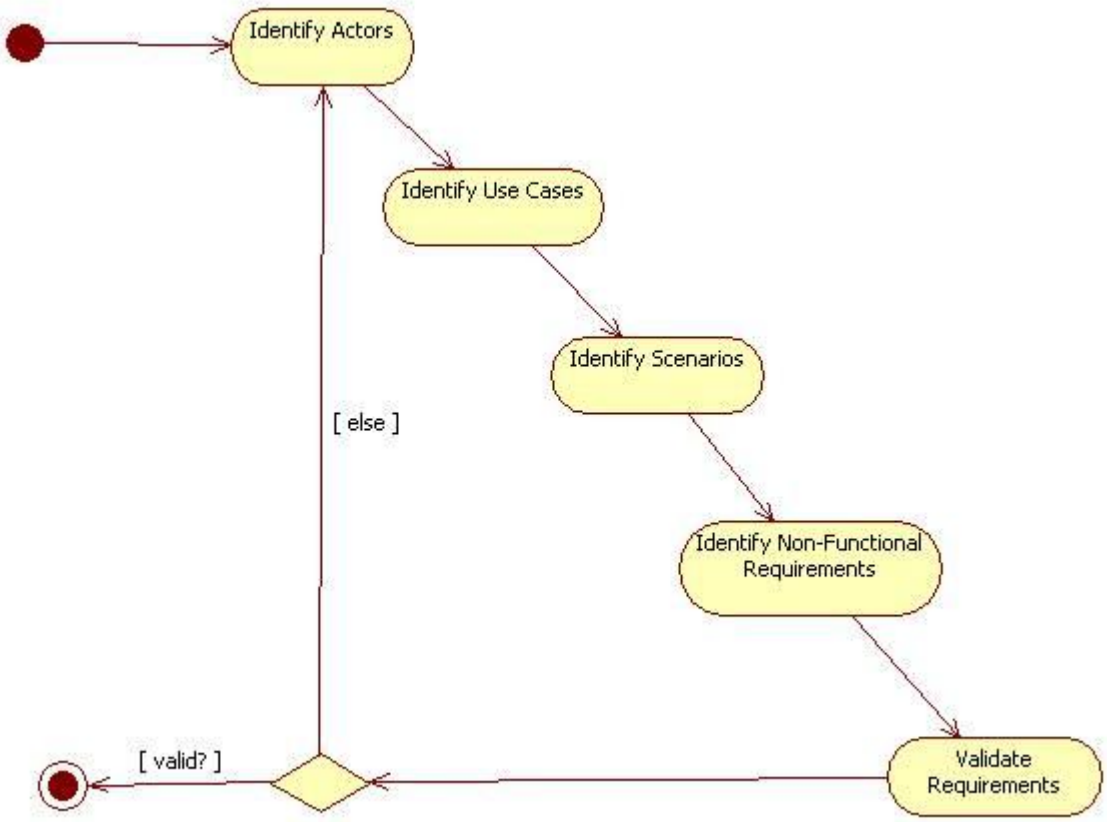
- Извличане на изискванията (Requirements Gathering).
- Анализ на изискванията (Requirements Analysis)
- Проектиране (System Design)
- Разработка на системата (System Implementation)

Ето една диаграма на активностите, показваща връзките и резултатите (продуктите) от тези процеси:



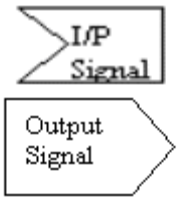
Всички заинтересовани лица (възложител, потребители и разработчици) участват в процеса на извличане на изискванията, докато само разработчиците участват в останалите процеси. Резултатът (продуктът) от Requirements gathering е "функционална спецификация" (Functional specification document), която е входен документ за следващата фаза - „анализ на изискванията". Резултатът от фаза Requirement Analysis е „Analysis Model". "Functional specification document" плюс "Analysis Model" формират SRS (Software Requirement Specification), която е вход за следващата фаза – „Design" и т.н.

От своя страна, процесът на извличане на изискванията (**Requirements gathering**) е комбинация от 5 основни дейности, които могат да се представят чрез следната диаграма на активности:



b) сигнали в Activity диаграмите:

Когато едно действие изпраща или приема съобщения, тогава действието се нарича „**сигнал**”. Сигналите биват:



или

a. входен сигнал (действието получава съобщение отвън) – вдлъбнат многоъгълник

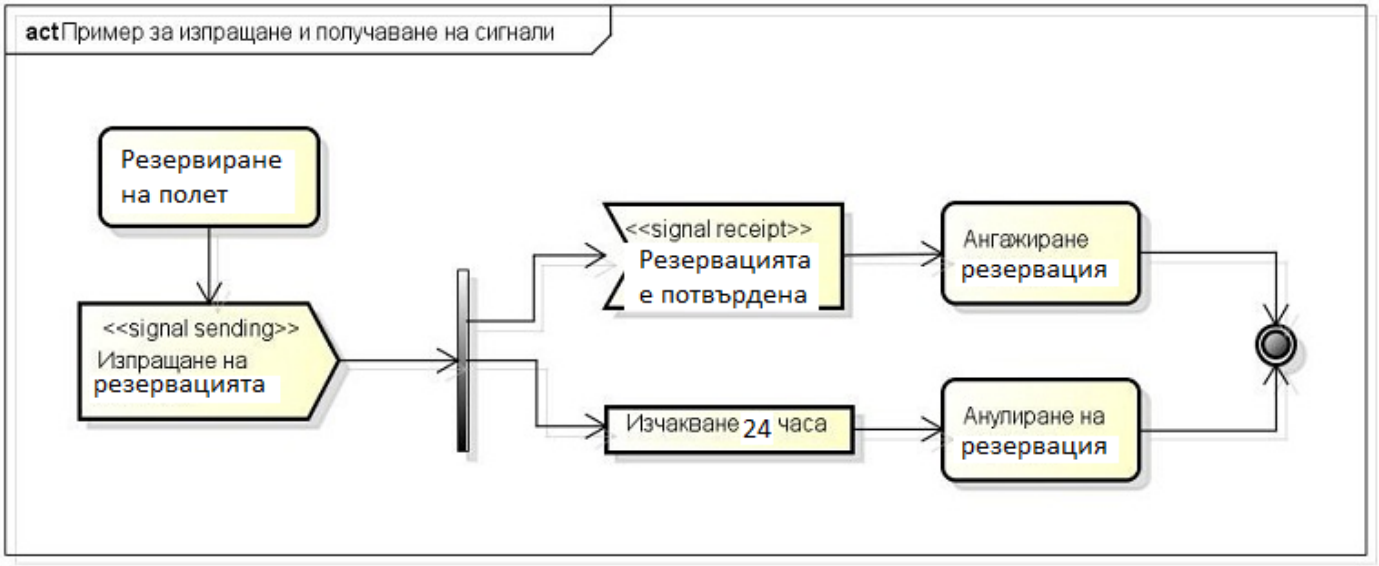
b. изходен сигнал (действието изпраща съобщение) – изпъкнал многоъгълник.

Пример: Представя се пример за диаграма на активност (за online „Резервация на полет”), при която действията са сигнали (тоест, изпращат или получават сигнали).
Имаме:

- Сигнал за изпращане (**Signal Sedning**)
- Сигнал за получаване/приемане/ (**Signal Receip**)

Диаграмата показва и трети вид сигнал:

- Сигнал за време (Time Signal) - Настъпва поради изменение на определено време.



В заключение:

Същността на тези диаграми е ***активността*** (activity), тоест **състояние на системата, в което тя изпълнява някаква работа**. След завършването на една активност следва преход към друга активност. Вижда се какви елементи има тази диаграма:

- На диаграмата имаме символи за начало (start) и край (finish).
- На диаграмата може да има и паралелно разклонение (fork), което пуска няколко едновременно паралелно работещи клона. Такива клонове могат да се обединяват (всички или част) чрез конструкция, наречена паралелен съединител (join).
- В диаграмата могат да се използват символи за логическо разклонение (decision) и съединение (merge). По клоновете, следващи логическото разклонение се обозначават условията за преход.

Достойнство на диаграмата на активност се явява възможността да се визуализират **отделни аспекти на поведението на** разглежданата система или да се реализират **отделни операции на класовете** във вид на процедурна последователност от действия. По този начин, **пълният Activity модел на системата може да съдържа една или няколко диаграми на активност**, всяка от които описва последователността на реализацията или най-важните варианти на използване или на нетривиални операции на класовете.

Виж. „Програмни спецификации . Ръководство за лабораторни упражнения” тема V.

Използвана литература:

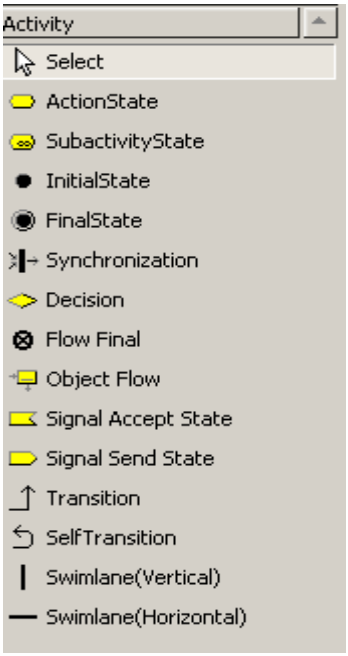
Фаулър и http://194.141.86.222/lecture/rkraleva/LetenSem/SoftTech/Ypr_SoftTech/st11.htm,
http://old.intuit.ru/department/pl/umlbasics/11/2.html#image.11.3
http://www.cs.sjsu.edu/~pearce/modules/lectures/ooa/requirements/index.htm

Помощни сайтове:

https://www.lucidchart.com/pages/uml/activity-diagram
http://www.evanetics.com/Articles/ar_objectModeling/saneUML.htm
http://flylib.com/books/en/4.445.1.113/1/

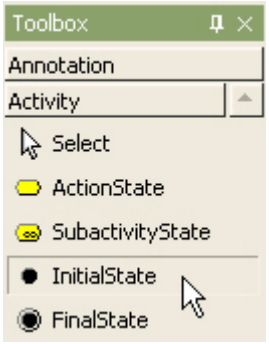
Приложение:

Ето ги графичните елементи на Activity Diagram в StarUML:



Процедура за създаване на начално състояние

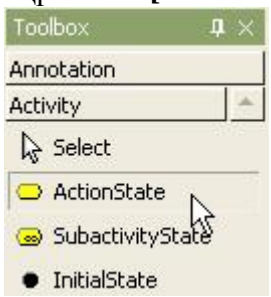
1. Щракнете **[Toolbox]** -> **[Activity]** -> **[InitialState]** бутон.



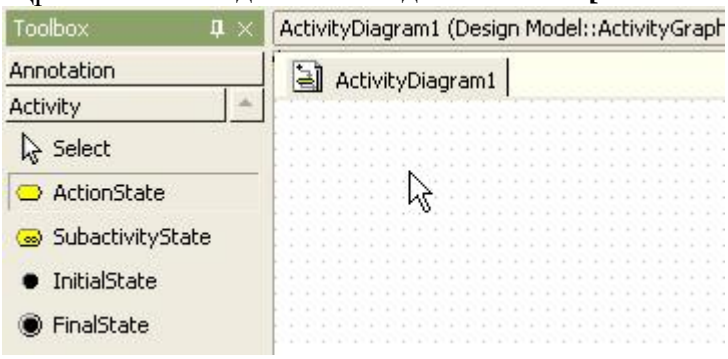
2. Щракнете там където искате да бъде InitialState в **[main window]:**

Процедура за създаване на action state (активно състояние)

1. Щракнете **[Toolbox]** -> **[Activity]** -> **[ActionState]** бутон.



2. Щракнете там където искате да се появи в **[main window]**.



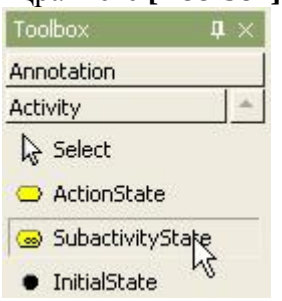
3. И „action state” се създава в диаграмата, и quick dialog се показва.

4. Въведете име на action state в „quick dialog” и натиснете **[Enter]:**

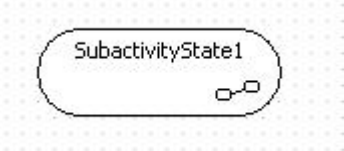
SubactivityState (Състояние на под-активност)

Едно състояние на под-активност (subactivity) представлява изпълнение на не-атомарна поредица от стъпки, която има някаква продължителност, т.е. вътрешно то се състои от набор от действия, която вероятно чака настъпване на събития.

1. Щракнете **[Toolbox] -> [Activity] -> [SubactivityState]** бутон.



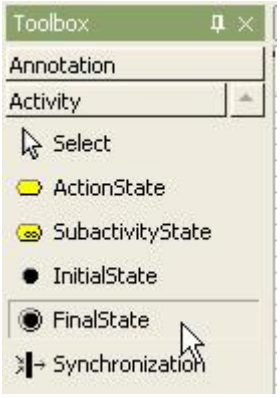
2. Изберете място в **[main window]** и щракнете. Може да сложите име в показания quick dialog и да натиснете **[Enter]**. Резултатът е:



Крайно състояние (FinalState)

Процедура за създаване на final state

1. Щракнете **[Toolbox] -> [Activity] -> [FinalState]** бутон.



2. Щракнете, където искате да бъде поставено в **[main window]**.

Decision (Решение)

A state diagram (and by derivation an activity diagram) expresses a decision when guard conditions are used to indicate different possible transitions that depend on Boolean conditions of the owning object.

Процедура за създаване на decision

In order to create Decision,

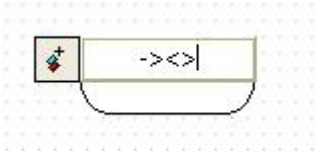
1. Щракнете **[Toolbox] -> [Activity] -> [Decision]** бутон.

2. And Щракнете at the position where Decision will be placed in the **[main window]**. The decision is created on the diagram.

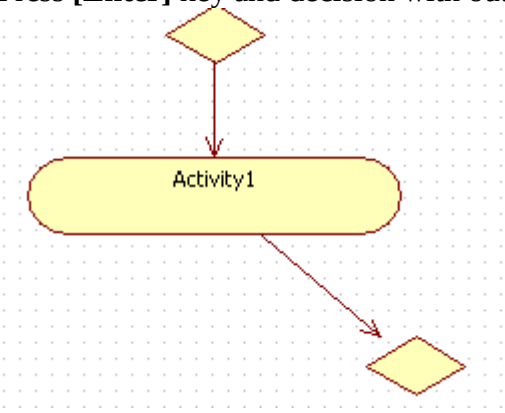
Процедура за създаване на decision от състояние

За да се създаде „decision” с вход - избран обект, използвайте „shortcut creation syntax”:

1. Два пъти-Щракнете в състоянието и в „quick dialog”-а, въведете "-><>"("<-<>" за изход от decision) стринг.



2. Press **[Enter]** key and decision with outgoing transition from selected state is created.



Flow Final (Край на поток)

Процедура за създаване на flow final

In order to create Flow Final,

1. Щракнете **[Toolbox] -> [Activity] -> [Flow Final]** бутон.

2. Щракнете позицията, където ще се намира края на потока (Flow Final) в **[main window]**:

Object Flow (Поток от обекти)

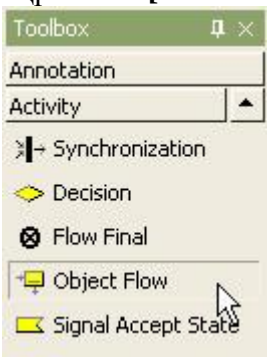
Семантика

Един поток от обекти е единият от 2-та типа дъги в диаграмата на активности и указва директната връзка (потоците) между възлите на активност, докато другия тип дъги – указва потока на управление между възлите на активност. Един поток от обекти носи само данни за обекта, не може да изпълнява контролни функции.

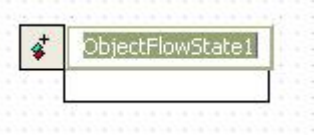
Процедура за създаване на object flow

За да се създадеObject Flow,

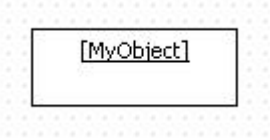
1. Щракнете **[Toolbox] -> [Activity] -> [Object Flow]** бутон.



2. Щракнете, където искате да го поставите в **[main window**] и му задайте име в quick dialog-a.



3. Натиснете **[Enter]** .

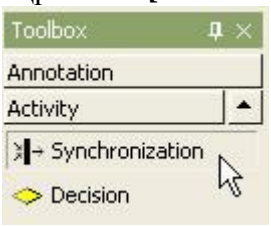


Synchronization (Синхронизация)

Процедура за създаване на synchronization bar

In order to create Synchronization,

1. Щракнете **[Toolbox] -> [Activity] -> [Synchronization]** бутон.



2. Щракнете, където желаете да се появи в **[main window]**..

3. Ще се получи следната фигура.

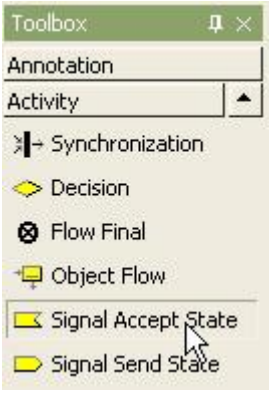
Signal Accept State (Състояние на приемане на сигнал)

Семантика

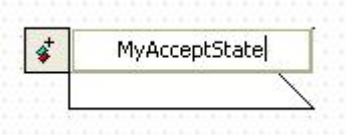
Приемането на сигнал може да бъде показано с един назъбен правоъгълник (пентагон), както е показано по-долу. Сигнатурата на сигнала е показана вътре в символа. Една не-надписана „transition arrow” се рисува от предното състояние (action state) към назъбен правоъгълник и една друга не-надписана „transition arrow” се рисува до следващото „action state”. Една пунктирана стрелка може да нарисува от един обект до жлеба на пентагона, за да покаже изпращача на сигнала, но това не е задължително.

Процедура за създаване на състояние на приемане на сигнал („signal accept state”)

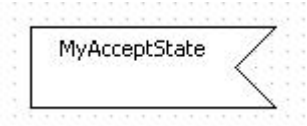
1. Щракнете **[Toolbox]** -> **[Activity]** -> **[Signal Accept State]** бутон.



2. Щракнете, там където искате да се постави в **[main window]**.



3. Напишете името на състоянието за приемане на сигнал и натиснете **[Enter]**.

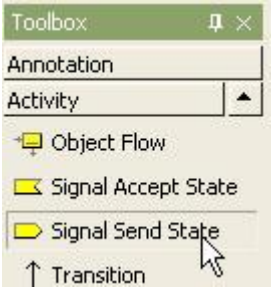


Signal Send State (Състояние на изпращане на сигнал)

Указва се с изпъкнал правоъгълник. Сигнатурата на сигнала се показва вътре в символа. Една не-надписана „transition arrow” се рисува от предното „action state” до пентагона, а друга от пенатагона до следващото „action state”. Отново, пунктирана линия може да се нарисува от изпъкналата точка на пентагона до един обект (an object symbol) за ад се покаже получателя на сигнала.

Процедура за създаване на signal send state

1. Щракнете **[Toolbox]** -> **[Activity]** -> **[Signal Send State]** бутон.

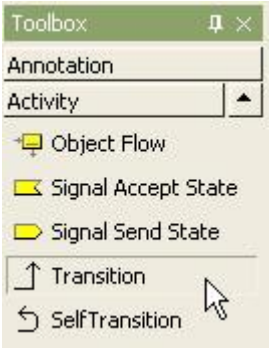


2. Щракнете, там където искате да се постави в **[main window]**:

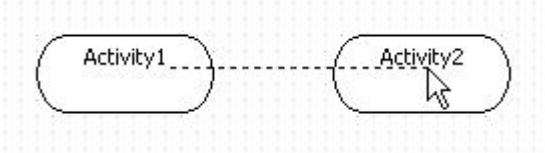
Transition (Преход)

Процедура за създаване на преход (transition)

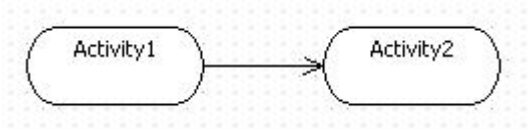
1. Щракнете **[Toolbox]** -> **[Activity]** -> **[Transition]** бутон.



2. Дръпнете и пуснете между състоянията в посока на прехода (in transition direction) в **[main window]**.



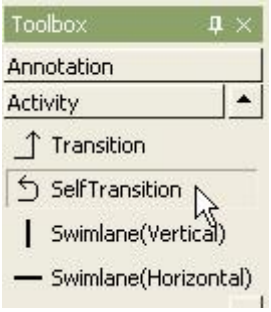
3. Резултат:



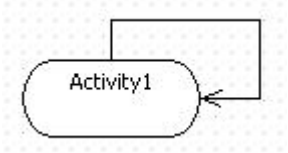
SelfTransition (Преход към себе си)

Процедура за създаване на self-transition

1. Щракнете **[Toolbox]** -> **[Activity]** -> **[SelfTransition]** бутон.



2. Щракнете състоянието за което ще се отнася в **[main window]**. Резултат:

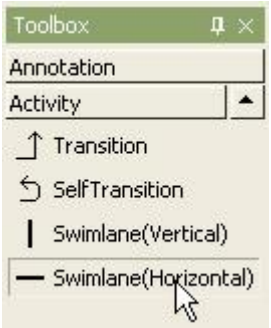


Swimlane

Действията (Actions) и под-дейностите или под-активности (subactivities) могат да бъдат организирани в „swimlanes”. „Swimlanes” се използват за организиране на отговорностите на действия и по-активности. Те често кореспондират на организационна една в бизнес модела.

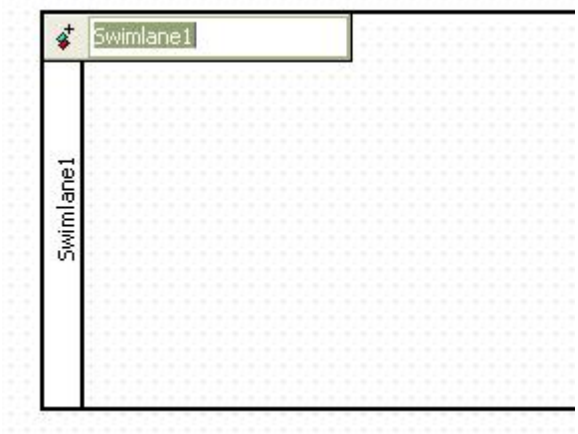
Процедура за създаване на хоризонтална swimlane

1. Щракнете **[Toolbox] -> [Activity] -> [Horizontal Swimlane]** бутон.



2. И издърпайте (drag) границата в която Horizontal Swimlane ще се разположи в **[main window]**.

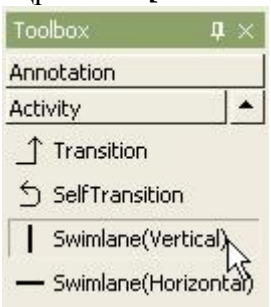
3. Така, horizontal swimlane се създава в диаграмата. Въведете името на swimlane в „quick dialog" и натиснете **[Enter]**.



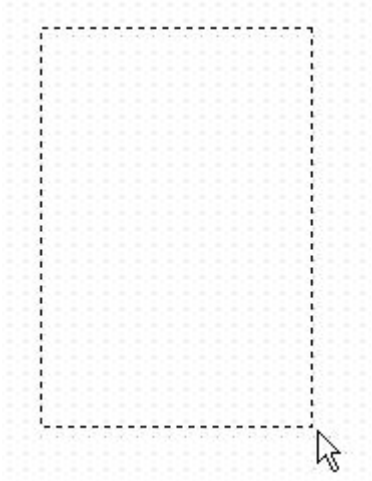
Процедура за създаване на vertical swimlane

Забележка: процедурата е подобна като при хоризонтална swimlane

1. Щракнете **[Toolbox] -> [Activity] -> [Vertical Swimlane]** бутон.



2. Очертаваме границата в **[main window]**.



3. В quick dialog-а слагаме име и натискаме **[Enter]:**