Softwarové inženýrství

3. přednáška

Radek Janoštík

Univerzita Palackého v Olomouci

9. 10. 2025

Outline

- Modelování
- UML – Unified Modeling Language
- UML – diagramy
- Doporučená četba

Modelování

- Jednoduché činnosti je člověk schopen uchopit celé
  - Dokáže si představit všechny možnosti, problémy a v hlavě je vyřešit
- Příklad stavby:
  - *Psí bouda* – vezmete hřebíky, kladivo, pilu a pár desek. Stlučete psí boudu
  - Pokud nejste úplně nezruční, výsledek splní účel. Pes bude spokojen
  - *Přístřešek pro auto* – vezmete hřebíky, kladivo, pilu, pár desek a trámů
  - Na papír si narýsujete základní rozměry, sklony, ...
  - Uděláte přístřešek na auto
- Jak budete postupovat při stavbě domu? Velké budovy?
  - Různé plány, papírové modely, vizualizace
- U velkých projektů nedokáže člověk nahlédnou do veškeré problematiky
  - Není schopen odhalit dílčí problémy vs. odhalení koncepčních chyb

Modelování

- Modelování = zjednodušení reality
  - Oprostíme se od některých problémů abychom viděli ostatní
  - Různé pohledy na realitu, různé úhly, odhalení různých problémů
- Zjednodušené modely jsou snadněji uchopitelné pro lidi
- Lepší náhled na celkový systém (stavbu, SW)
- Pohled (a přehled) na celkovou strukturu
- Model slouží jako šablona pro reálný systém $\Rightarrow$ víme jak (a kam) postupovat
- Zároveň slouží jako dokumentace

Modelování SW

- Máme vývojářské ambice na obrovskou komplexní budovu, ale často modelujeme „jen boudu“
  - Můžeme mít štěstí a povede se nám to
  - Podaří se nám to i u dalšího produktu?
- Modelování při návrhu SW se ukazuje jako jeden ze společných jmenovatelů úspěšných SW produktů
  - Uznávaná inženýrská technika
- Různé typy modelů pro různé SW problémy
  - Jako celek jsou dobrým základem pro SW
- Primární je stále SW produkt
  - Dokumentace a konfigurace bývá vždy „až“ sekundární
  - Problém: Sekundární $\neq$ zbytečná

Principy modelování

1. Volba vhodného modelu ovlivní odhalené problémy a způsob jejich řešení
   - Vhodný pohled „na svět“ (Databáze, OOP, struktury)
   - Co nedokážeme (již) uchopit precizně, pojďme vhodně namodelovat
2. Různé modely je potřeba dělat s různou precizností
   - Potřebujeme mít jen letmý pohled na architekturu systému?
   - Potřebujeme vyřešit konkrétní interakci dvou komponent?
3. Modely musí být úzce spjaty s realitou
   - Používat reálné komponenty a „možné věci“
4. Jeden model nestačí – pro komplexní systém je potřeba více modelů z více „úhlů“
   - Jak je strukturována architektura systému
   - Model chování uživatele (co se dá dělat?)
   - Jakým způsobem plynou data
   - ...

Historie modelovacích jazyků

- Modelovací jazyk = soubor pravidel, kterými se řídíme při modelování
  - Známe-li pravidla, jsme schopni porozumět modelům od kolegů
  - Jazykové definice, grafické definice
  - Syntax a sémantika. Často grafické znázornění
- S nárůstem programovacích jazyků vzniká potřeba modelovat nezávisle na jazyku
- 80. a 90. léta vznik několika metod
  - Boochova metoda (návrh a konstrukce)
  - Object-Oriented Software Engineering – Jacobson (případy užití, požadavky)
  - Object Modeling Technique – Rumbaugh (datová analýza)
- Spolupráce na ujednocení $\Rightarrow$ UML
  - Netříštit síly, jednomu jazyku více lidí porozumí
  - Standardizace
  - Vylepšení všech metod
- Potřeba modelovat všechny fáze SW procesu
- Použití pro lidi, ale i pro stroje (spolupráce, nástroje)

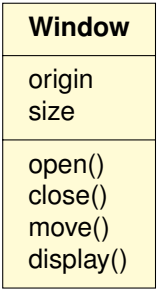
Úvod do UML (= Unified Modeling Language)

- Standardizovaný jazyk pro modelování návrhů SW
- Slouží k vizualizaci, specifikaci a dokumentování konstruktů a prvků v SW systémech
- Umožňuje modelovat různé typy aplikací – ERP, Webové aplikace, embedded
- Relativně jednoduché čtení (intuitivní porozumění)
- „Psaní“ trochu náročnější – stanovená „slovíčka“ a „gramatika“
- Vizualizace – stanovení sémantiky nákresů, standardizace
  - Kombinováno s textovým popisem
- Idea: Z přesných modelů si usnadníme konstrukci
  - Generátory zdrojových kódů
  - Opačný směr komplikovanější (co vynechat)

Základní prvky UML – Strukturální prvky – klasifikátory

- Podstatná jména jazyka – zobrazují koncepty nebo fyzické věci

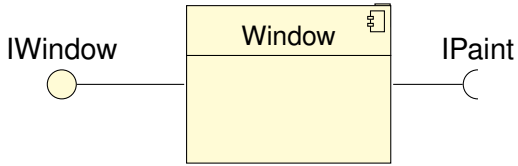
1. *Třídy* – formální popis objektů mající stejné vlastnosti, operace, vztahy a význam
   - Mohou implementovat (jedno a) více rozhraní
   - Zobrazení: *Obdélník se jménem v prvním řádku, vlastnostmi a operaci v dalších*

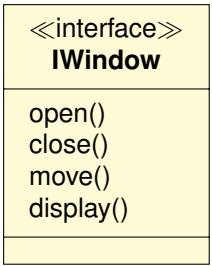


2. *Aktivní třídy* – Třídy obsahující více procesů či vláken – konkurence.
   - Zobrazení: *Stejně jako třídy, levý a pravý rámeček zdvojený*

Strukturální prvky – klasifikátory

3. *Rozhraní* – Kolekce operací popisující poskytovanou službu třídy či komponenty
   - „Externí pohled“, bez implementace
   - Zobrazení: Definice rozhraní: *Stejně jako třídy s klíčovým slovem «interface»*
   - Zobrazení: Poskytované rozhraní: *Malé kolečko s názvem, spojené čarou se třídou*
   - Zobrazení: Požadované rozhraní: *Malé půlkolečko s názvem, spojené čarou se třídou*

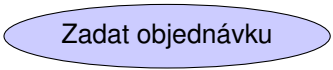




Strukturální prvky – klasifikátory

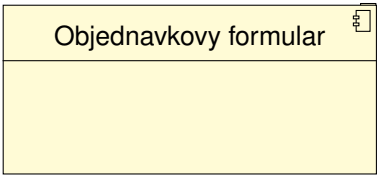
4. *Spolupráce* – Popisuje interakce mezi elementy poskytující společnou funkcionalitu
   - Funkcionalita je větší než součet funkcionalit zúčastněných prvků
   - Často slouží k popisu zvoleného *návrhového vzoru*
   - Zobrazení: *Čárkovaná elipsa s názvem uvnitř*

5. *Případ užití* – popis sekvence akcí s výstupem, které by měl systém umožňovat
   - Zobrazení: *Elipsa s plným okrajem s názvem případu užití*



Strukturální prvky – klasifikátory

6. *Komponenty* – modulární součásti systému, skrývají svou implementaci za rozhraní
   - Při zachování rozhraní mohou být snadno nahrazeny jinými
   - Zobrazení: *Jako třídy se speciální ikonou v pravém horním rohu*

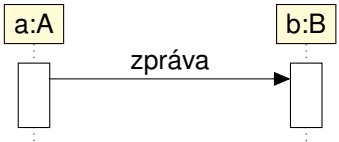


7. *Artefakty* – Části systému s fyzickou informací – soubory, knihovny, exe soubory
   - Zobrazení: *Jednoduchý obdélník s klíčovým slovem «artifact» a názvem*

Chování v UML

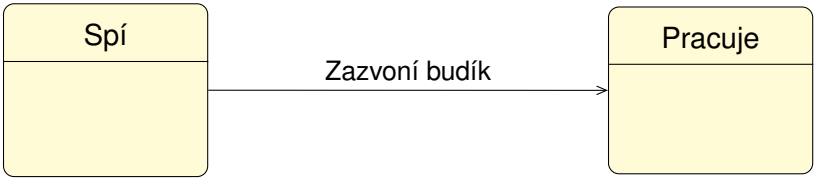
- Dynamické části jazyka UML – slovesa popisující chování v čase a prostoru
- Používány s propojením se strukturálními prvky jazyka

1. *Interakce* – popisuje soubor zpráv, které si objekty vyměňují pro dosažení cíle
   - Zobrazení: *Přímá šipka s popisem zprávy*



Chování v UML

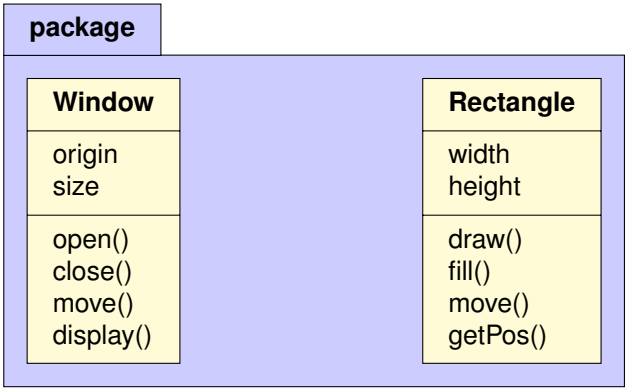
2. *Stavový stroj* – popis sekvence stavů objektu, kterými si během svého života může projít
   - Řízen událostmi
   - Obsahuje stavy a šipky jako přechody (podobné jako DKA)
   - Zobrazení: *Obdélník s kulatými rohy se jménem stavu. Šipka s popisem přechodu*



3. *Aktivita* – popis sekvence *akcí*, které systém během života provádí
   - Důležité je, který objekt provádí dané akce
   - *Akce* – základní krok v aktivitě
   - Zobrazení akce: *Obdélník s kulatými rohy se jménem stavu* (stejně jako stav, rozhoduje kontext)

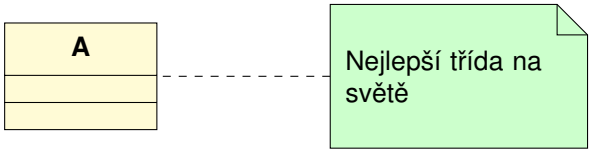
Seskupování objektů

- Prvky je pro přehlednost potřeba organizovat do souvisejících skupin
- Rozdělení modelu do *balíčků*:
  - Slouží pouze k organizaci návrhu nikoliv implementace
  - Existuje pouze v době vývoje, při nasazení jeho význam zaniká (vs. komponenta)
  - Zobrazení: *Obdélník jako „složka se jmenovkou“ obsahující další objekty*



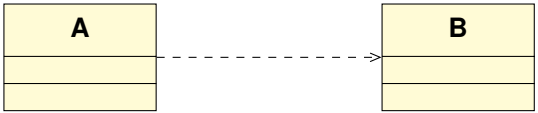
Poznámky

- Slouží k upřesnění všech částí UML modelů
- Textová specifikace podmínek, formální i neformální komentář
- Popis vyžadovaných komponent/akcí (odkaz na jiný model)
- Zobrazení: *Obdélník s textem (či obrázkem) s přehnutým rohem, čárkovaná čára k objektu*



Základní prvky UML – vztahy

1. *Závislosti* – sémantický vztah mezi prvky
   - Změnou *nezávislého* může být pozměněna funkcionalita *závislého*
   - Zobrazení: *Čárkovaná šipka orientována od závislého*



2. *Asociace* – strukturální vztah mezi prvky
   - Specifikace, jak spolu objekty souvisí
   - Často s popisem vztahu, typem vazby (1:N, M:N) a vlastnostmi
   - Účastníci jsou v modelu na stejné úrovni
   - Zobrazení: *Jednoduchá čára s popisky*

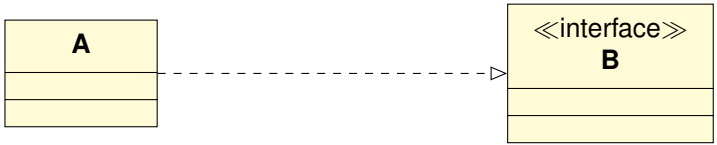
Základní prvky UML – vztahy

4. Agregace – speciální typ asociace modelující vztah „has-a“
   - Účastníci nejsou na stejné úrovni
   - Zobrazení: Čára s kosočtvercem u nadřízeného prvku

5. *Generalizace* – popis obecnosti/konkrétnosti prvků
   - Potomek (konkrétnější) sdílí strukturu a vlastnosti rodiče (obecnější)
   - Zobrazení: *Čára s trojúhelníkem u obecnějšího prvku*

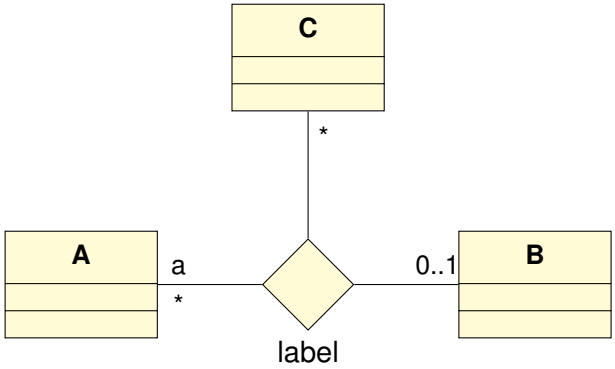
Základní prvky UML – vztahy

6. *Realizace* – významový vztah, jeden prvek vyžaduje funkcionalitu, druhý ji zaručuje
   - Významově mezi závislostí a generalizací
   - Používáno u rozhraní a spoluprací
   - Zobrazení: *Čárkovaná šipka s trojúhelníkem u rozhraní či případu užití*



N-ární vztahy

- Veškeré vztahy mohou být mezi více objekty než dvěma – n-ární vztahy
  - Zobrazení: *Kosočtverec na křižovatce vztahů*



Hello World v UML

- Mějte program v jazyce Java:

```
import java.awt.Graphics;
class HelloWorld extends java.applet.Applet {
   public void paint(Graphics g) {
      g.drawString("Hello, World!", 10, 10);
   }
}
```

- Pojďme nakreslit pár modelů (na tabuli)

Základní prvky UML – diagramy

- Diagramy slouží ke vizualizaci částí systému z různých úhlů pohledu
- Graf, kde jsou uzly (strukturální prvky) propojeny cestami (vztahy)
- Úhly pohledu jsou standardizovány $\Rightarrow$ vznik 13 základních, jasně definovaných diagramů
- Každý se zaměřuje na jinou část problematiky SW procesu
- S dodržením pravidel UML lze dělat i jiné diagramy

Diagram tříd

1. *Diagram tříd* – zobrazení tříd, rozhraní a spolupráce a vztahy mezi nimi
   - Nejběžnější diagram, často lze generovat z (OO)kódu a naopak
   - Zobrazuje pouze statickou hierarchii, ne změny, stavy, procesy

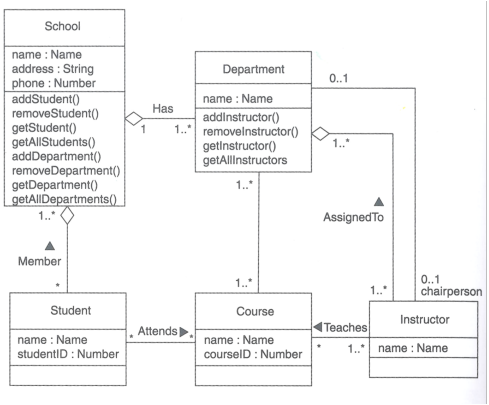


Diagram komponent

2. *Diagram komponent* – dokumentuje zapouzdřené třídy a jejich rozhraní
   - Prvky jsou konkrétní komponenty zajišťující funkcionalitu pro uživatele
   - Komponenty mohou být vnořené ⇒ *diagram složených struktur*

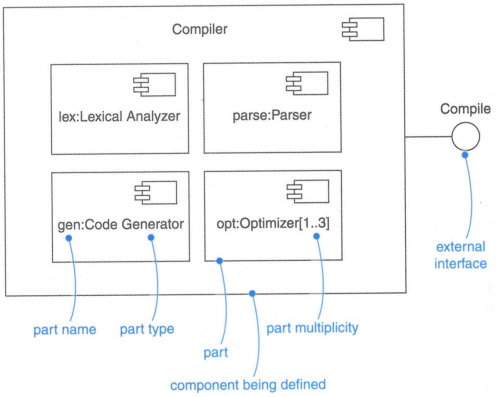


Diagram objektů

4. *Diagram objektů* – zachycení objektů z diagramu tříd v konkrétním čase
   - Zachycuje stav před/po interakci
   - Náhled na složitější datové struktury

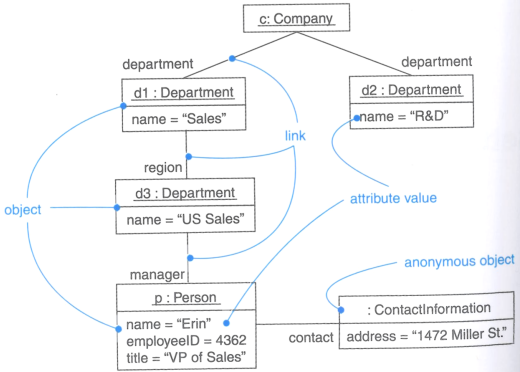


Figure 14-1: An Object Diagram

Diagram artefaktů

5. *Diagram artefaktů* – modeluje vnitřní fyzickou stránku systému
   - Statické prvky implementace (knihovny, soubory, obrázky, . . .)
   - Ale i „aktivní“ – server, databáze
   - Podobné diagramu tříd, zaměřeno na statické prvky

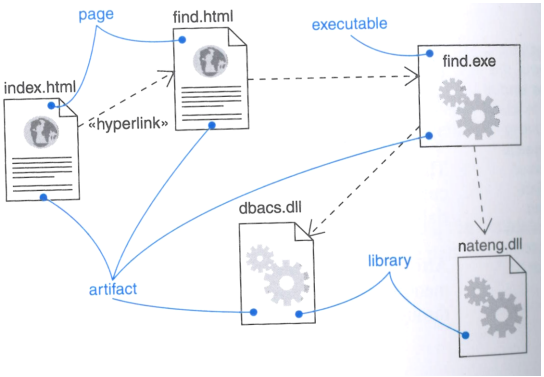


Diagram artefaktů

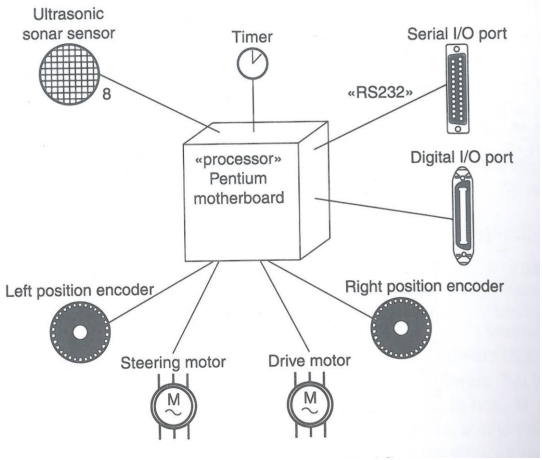


Diagram nasazení

6. *Diagram nasazení* – modeluje fyzickou stránku nasazeného systému
   - Často obsahuje *custom* objekty, které graficky znázorňují konkrétní části
   - Velmi často opomíjená část dokumentace – *kde, co a jak běží*
   - Modeluje embedded, client/server a distribuované systémy

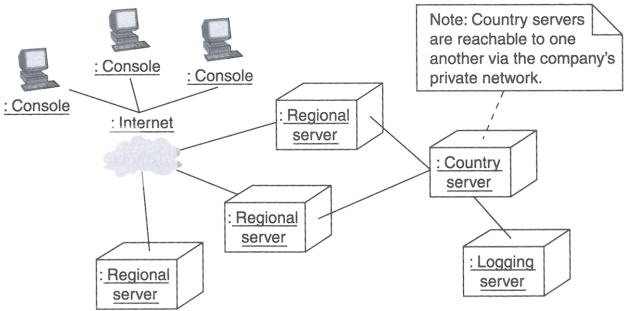
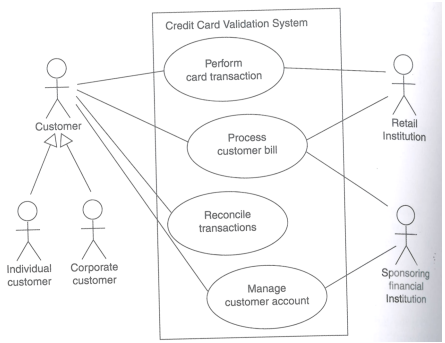


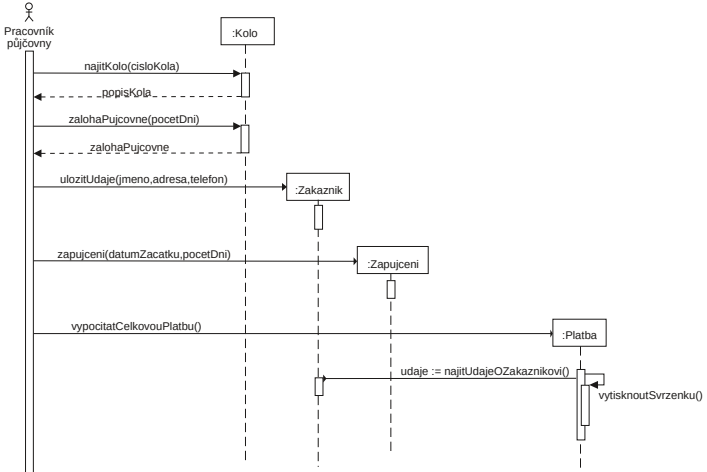
Diagram případů užití

1. *Diagram případů užití* – modeluje kdo(aktér) a jak(případ užití) může systém používat
   - Zobrazuje funkční požadavky a vztahy mezi nimi (rozšíření funkcionality)
   - Hojně používán při tvorbě specifikace systému
   - Vnější pohled na prvky subsystém – popis co mohou dělat



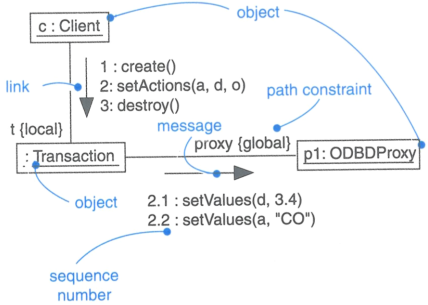
Interakční diagramy – sekvenční diagram

2. *Sekvenční diagram* – zobrazuje spolupráci objektů (zasílání zpráv) v čase
   - Zobrazení chování objektů při určitém scénáři
   - Čas roste shora dolů



Interakční diagramy – komunikační diagram

3. *Komunikační diagram* – zobrazuje strukturu objektů při zasílání zpráv
   - Zobrazuje dynamický pohled na role, propojení a zprávy zasílané instancemi v dané roli
   - Sémanticky ekvivalentní s předchozím, chybí časová osa ⇒ číslování zpráv



Stavový diagram

4. *Stavový diagram* – zobrazuje stavový stroj zobrazovaného objektu
   - Přechody, události, aktivity – dynamický pohled na systém
   - Jak se mění stav objektu při konkrétních událostech

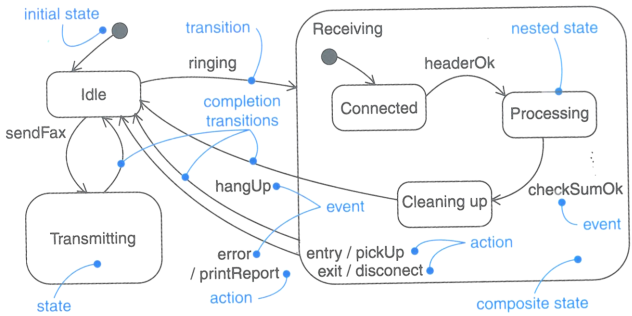


Diagram aktivit

5. *Diagram aktivit* – modelování složitějších (dynamických) případů užití (větvení)
   - Podmínky a konkrétní hodnoty při vstupu a výstupu z akci
   - Zobrazení „toku dat“ z *akce* do *akce*

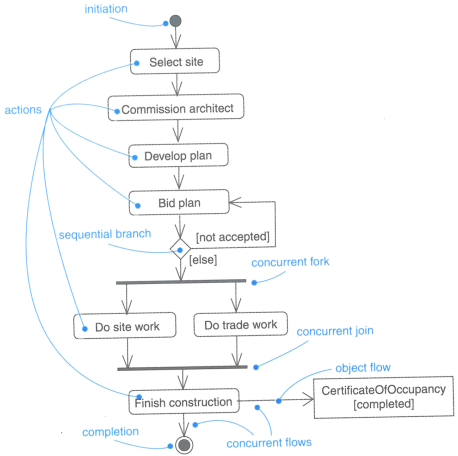
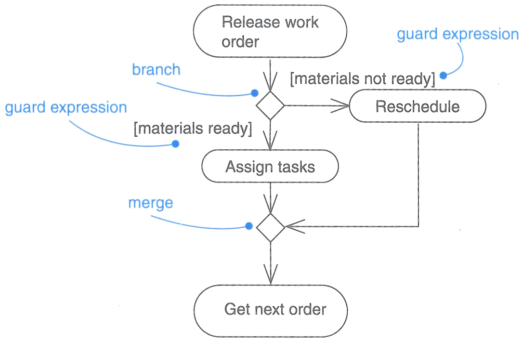


Diagram aktivit



Doporučená literatura

- Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language User Guide (2th edition). Addison Wesley, 2005. ISBN: 321267974
- Joseph Ingeno. Software Architect's Handbook. Packt, 2018. ISBN: 978-1-78862-406-0
  - Chapter 12 – Documenting and reviewing software architecture
- Dan Pilome, Neil Pitman. UML 2.0 In a nutshell O'Reilly. 2005. ISBN: 978-0-596-00795-9