(Rational) Unified Process

© 2008 Radek Ošlejšek
FI MU Brno

oslejsek@fi.muni.cz
http://www.fi.muni.cz/~oslejsek/PA103

Vývojové procesy

UML je pouze sada modelovacích nástrojů (syntaxe + sémantika), na jejichž správné použití potřebujeme vědět:

- **co** modelovat v **analytickém modelu**?
- **co** modelovat v **modelu architektury** systému?
- **co** modelovat v **modelu detailního návrhu**?
- **jak** vzájemně souvisí tyto modely (upřesňování, rozsah, mapování)?
- **kdy a v jakém pořadí** vytvářet tyto modely?
- **které techniky/diagramy** použít v které etapě?

WHAT - WHY - WHO - WHEN - HOW

Osvědčené principy z praxe

- Praxe přinesla šest principů, které se vyplatí dodržovat:
  - Iterativní vývoj
  - Správa požadavků
  - Používání komponentové architektury
  - Grafické modelování (UML)
  - Kontrola kvality
  - Řízení změn

Iterativní vývoj

- Iniciální návrh většinou nesplňuje všechny klíčové požadavky
- Odhalení chyb návrhu až v pozdějších etapách vývoje má za následek prodražení projektu a/nebo jeho selhání.
- **Čas a peníze věnované na implementaci chybného návrhu jsou nenávratné**
- Výhody:
  - Snadnější řízení rizik (rizika jsou rozpoznána včas)
  - Iniciální iterace umožňují včasnou zpětnou vazbu od uživatelů
  - Testování a integrace jsou průběžné
  - Zaměření na krátkodobější cíle v rámci jedné iterace
  - Vývoj je lépe měřitelný
  - Neúplné implementace mohou být postupně doplňovány

Správa požadavků

- Vyhodnocení, organizace a dokumentace požadované funkcionality a omezení.
- Sledování změn a dokumentace kompromisů a rozhodnutí
- Požadavky firemních procesů (business requirements) jsou snadno zachytitelné pomocí případů užití.
- Případy užití jsou důležité nástroje pro plánování.
- Případy užití řídí celý proces vývoje, od analýzy až po testování.

Použití komponentové architektury

- Navrhujte, implementujte a testujte svoji architekturu předem.
- Systematická cesta jak vytvořit „dobrou“ architekturu
  - odolnost proti změnám díky dobře definovaným rozhraním
  - (znovu)použití komponent
  - odvození z případů užití

Grafické modelovaní

- Zachycení struktury a chování architektur a komponent
- Zobrazení, jak k sobě elementy „pasují“ (konzistence modelů)
- Schování/ukázání detailů podle potřeby
- Udržování konzistence mezi návrhem a implementací
- Podpora jednoznačné komunikace

- **Grafické modelování zvyšuje schopnost uřídit složitost**

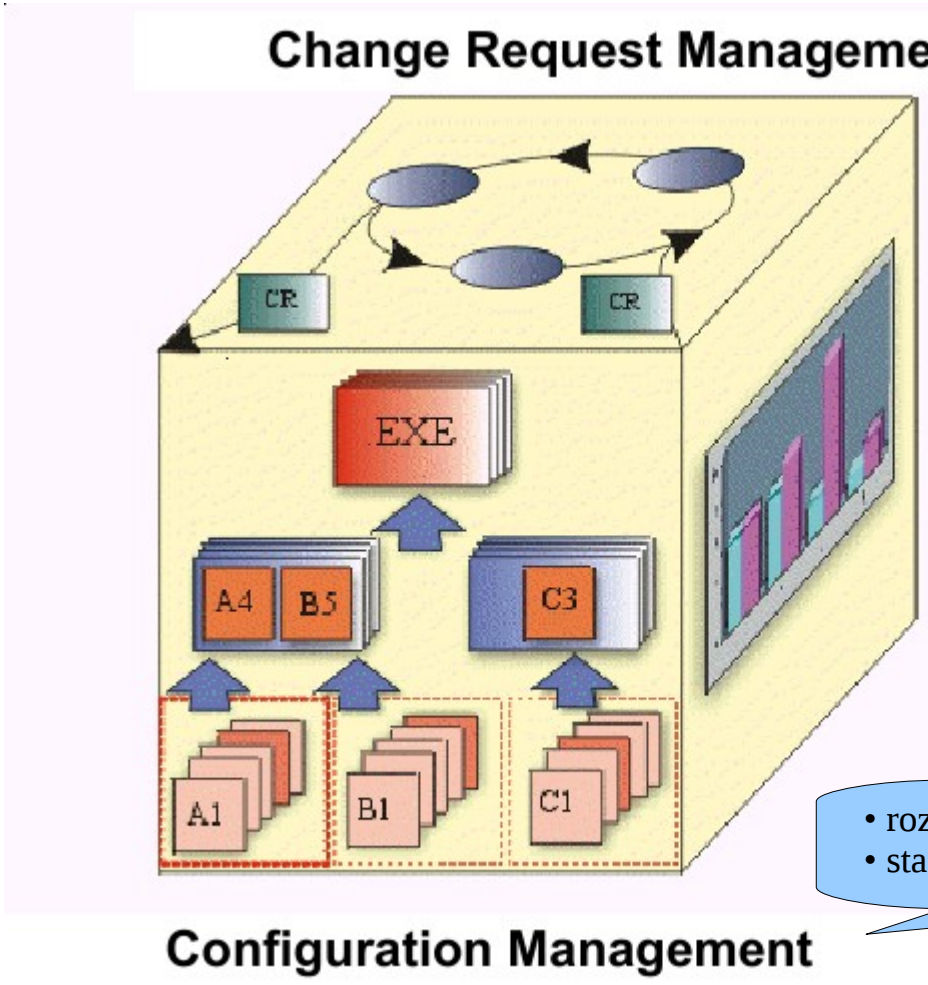
=> UML

Kontrola kvality

- Vytváření testů pro každý primární scénář, abychom si byli jisti, že všechny požadavky jsou správně implementovány
- Kontrola spolehlivosti SW – memory leaks, úzká místa, ...
- Nepřijatelný výkon aplikace je stejný problém jako chybějící funkcionalita nebo nepřijatelná spolehlivost
- Testování každé iterace – automatizované testy!

- **Je 100 až 1000-krát dražší najít a opravit chybu v softwaru až po jeho uvedení do provozu**

Řízení změn



• zachycení požadavků na změny od různých zdrojů
(nové požadavky od uživatelů,
opravy chyb od testerů, ...)

Measurement

• počet uzavřených změn za den
• počet nových změn za den

• rozdělení do subsystémů pro jednotlivé týmy
• stanovení „secure workspaces“

(Rational) Unified Process

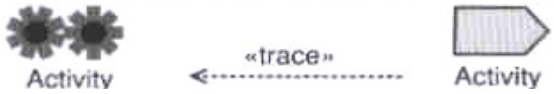
- Unified Process (UP)
  - Jacobson, Booch, Rumbaugh
  - otevřený, obecný koncept
- Rational Unified Process (RUP)
  - komerční implementace UP od IBM
  - rozšiřuje UP
  - liší se v detailech
  - UP i RUP mají mnohem více společného než rozdílného
- Proces (R)UP považujte za **kostru,** do které přidáváte a ze které odvozujete projektově specifickou, přizpůsobenou a podrobnou definici procesu

(R)UP: worker

- **Role, pracovník** *(worker)*
  - modeluje „kdo“ v procesu vývoje softwaru
  - role v projektu pro jednotlivce nebo tým
  - každý *worker* může být realizovaný několika jednotlivci nebo týmy a každý jednotlivec nebo tým může vystupovat v několika *workers.*
  - v RUPu se používá termín *role* namísto *worker*, význam je ale stejný

(R)UP: activity

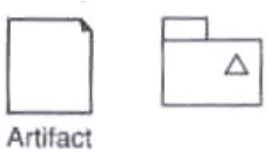
- **Aktivita** *(activity)*



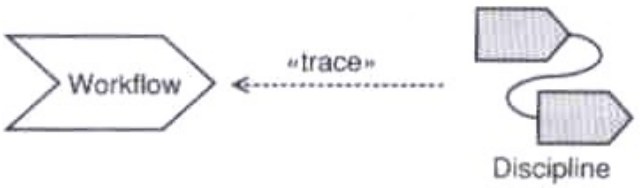
  - modeluje „co“ v procesu vývoje softwaru
  - úkol v projektu vykonávaný jednotlivci nebo týmy
  - jednotlivci a týmy musí při provádění aktivity vždy přijmout konkrétní roli; UP i RUP proto s aktivitami asociuje role (workers)
  - aktivity mohou být dekomponovány na podrobnější úrovně

(R)UP: artifact

- **Artefakt** *(artifact)*
  - modeluje „co“ v procesu vývoje softwaru
  - vstupy a výstupy projektu
    - zdrojový kód, spustitelný program, standardy, dokumentace, ...
  - mohou být znázorněny různými ikonami v závislosti na tom, co představují

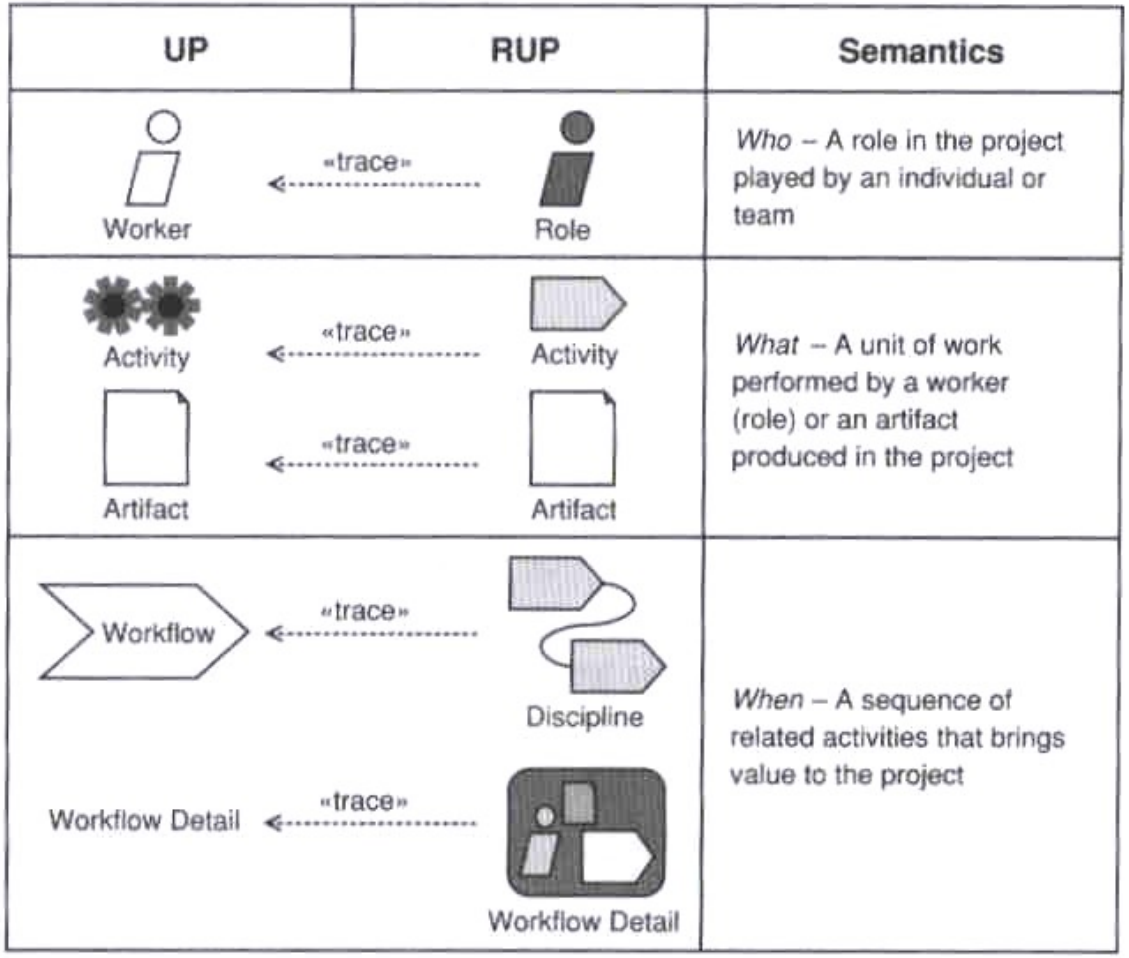


(R)UP: workflow

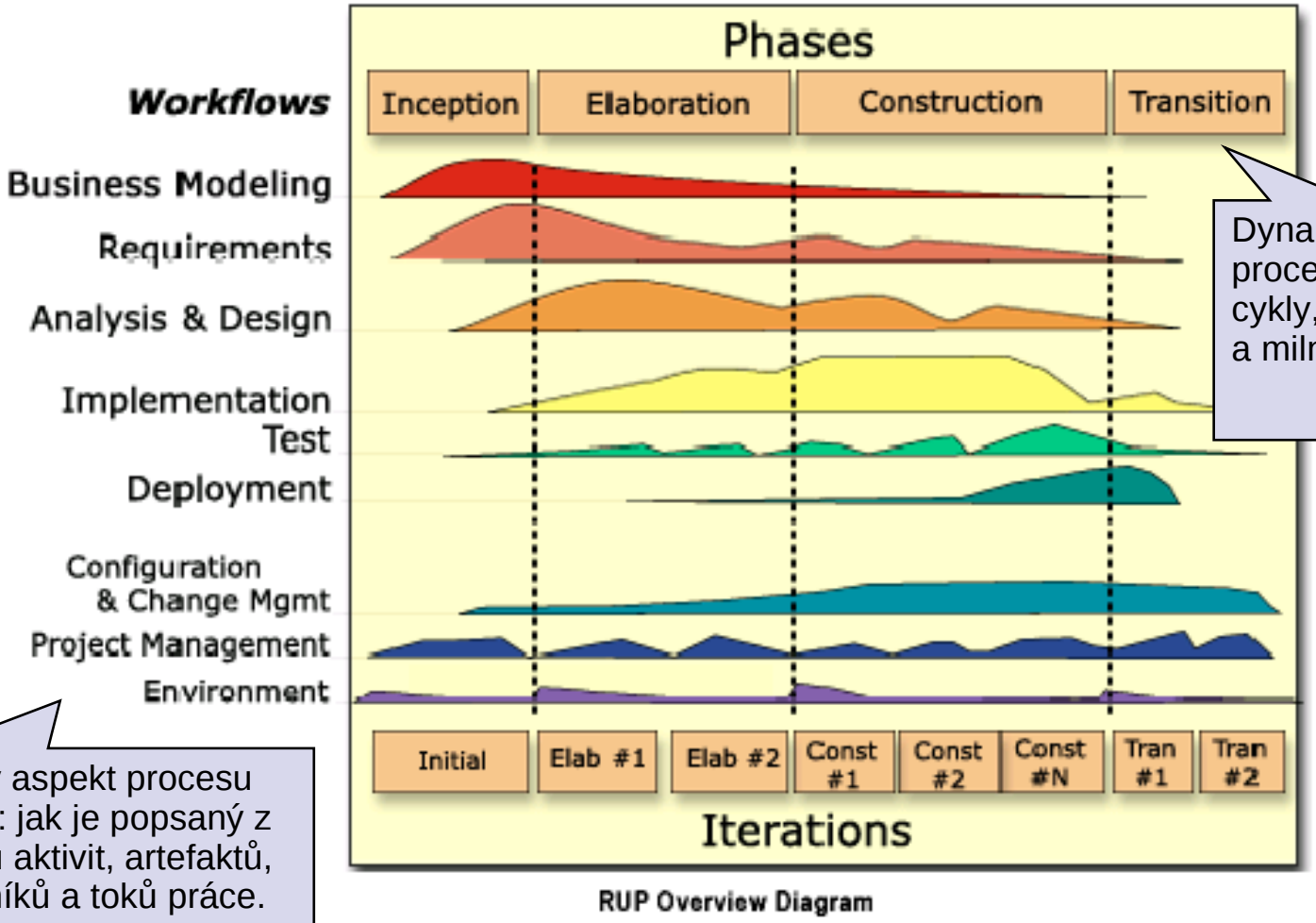


- **Tok práce** *(workflow)*
  - modeluje „kdy“ v procesu vývoje softwaru
  - sekvence aktivit vykonávaných pracovníky (workers)
  - mohou být dekomponovány na jeden a více detailnějších modelů
  - detailnější toky práce jsou pouze odkazovány ikonou

Notace UP a RUP



RUP: Dvě dimenze procesu vývoje



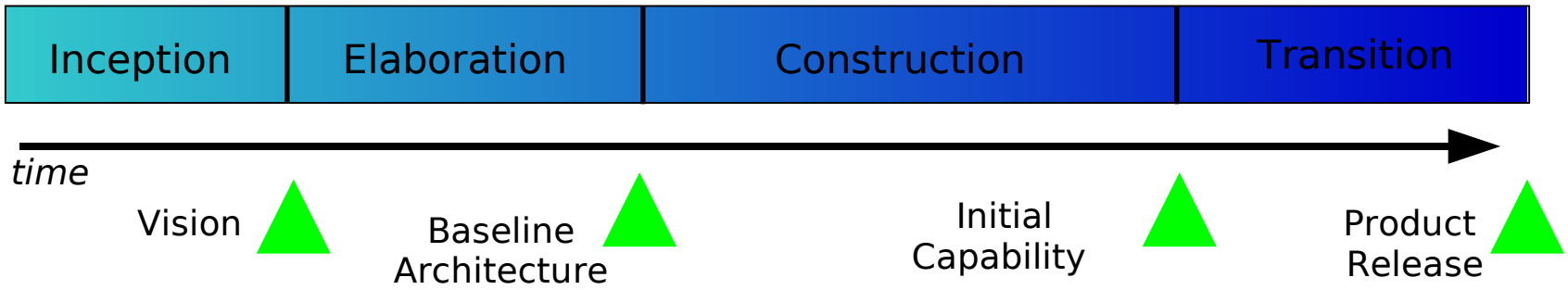
RUP Overview Diagram

Dynamický aspekt procesu (čas): cykly, fáze, iterace a milníky.

Statický aspekt procesu (obsah): jak je popsaný z pohledu aktivit, artefaktů, pracovníků a toků práce.

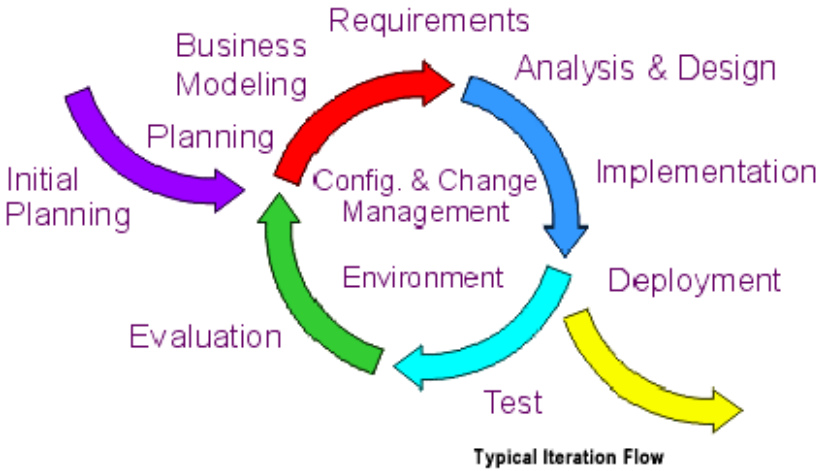
RUP: Cykly a fáze

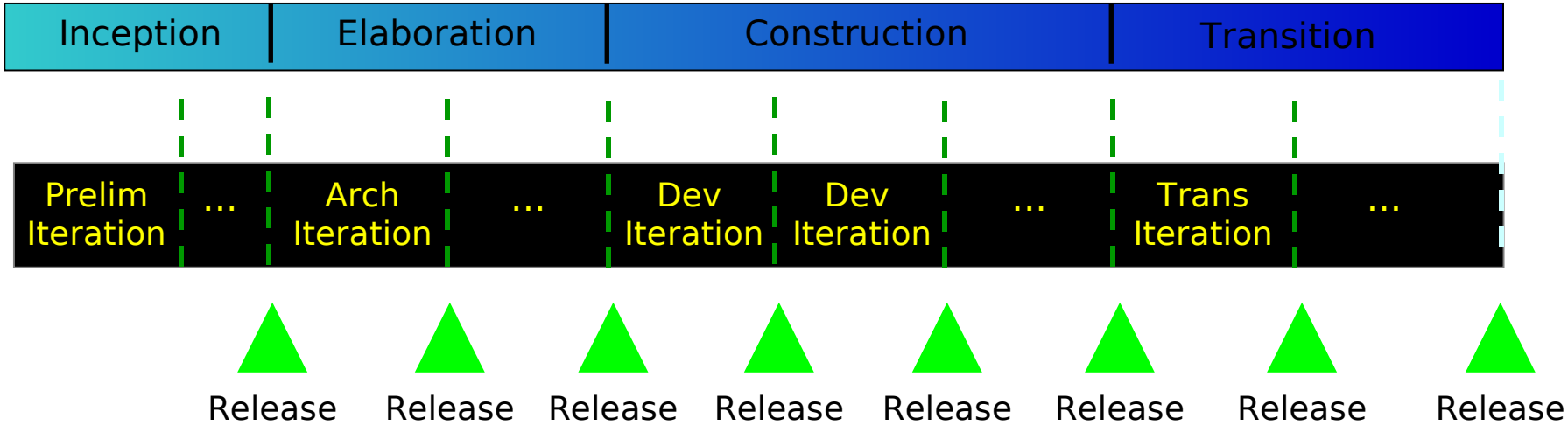
- Výrobek prochází různými **cykly** (= inkrementy v inkrementálním vývoji)
  - každý cyklus produkuje novou verzi systému (release)
- Každý cyklus má následující **fáze**:
  - **Zahájení (Inception)**: potvrzení konceptu, proveditelnost, cíle
  - **Rozpracování (Elaboration)**: detailní požadavky, scénáře použití, architektura, komponenty, vzory interakce na vysoké úrovni
  - **Konstrukce (Construction)**: zdokonalení architektury, implementační iterace řízené rizikem
  - **Předání (Transition)**: uživatelské přijetí, ...



Iterace

- Každá fáze může být dále rozdělena do různých **iterací.**
- Každá iterace obsahuje **analýzu, návrh** a **implementační** aktivity.
- Každá iterace produkuje spustitelnou verzi (release), buďto interní nebo externí





Fáze zahájení (Inception Phase)

- **Cíl**
  - ustanovení obchodního případu u nového systému
  - specifikace rozsahu
- **Výstup**
  - obecný přehled o požadavcích na projekt, tj. klíčové požadavky
    - iniciální model případů užití a iniciální model domény (hotovo z 10-20%)
    - popis případů užití a aktérů jednou větou
  - iniciální obchodní případ, včetně:
    - kritérií úspěšnosti
    - iniciální posouzení rizik
    - odhad potřebných zdrojů
- **Milník**
  - definice cílů životního cyklu

Fáze rozpracování (Elaboration Phase)

- **Cíl**
  - analyzovat problémovou oblast
  - ustanovit základy architektury
  - určit nejrizikovější části projektu
  - vytvořit kompletní plán pro dokončení projektu
- **Výstup**
  - model případů užití a model domény z 80% hotové
    - seznam aktérů, dokumentace případů užití
  - realizovatelná architektura a její průvodní dokumentace
  - aktualizovaný obchodní případ, vč. aktualizace posouzení rizik
  - plán vývoje celého projektu
    - rozdělený do iterací, odhad ceny pro jednotlivé iterace
- **Milník**
  - architektura životního cyklu

Fáze konstrukce (Construction Phase)

- **Cíl**
  - inkrementálně vyvíjet a dokončit softwarový produkt, který bude připravený k předání zákazníkovi
- **Výstup**
  - kompletní model případů užití a návrhový model tříd
  - spustitelné verze s novými funkcemi
  - uživatelská dokumentace
  - dokumentace nasazení (administrace)
  - hodnotící kritéria pro každou iteraci
  - popisy verzí
  - aktualizovaný vývojový plán
- **Milník**
  - iniciální provozuschopný systém

Plánování konstrukčních iterací

Co se dělá během 1.iterace ?

- základní funkcionalita
  - buď implementovaná (pokud je vysoce riziková) nebo jako protézy (v případě nízkého rizika)
- případy užití s nejvyšším rizikem
  - úprava seznamu rizik podle stávající situace (složité problémy, které jsou rozhodující a pro které dosud není známé řešení)
  - stanovení priority rizik a rozhodnutí, které bude řešeno nejdříve
- primární scénáře případů užití
  - prvním cílem je vyřešení vysoce rizikových výzev, plná funkcionalita přijde na řadu později

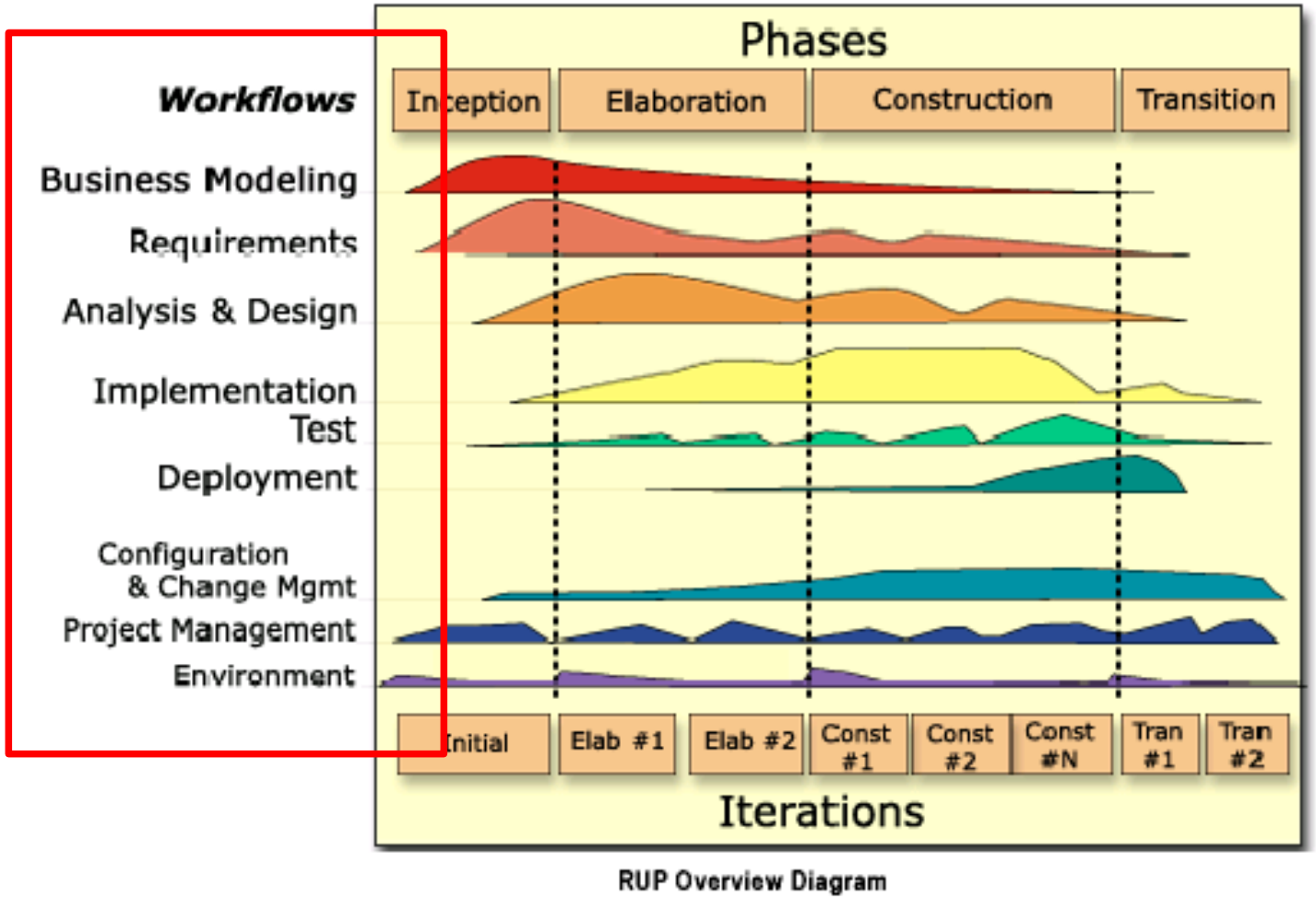
Každá iterace vytváří **proveditelnou, testovatelnou**, a v dalších iteracích dokonce **nasaditelnou verzi systému**.

- protézy, simulátory, přidané testovací komponenty
- testováno jinými lidmi než autory implementací

Fáze předání (Transition Phase)

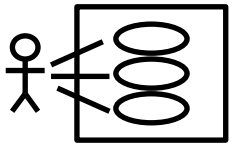
- **Cíl**
  - předat software uživatelům
- **Výstup**
  - spustitelné verze
  - aktualizovaný model systému
  - hodnotící kritéria pro každou iteraci
  - popisy verzí
  - aktualizovaná uživatelská dokumentace
  - aktualizovaná administrátorská dokumentace
  - “post-morten” analýza systému
- **Milník**
  - nová verze (product release)

RUP: Workflows



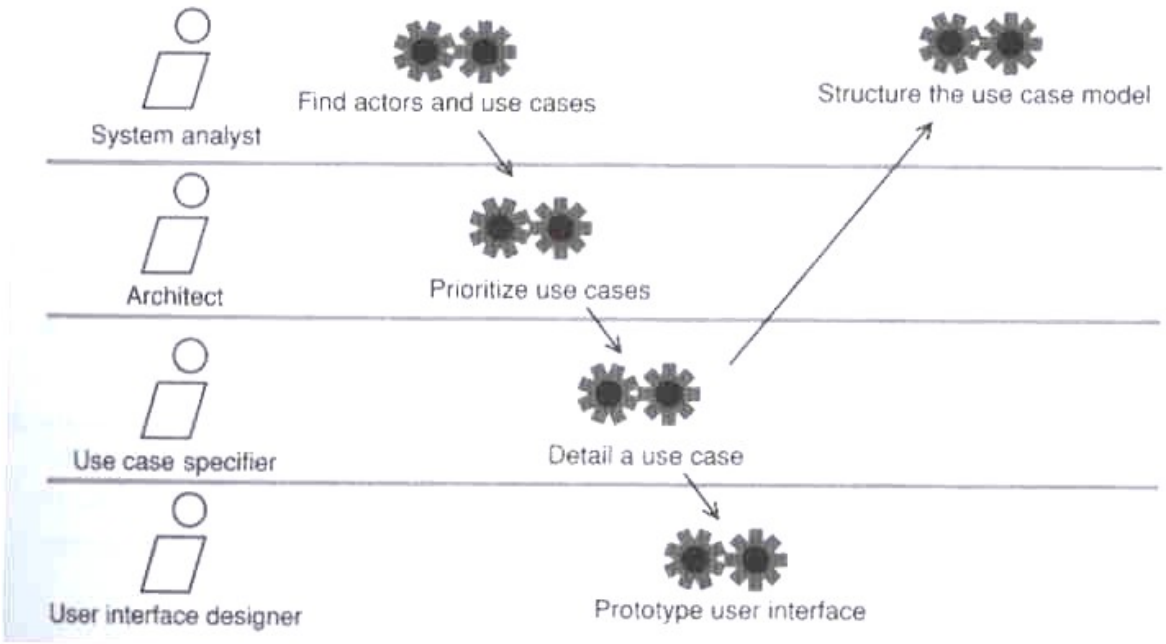
RUP Overview Diagram

Workflow: Requirements (I)



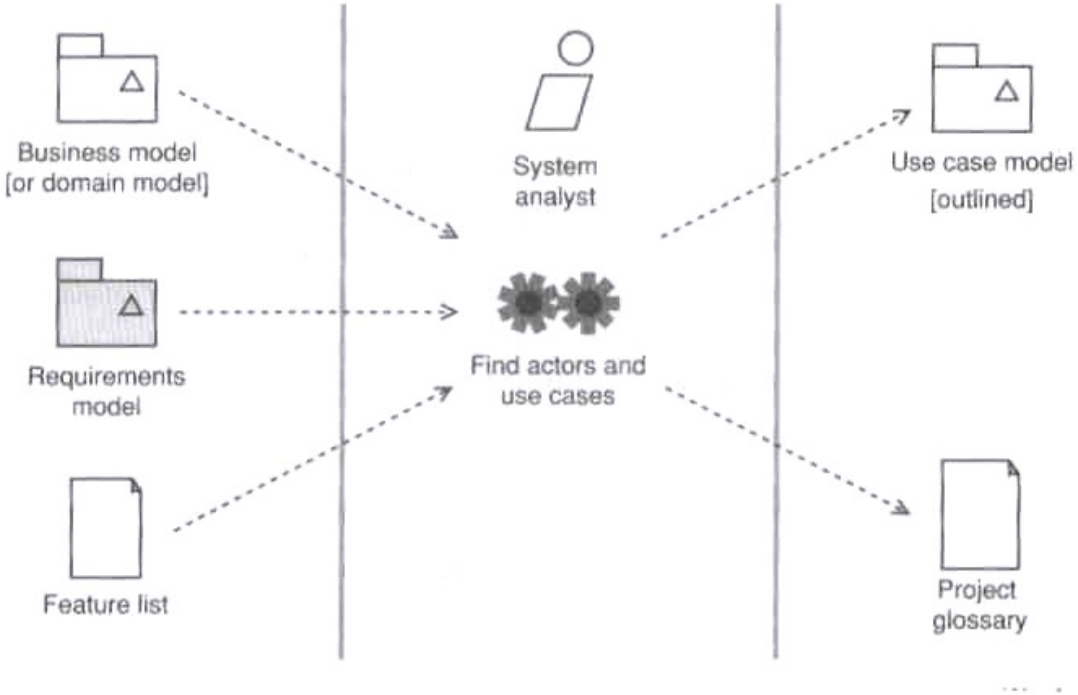
- Zachycení uživatelských požadavků
- Etapy:
  - Modelování případů užití (*use case modeling*)
    - nalezení prvotní hranice systému
    - nalezení aktérů nalezení případů užití
    - základní specifikace případů užití (hlavní toky událostí)
  - Pokročilé modelování případů užití *(advanced use case modeling, detail a use case)*
    - gen/spec aktérů a případů užití
    - *<<extend>>* a *<<include>>*
      - vyhněte se funkční dekompozici!

Workflow: Requirements (II)



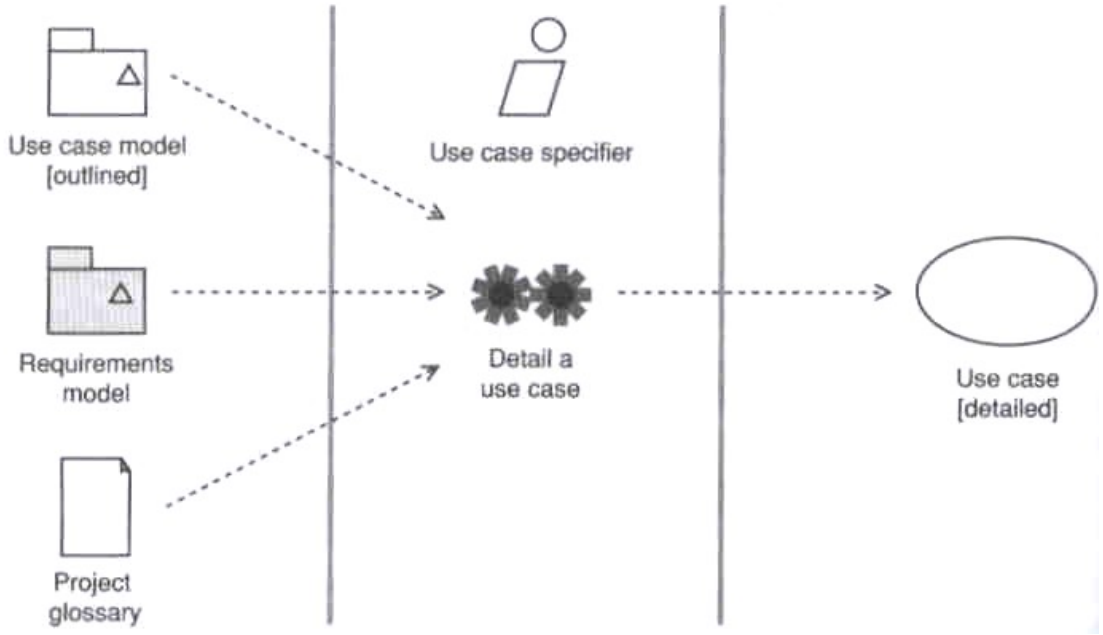
tasks for the UP requirements workflow

Workflow: Requirements (III)



activity *Find actors and use cases*

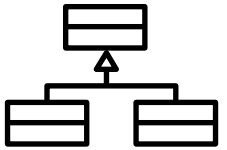
Workflow: Requirements (IV)

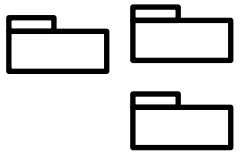


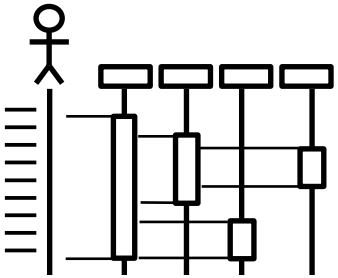
activity *Detail a use case*

Workflow: Analysis

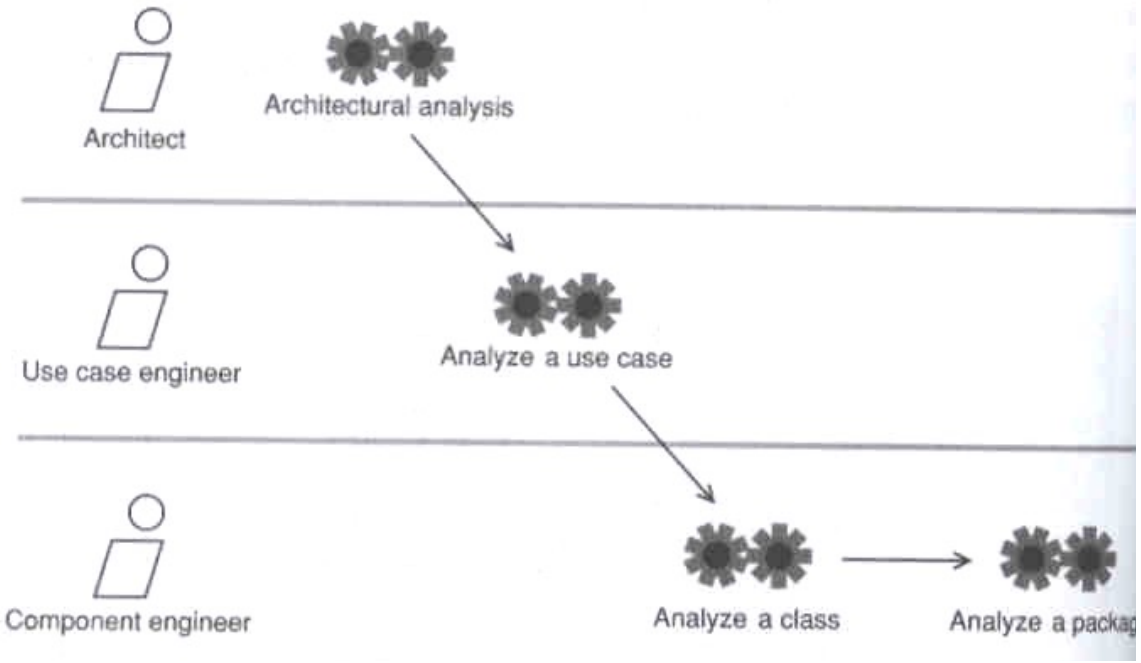
- Model doménové oblasti
- Nalezení základních objektů a tříd
- Nalezení asociací a závislostí
- Dědičnost
- Vytvoření analytických balíčků
- Realizace případů užití
  - modelování p.u. jako spolupráci objektů





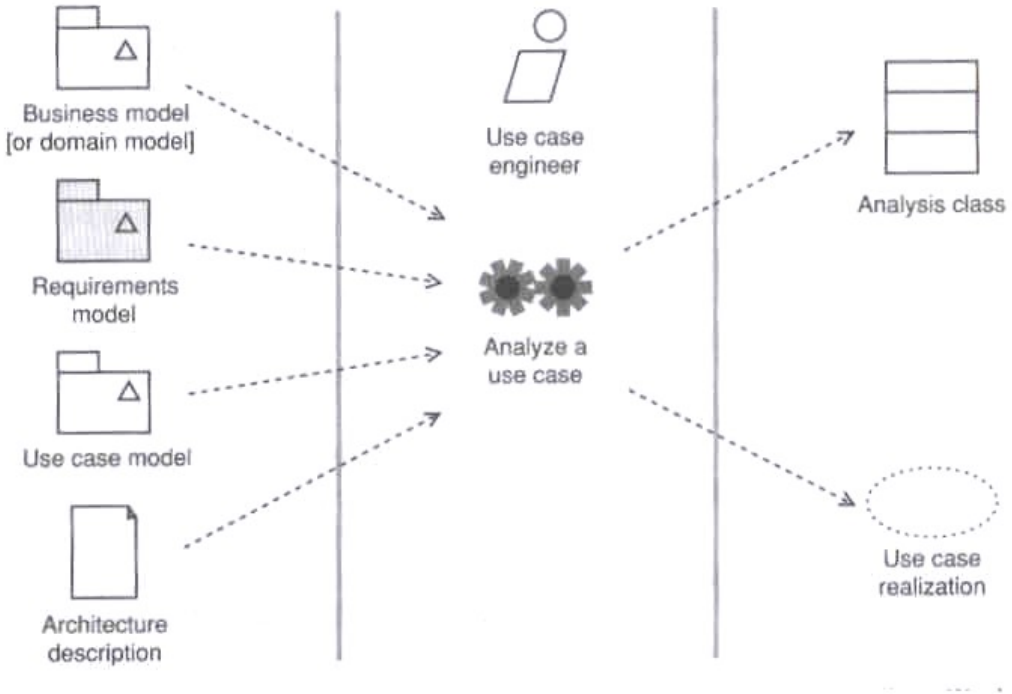


Workflow: Analysis



UP analysis workflow

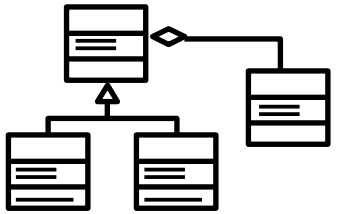
Workflow: Analysis

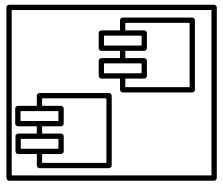


UP activity *Analyze a use case* (classes interaction)

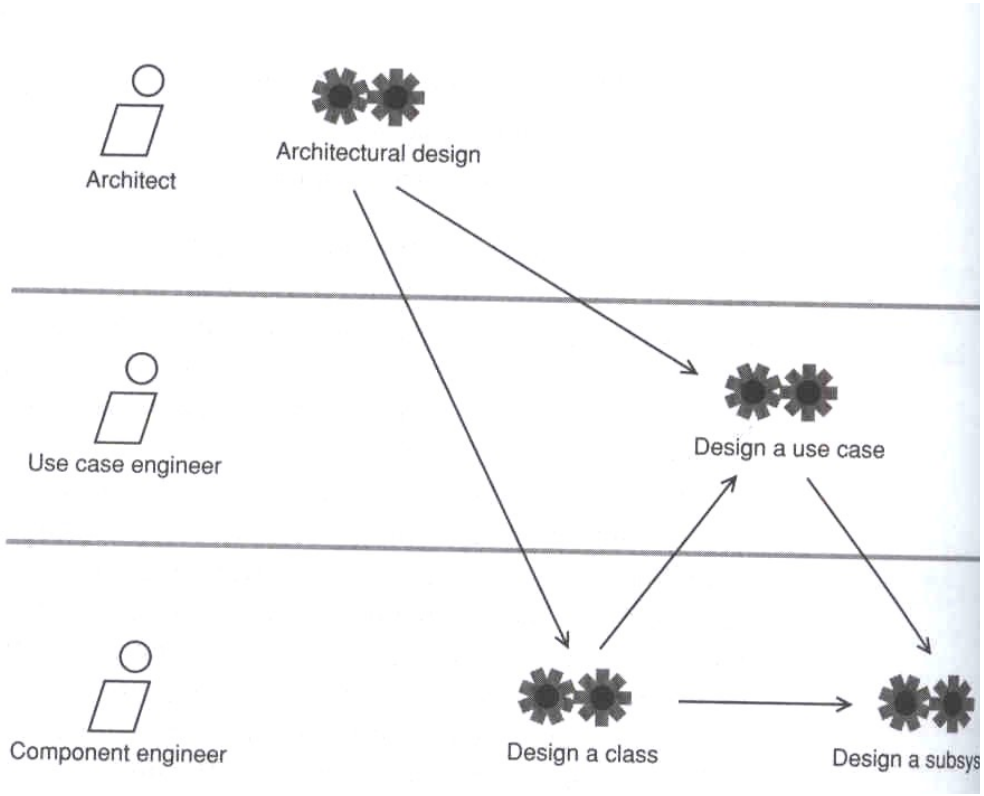
Workflow: Design (I)

- Detailní model
- Vytvoření návrhových tříd
  - všechny potřebné detaily, viditelnost, ...
- Upřesnění analytických vztahů
  - agregace a kompozice
  - implementace M:N asociací
  - asociační třídy
  - směr asociace
- Rozhraní a komponenty
- Návrhové vzory



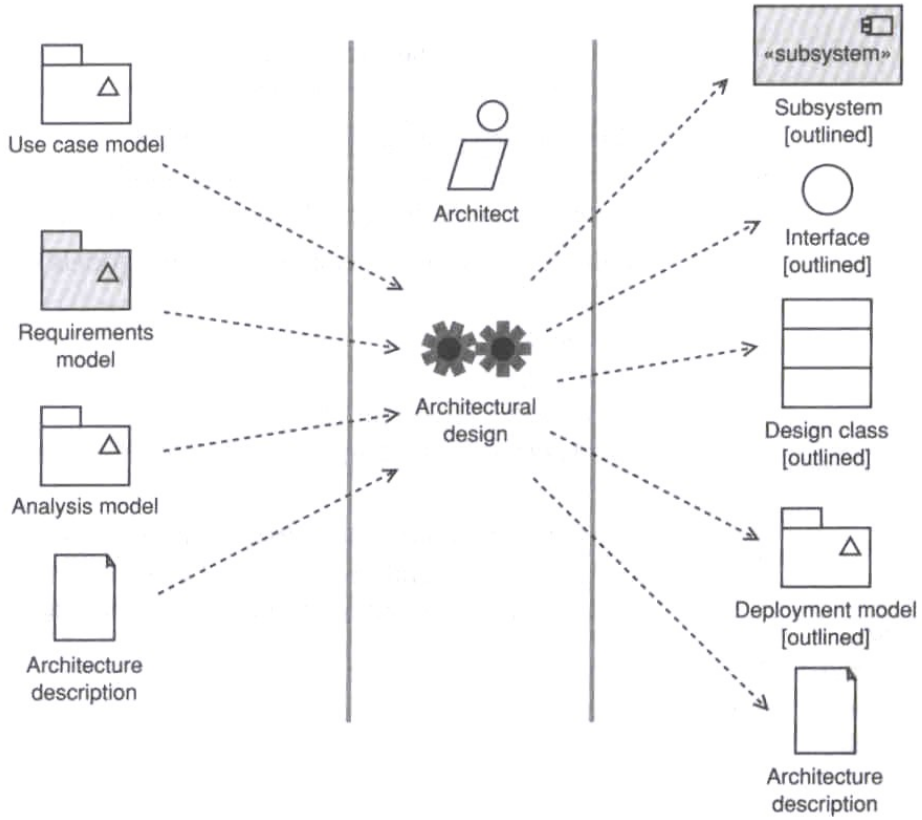


Workflow: Design (II)



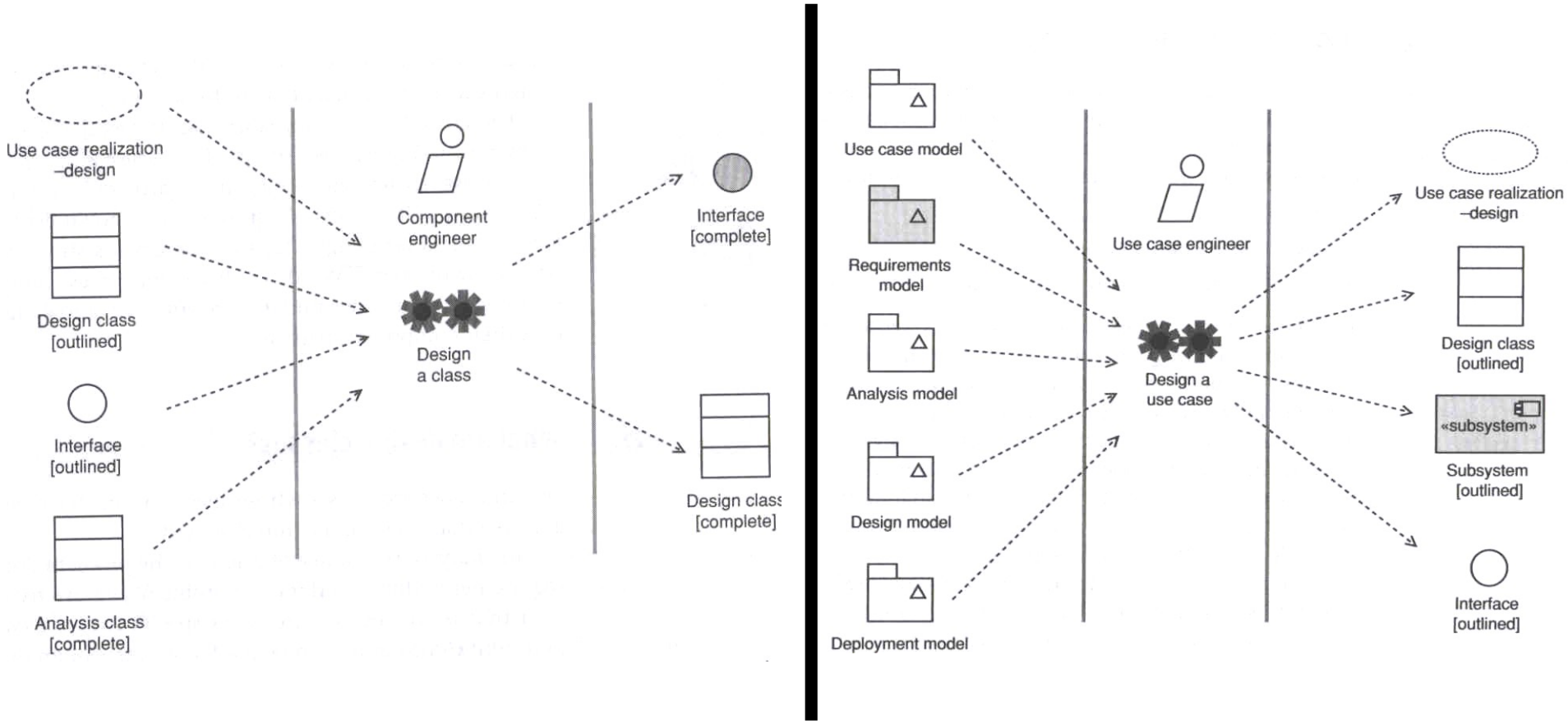
UP workflow for *design*

Workflow: Design (III)



UP activity *Architectural design*

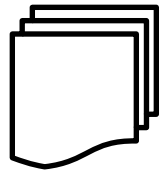
Workflow: Design (IV)



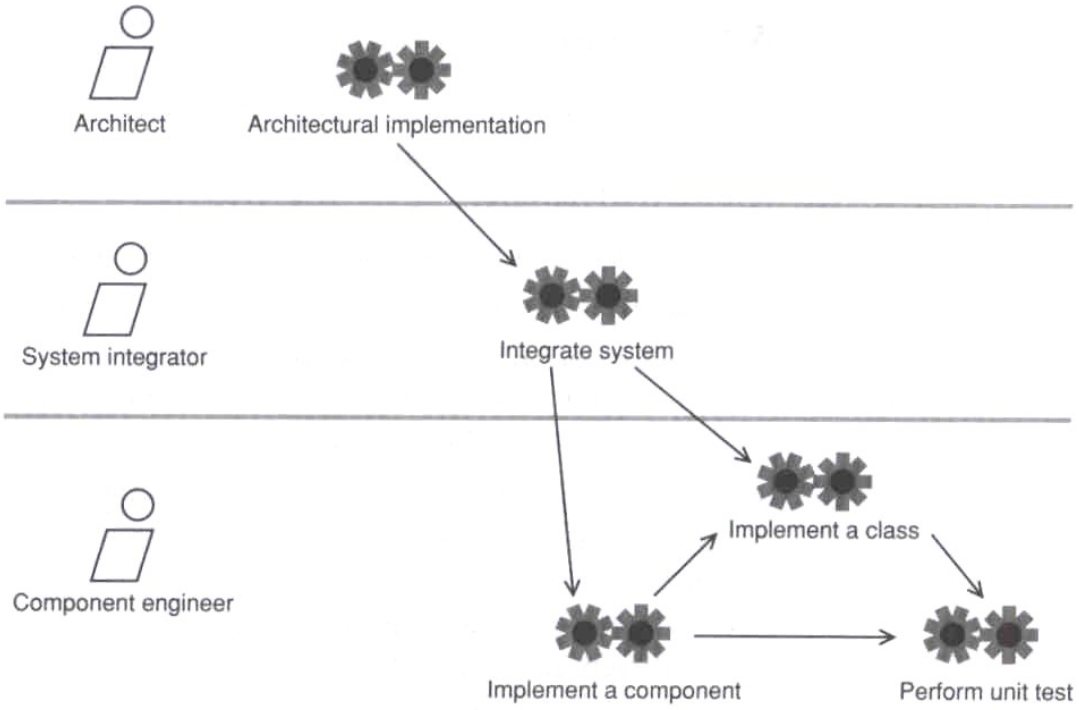
UP activities *Design a class* and *Design a use case* occure concurently and iteratively

Workflow: Implementation (I)

- Rozmístění (deployment)
- Implementace (kódování)
- Testování

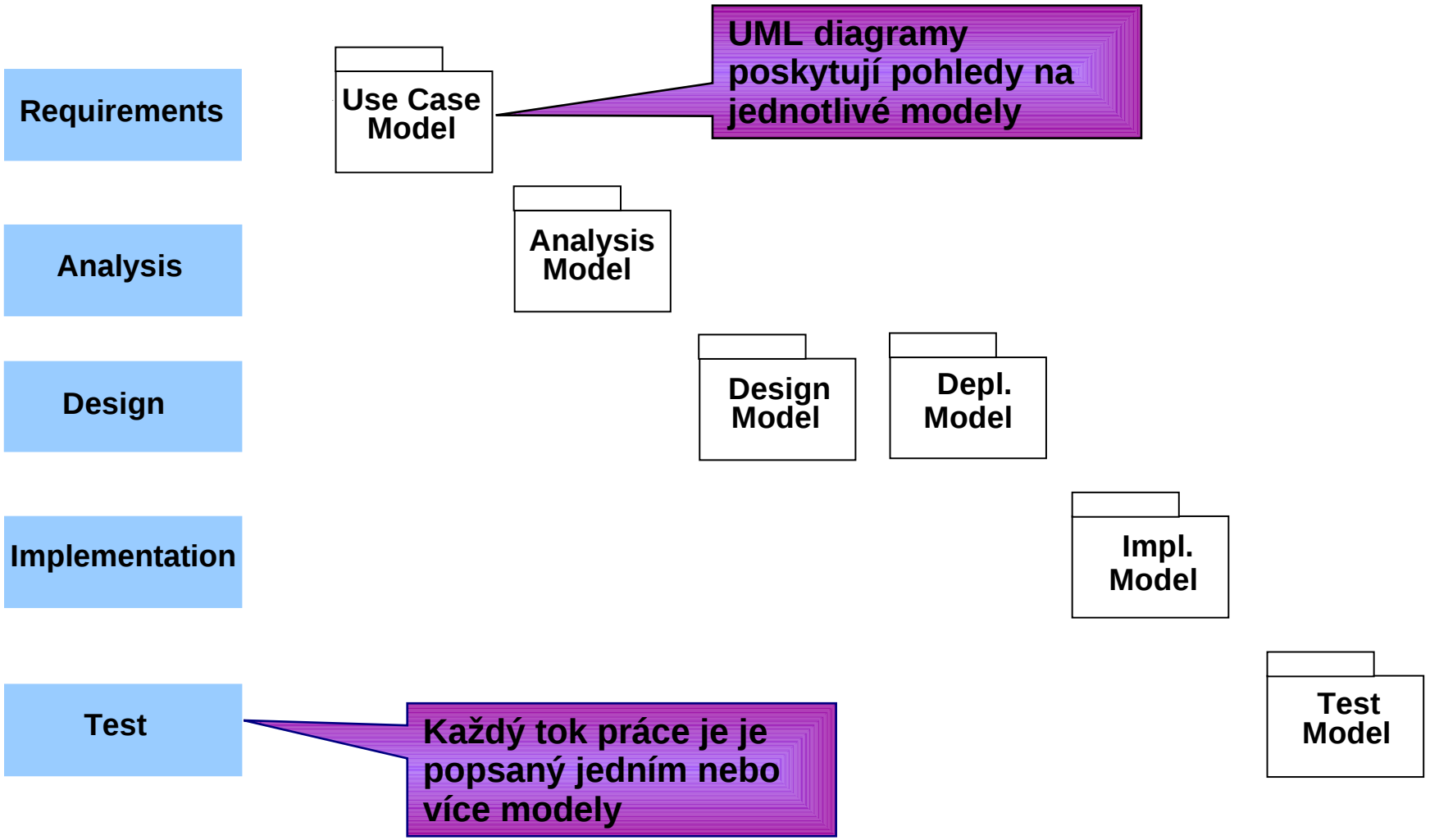


Workflow: Implementation (II)

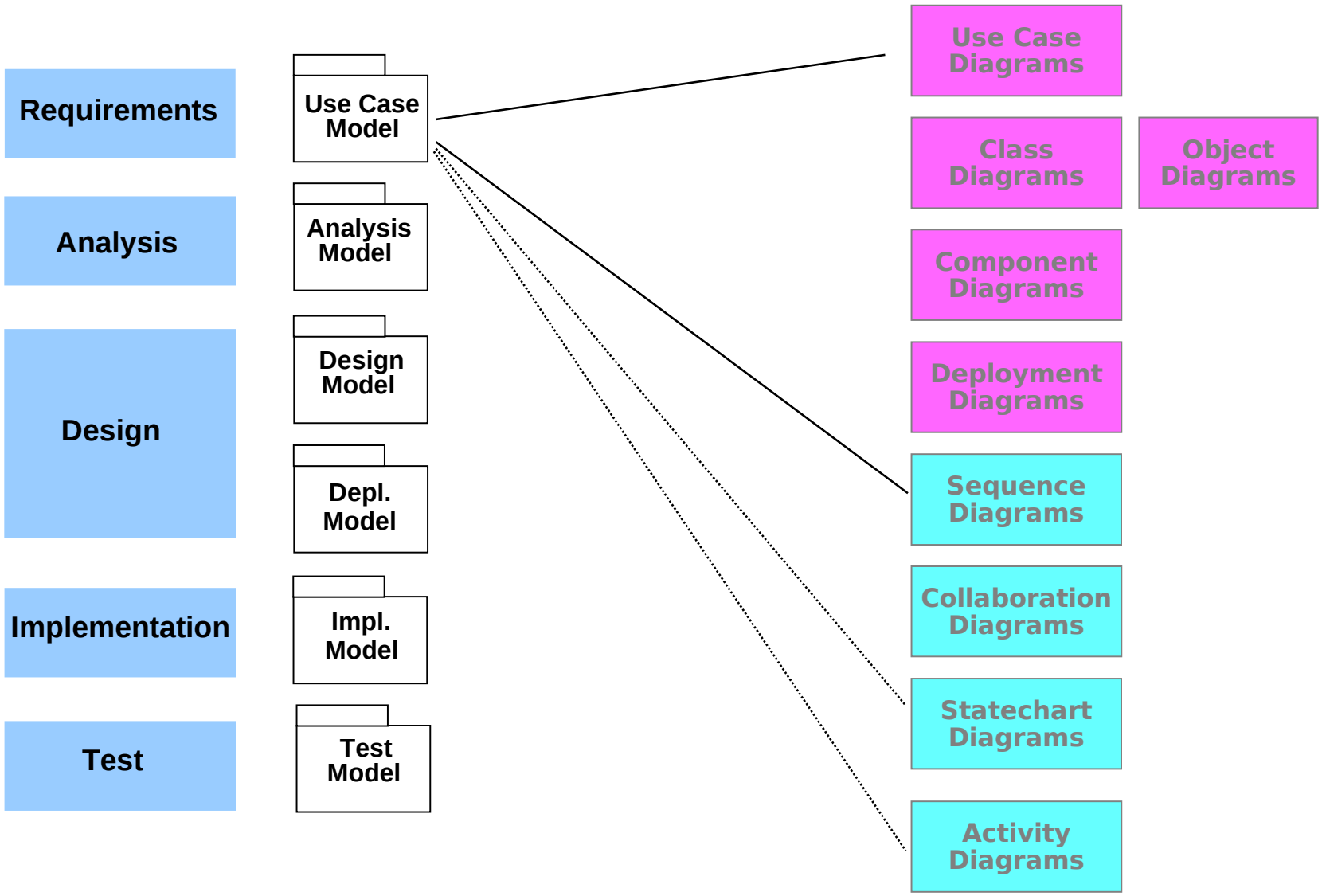


Implementation workflow

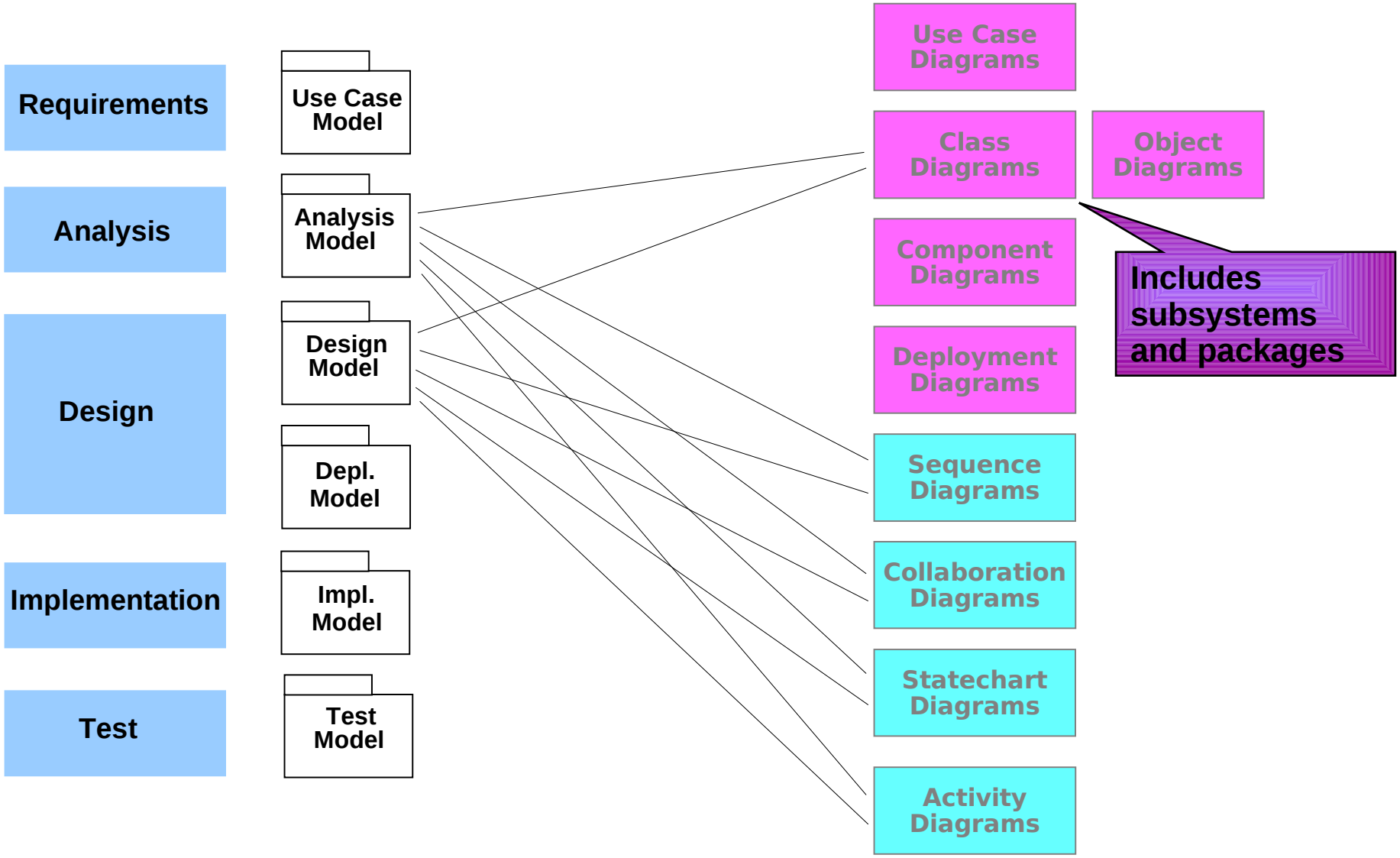
Toky práce a modely



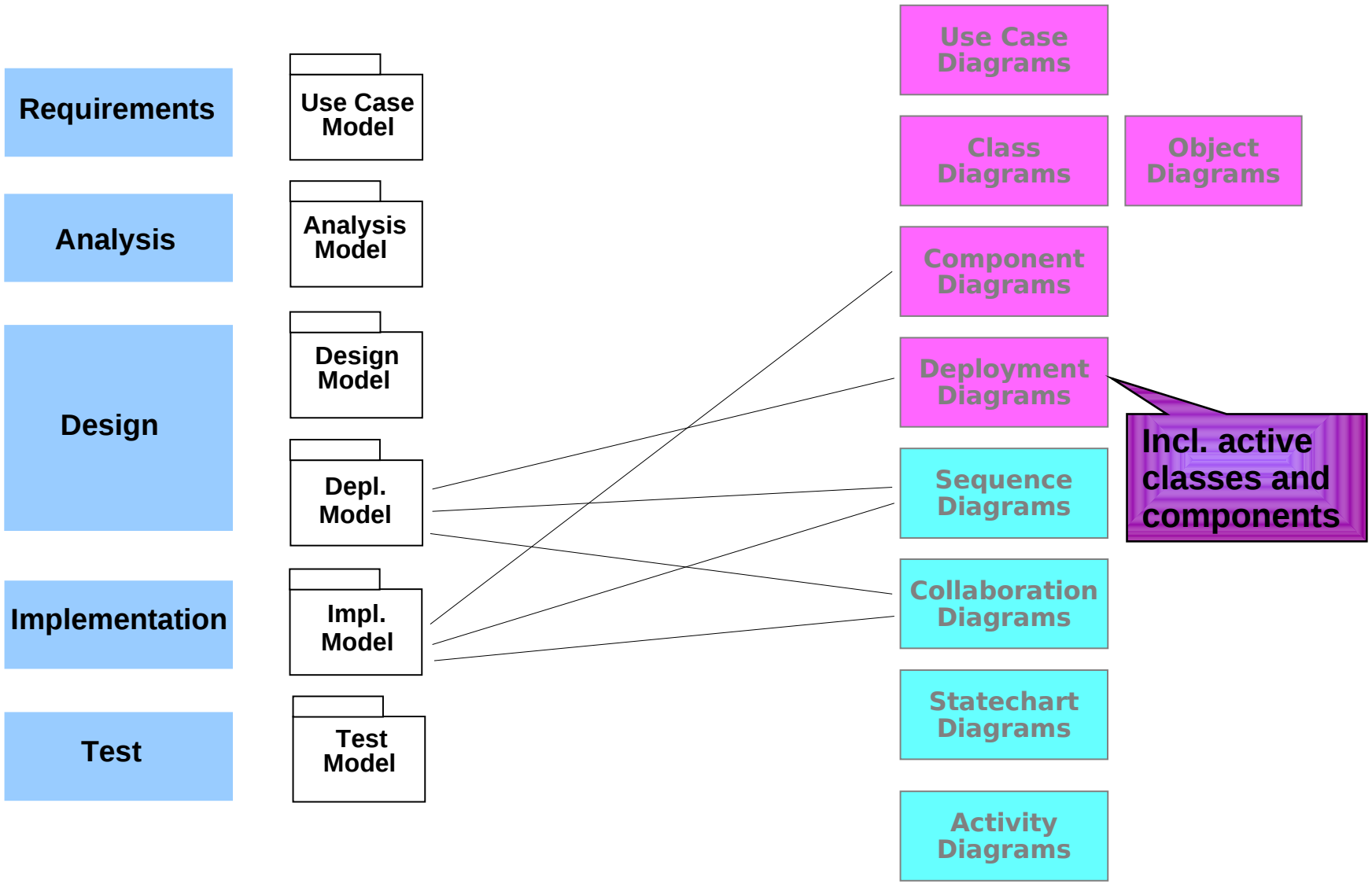
Use Case Model



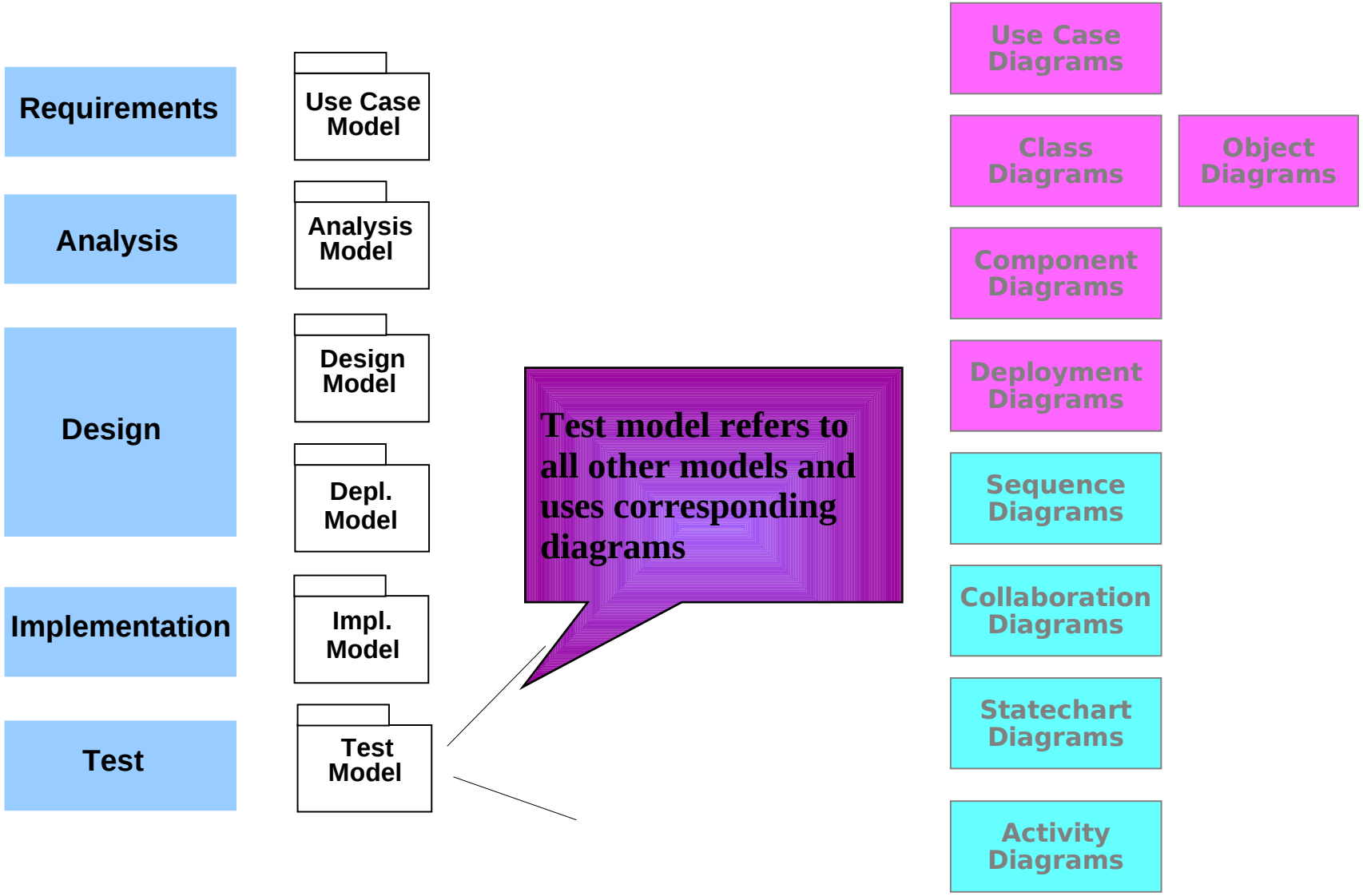
Analysis & Design Model



Deployment and Implementation Model



Test Model


Requirements
Analysis
Design
Implementation
Test
Use Case Model
Analysis Model
Design Model
Depl. Model
Impl. Model
Test Model
Test model refers to all other models and uses corresponding diagrams
Use Case Diagrams
Class Diagrams
Object Diagrams
Component Diagrams
Deployment Diagrams
Sequence Diagrams
Collaboration Diagrams
Statechart Diagrams
Activity Diagrams