Softwarový proces

Radek Mařík

CA CZ, s.r.o.

September 14, 2007

CA Labs

Transforming IT Management

Obsah

Cyklus vývoje softwaru.

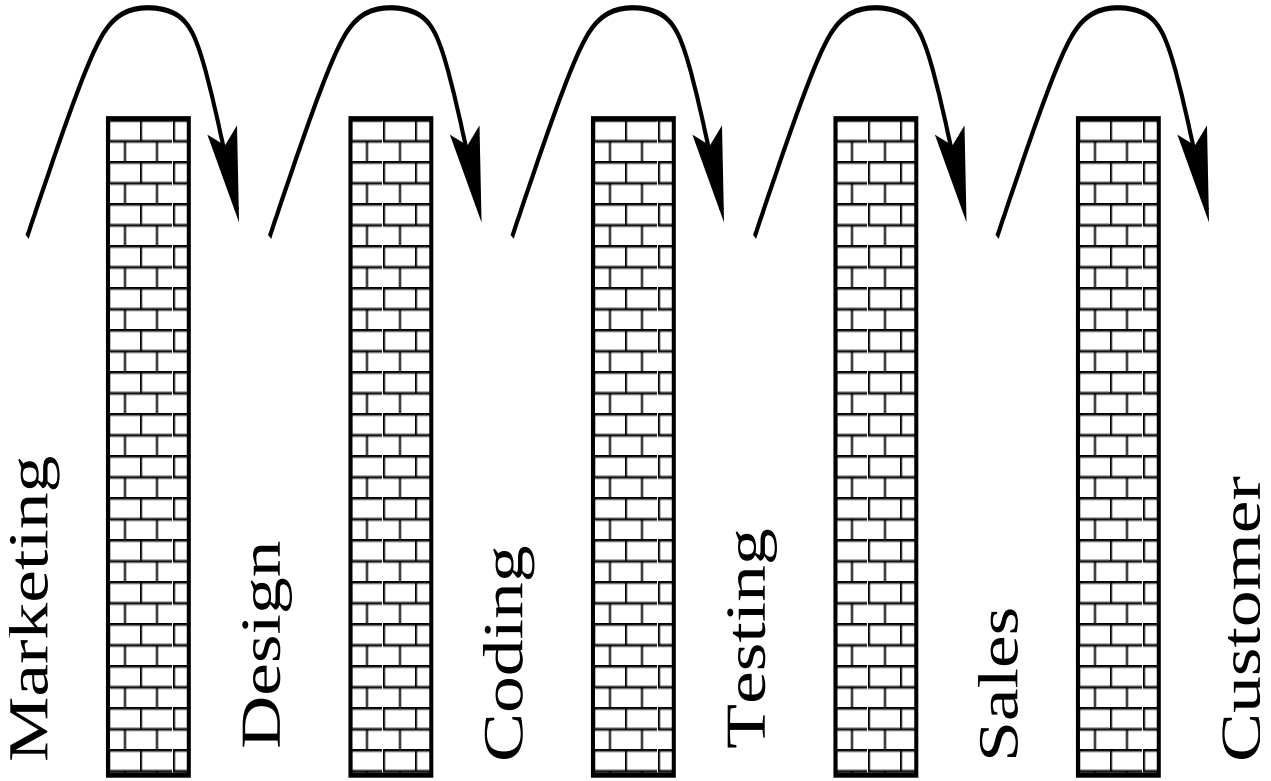
IEEE/ANSI, 1991(Std 1074-1991) Množina aktivit, které vytváří procesy nutné k vývoji a údržbě softwaru.

Typický model vývojového cyklu

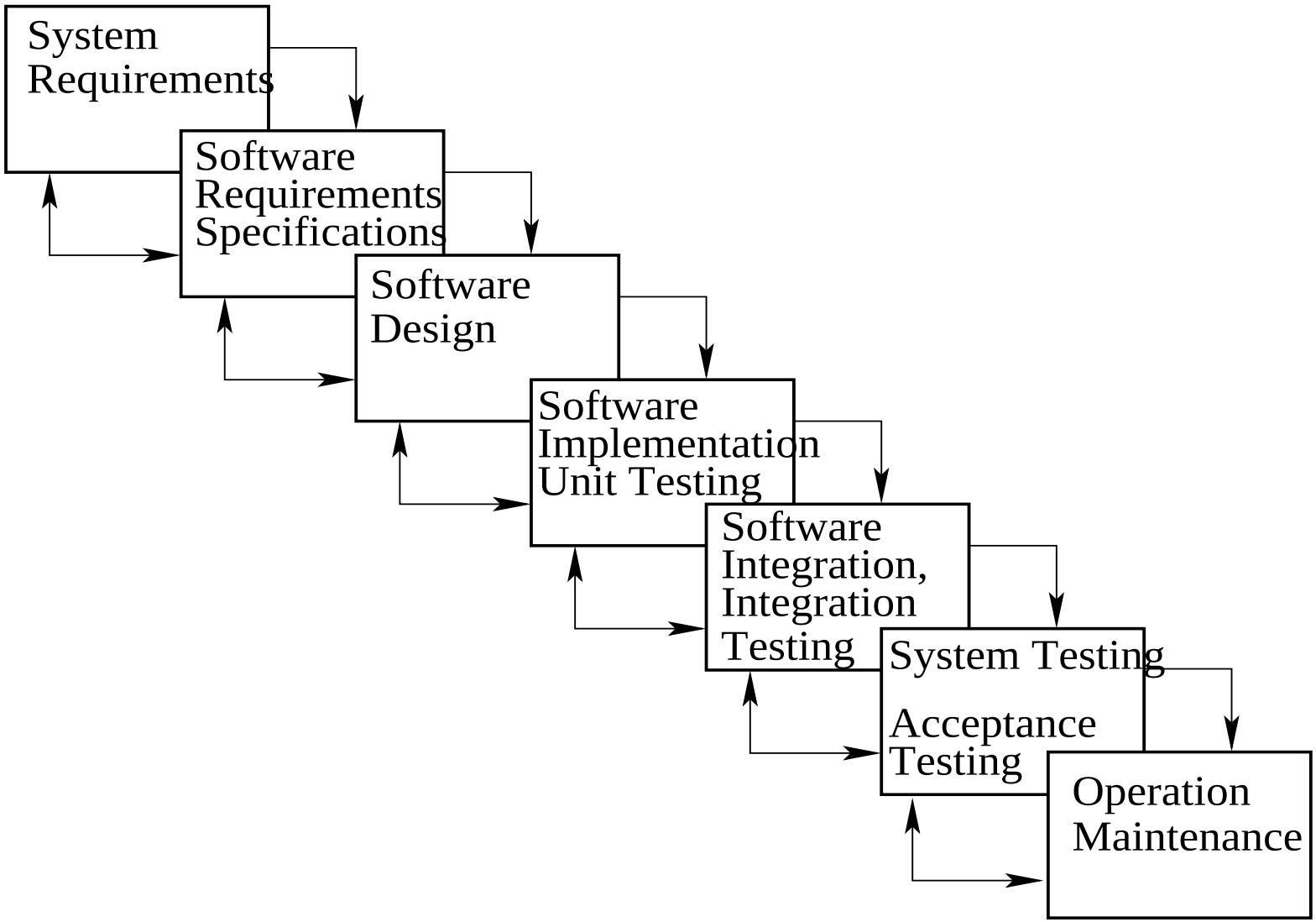
- koncept,
- požadavky,
- návrh,
- implementace (kódování),
- testování,
- používání a údržba.

Vývoj “přes stěnu” [Kol95]

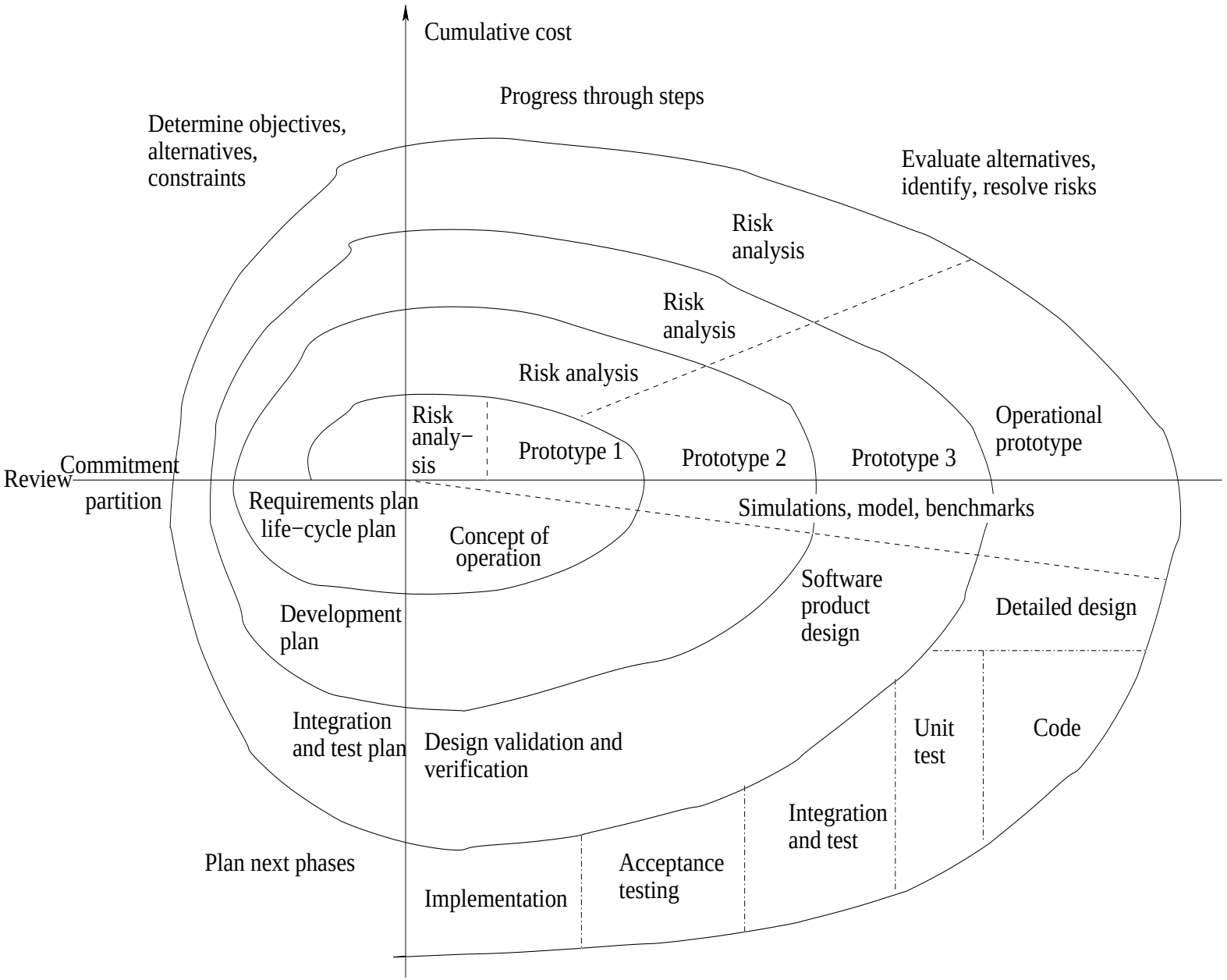
- rozděl a panuj [Kan95],
- přespecializované prostředí,
- konkurenční oddělení *impéria*,
- ukazování prstem.
- Sekvenční procesy vytváření kvality musí spolupracovat.



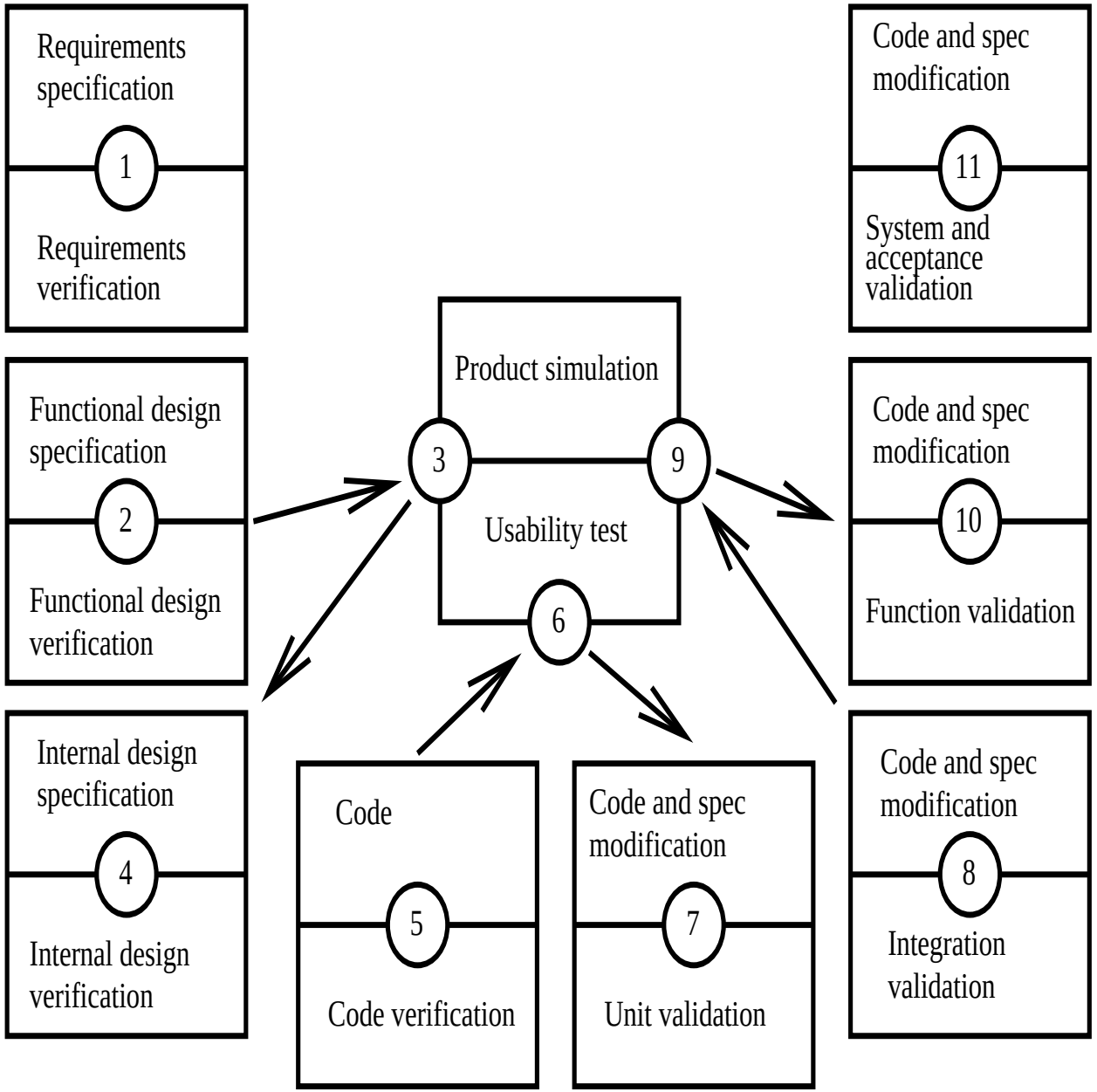
Vodopádový model [Kan95]



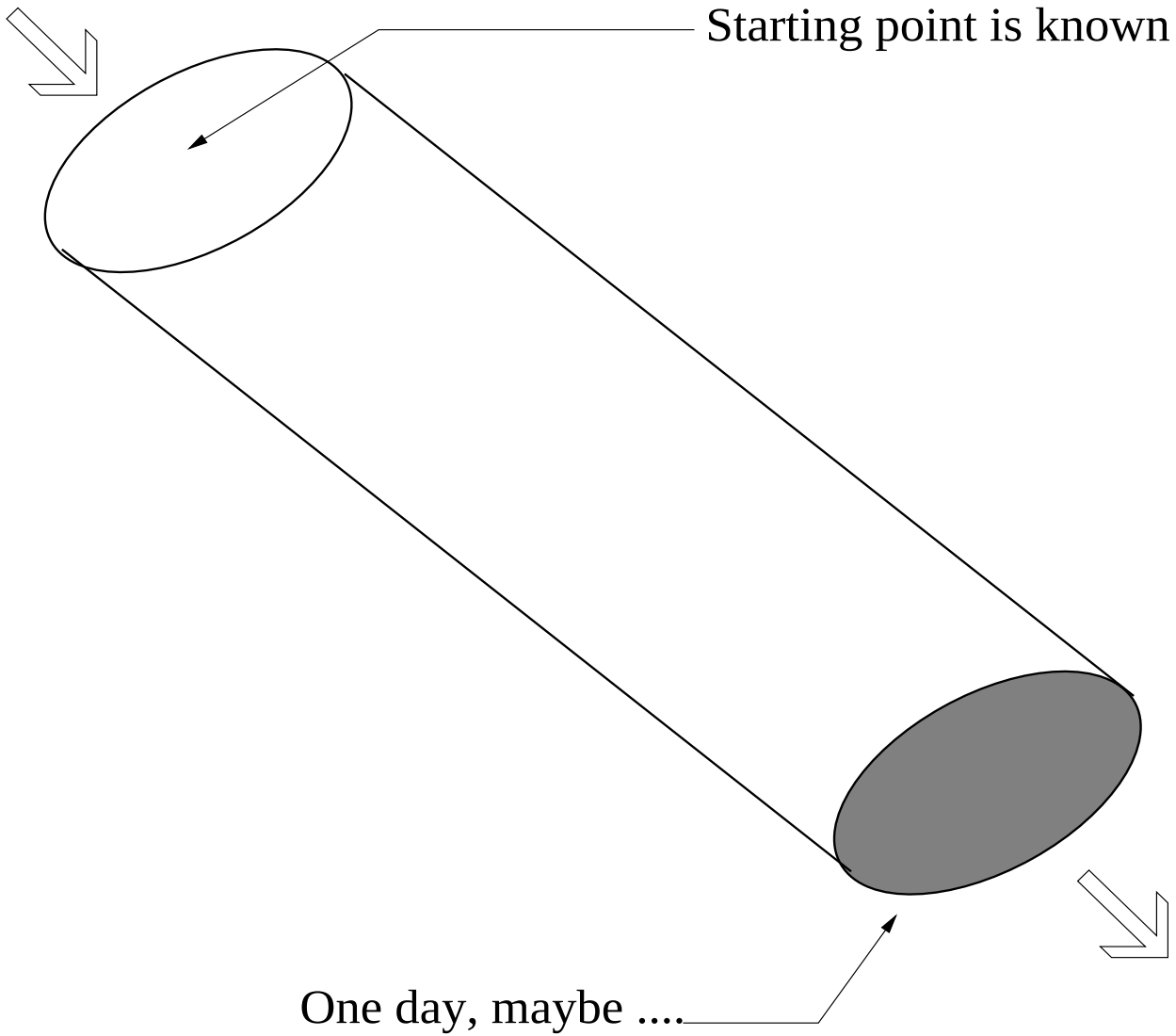
Spirálový model [Kan95]



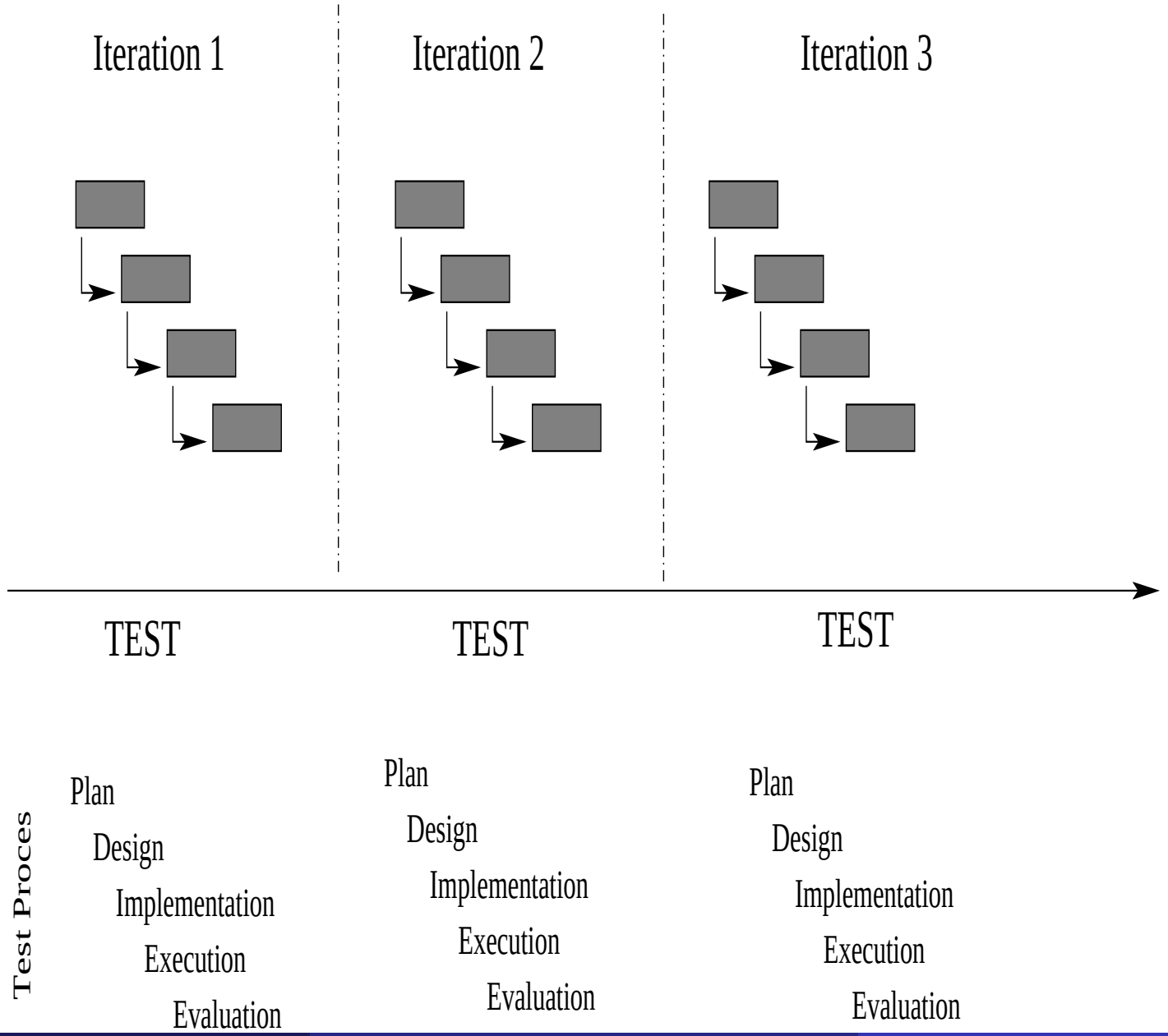
U model [Kit95]



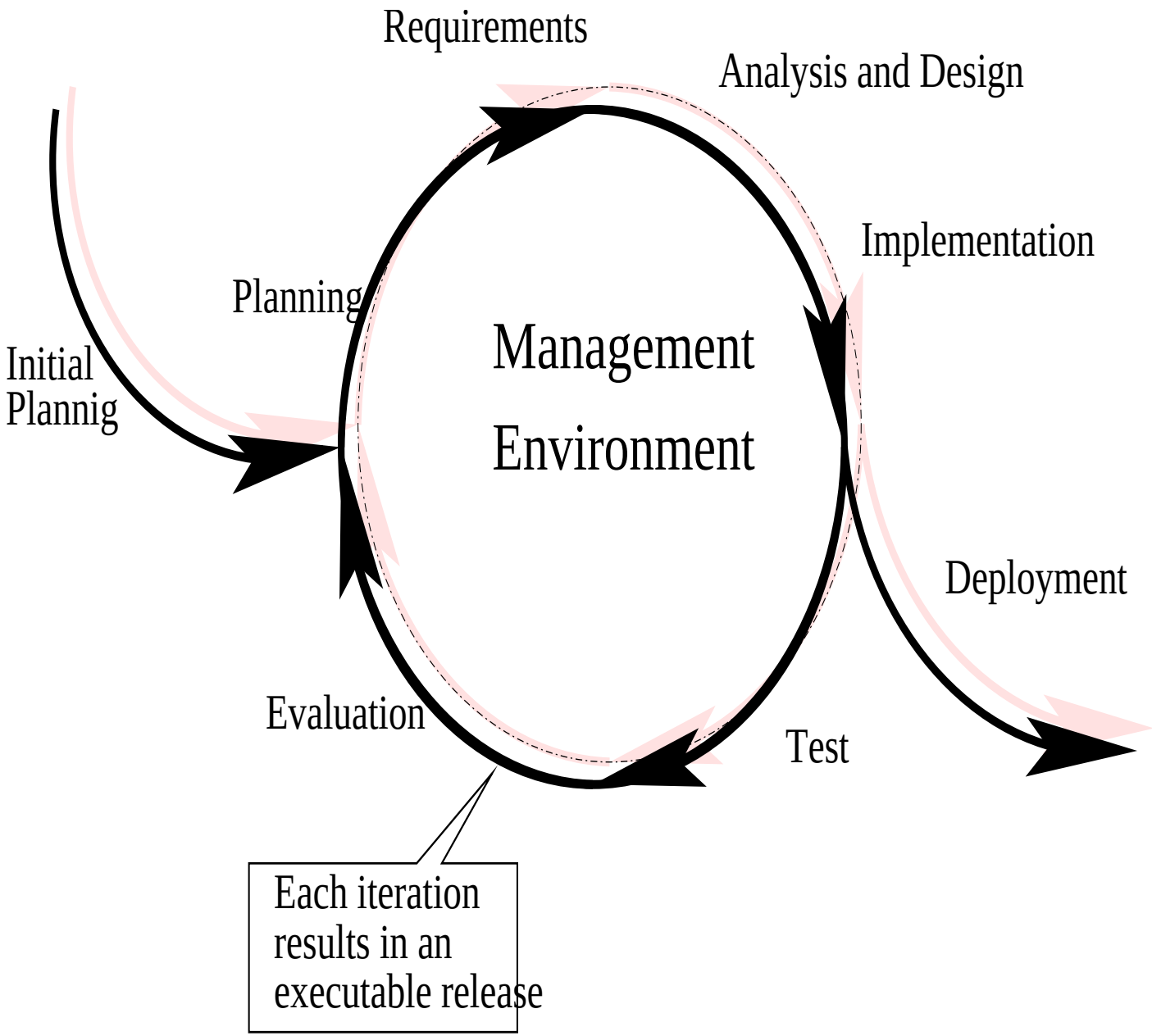
Tunelový model [Mul97]



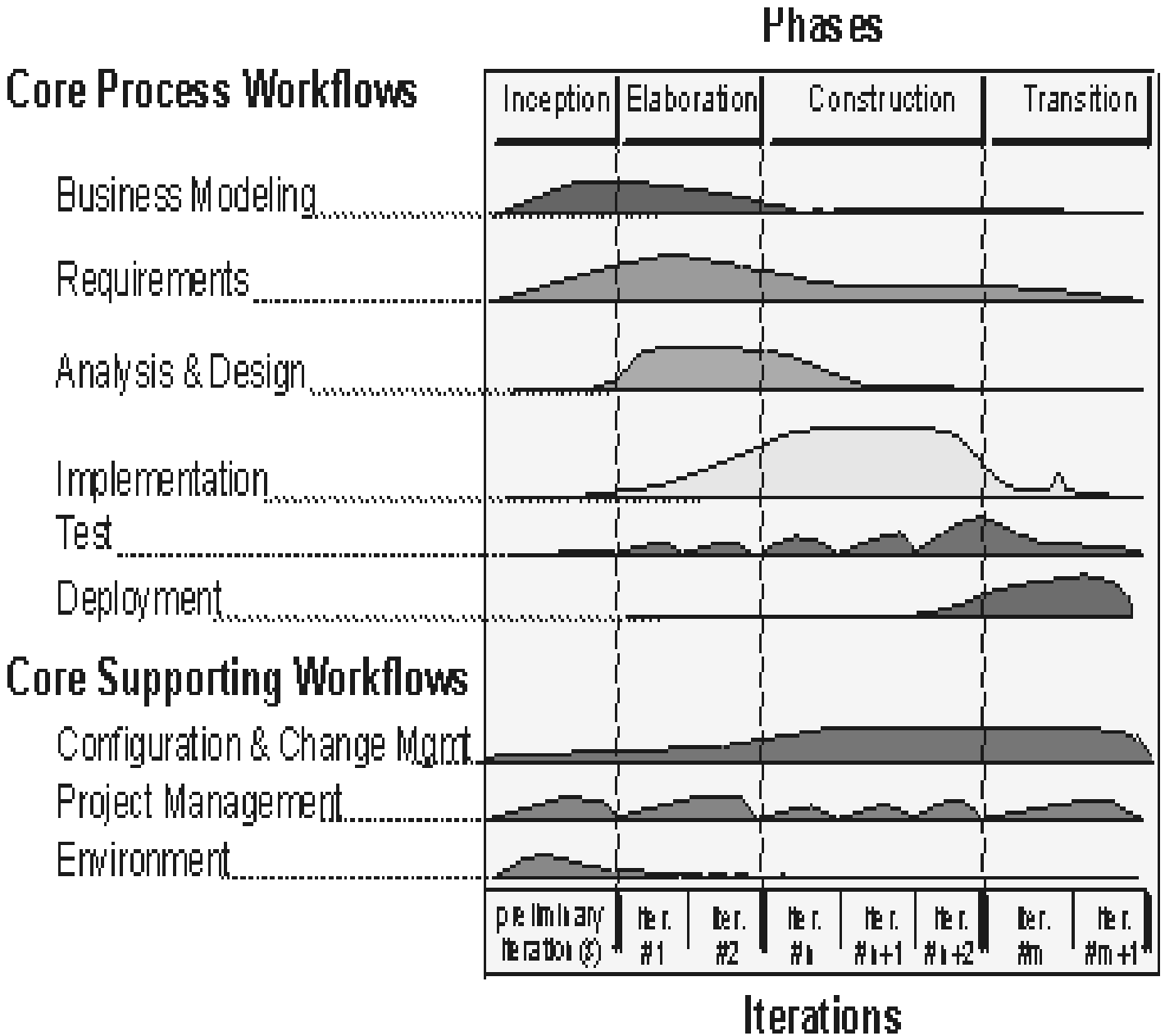
Iterativní model [Kru99]



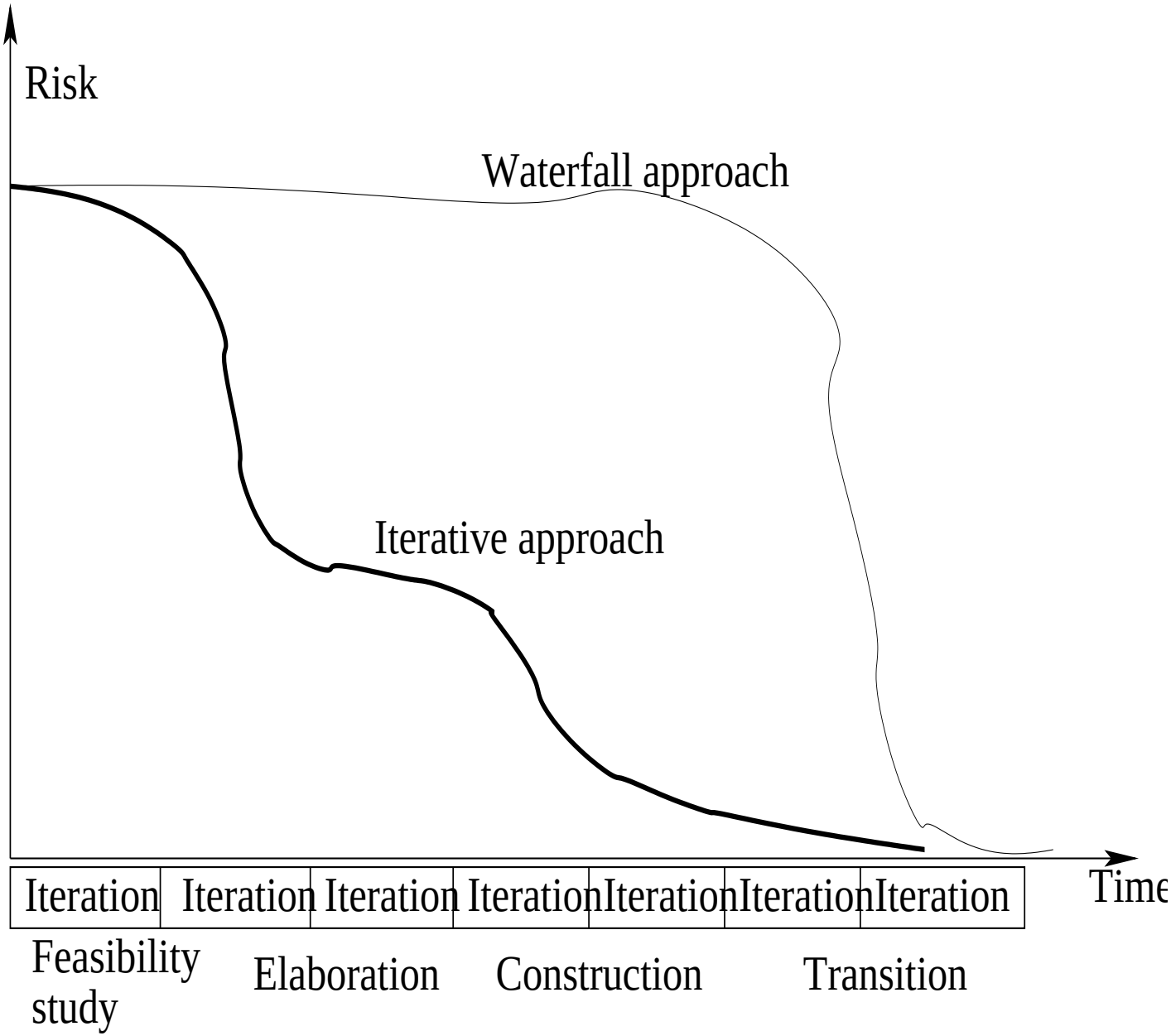
Iterativní a inkrementální model [Kru99]



Fázování procesních aktivit [Kru99]

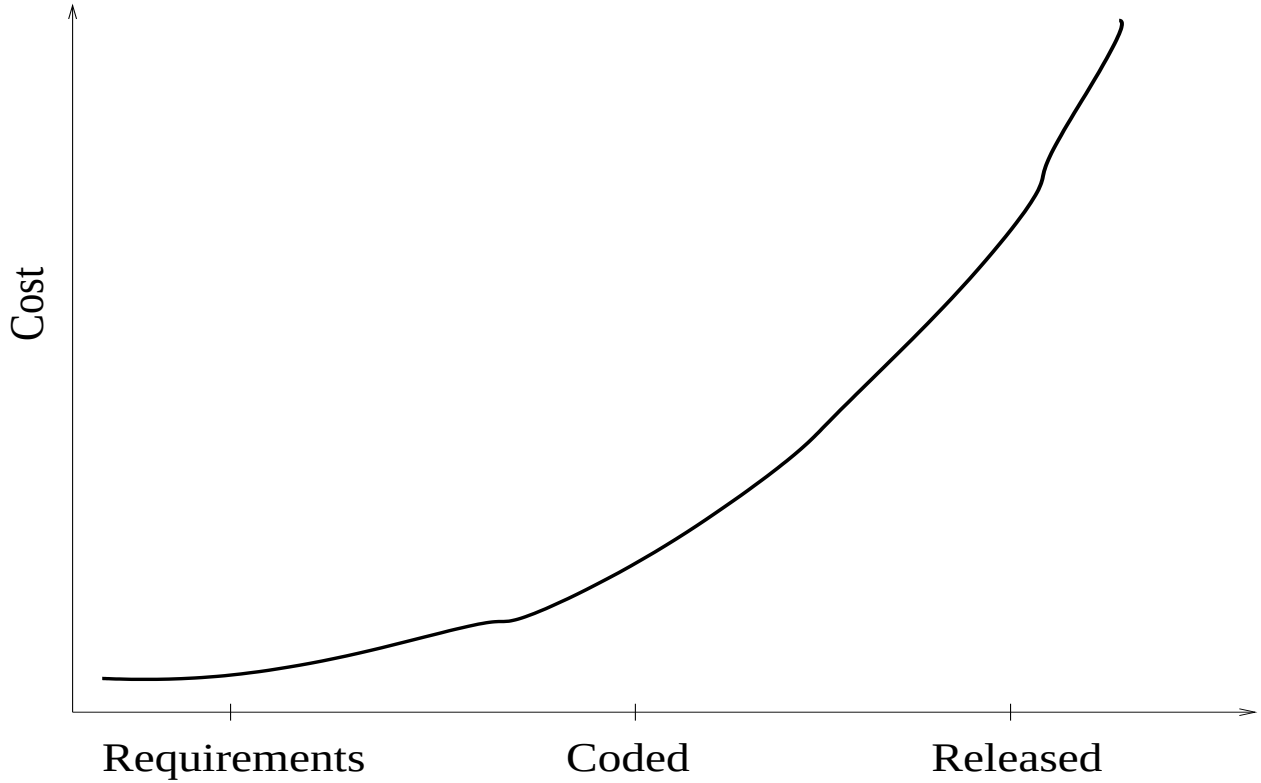


Riskové řízení [Kru99, Rat99]

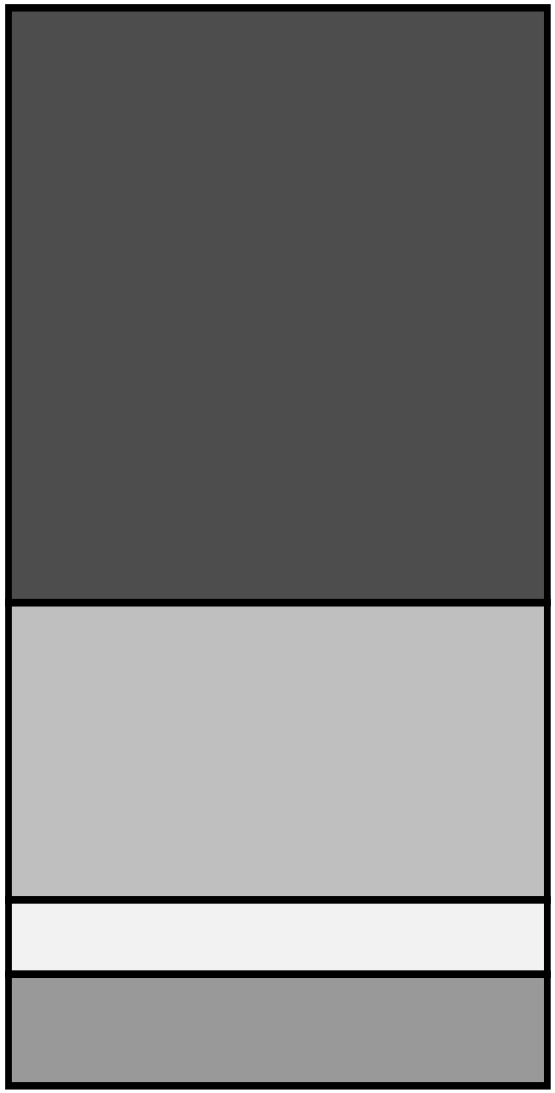


Cena nalezení a opravy chyb [KFN93]

Čím dříve je chyba nalezena a opravena, tím je levnější.



Distribuce chyb



56 % Requirements

27 % Design

7 % Code

10 % Other

Literatura I

Martin Fowler and Kendall Scott.

UML Distilled, Applying the Standard Object Modeling Language.
Addison-Wesley, 1997.

Jonathan Jacky.

The Way of Z: Practical Programming with Formal Methods.
Cambridge University, 1997.

Stephen H. Kan.

Metrics and Models in Software Quality Engineering.
Addison-Wesley, 1995.

Cem Kaner, Jack Falk, and Hung Quoc Nguyen.

Testing Computer Software.
International Thomson Computer Press, second edition, 1993.

Edward Kit.

Software Testing in the Real World.
Addison-Wesley, 1995.

William J. Kolarik.

Creating Quality: Concepts, Systems, Strategies, and Tools.
McGRAW-HILL, INC., 1995.

Philippe Kruchten.

The Rational Unified Process.
Addison-Wesley, 1999.

Literatura II

Pierre-Alain Muller.

Instant UML.
Wrox Press Ltd., 1997.

Rational software symposium 1999.

Unicorn, Praha, Czech Republic, February 1999.

Sally Shlaer and Stephen J. Mellor.

Object-Oriented Systems Analysis: Modeling the World in Data.
Prentice Hall, 1988.

Software Project Example [Jac97]

Systém řízení dokumentů

Výňatek z neformálního popisu:

- Jestliže chce uživatel změnit dokument a má na tuto operaci povolení, nikdo jiný jej právě nemění, pak uživatel si může vypůjčit (check out) daný dokument.
- Jakmile si uživatel vypůjčí dokument, ostatní si jej již vypůjčit nemohou, ale mohou jej číst.
- Když uživatel skončí editaci dokumentu, měl by jej vrátit zpět (check in), a takto povolit jiným uživatelům jeho editaci.

Grafická notace [FS97, Mul97]

- **Notace** je grafické vyjádření modelů, určuje syntaxi modelovacího jazyka.
- jazyk modelování použitelný jak lidmi tak stroji,
- Slouží k:
  - výstavbě toho správného systému - toho, který splní potřeby uživatele za rozumnou cenu,
  - komunikaci s experty domény - vysvětlení práce jiným,
  - výměně informace mezi různými účastníky,
  - snadnou manipulaci s modely,
  - reprezentaci celých systémů použitím objektově orientovaných konceptů

$$\text{Object} = \text{Stav} + \text{Chování} + \text{Identita}$$

Unified Model Language - UML [Mul97, FS97]

- grafická notace,
- průmyslový standard (OMG 1997),
- 9 různých typů diagramů . . . různých pohledů na softwarový systém.
- pomáhá komunikaci mezi lidmi,
- zachycuje systém na vysoké úrovni abstrakce.

Některé modely

model případů použití	popisuje požadavky uživatele,
model tříd	zachycuje statickou strukturu,
statový model	vyjadřuje dynamické chování objektů,
model interakcí	představuje scénáře a toky zpráv,
implementační model	určuje pracovní jednotky,
model rozmístění	poskytuje detaily příslušné k rozvržení procesů.

Diagram případu použití

- je reprezentace funkcionality systému z pohledu uživatele.
- **Případ použití** je typická interakce mezi uživatelem a systémem počítače,
  - zachycuje nějakou uživatelem viditelnou funkci,
  - může být malý či velký,
  - zajisťuje diskrétní cíl pro uživatele.
  - Případy použití se točí okolo externě vyžadované funkcionality.
- **Účastník** je role, kterou hraje uživatel vůči systému,
  - může být externí systém, který potřebuje informaci od daného systému.
- Stereotyp *rozšíření* popisuje variaci obvyklého chování.
- Stereotyp *použití* se doporučuje použít pokud se část případu opakuje.
- **Scénář** referuje jednu cestu skrz daný případ použití.

RCS Use Cases

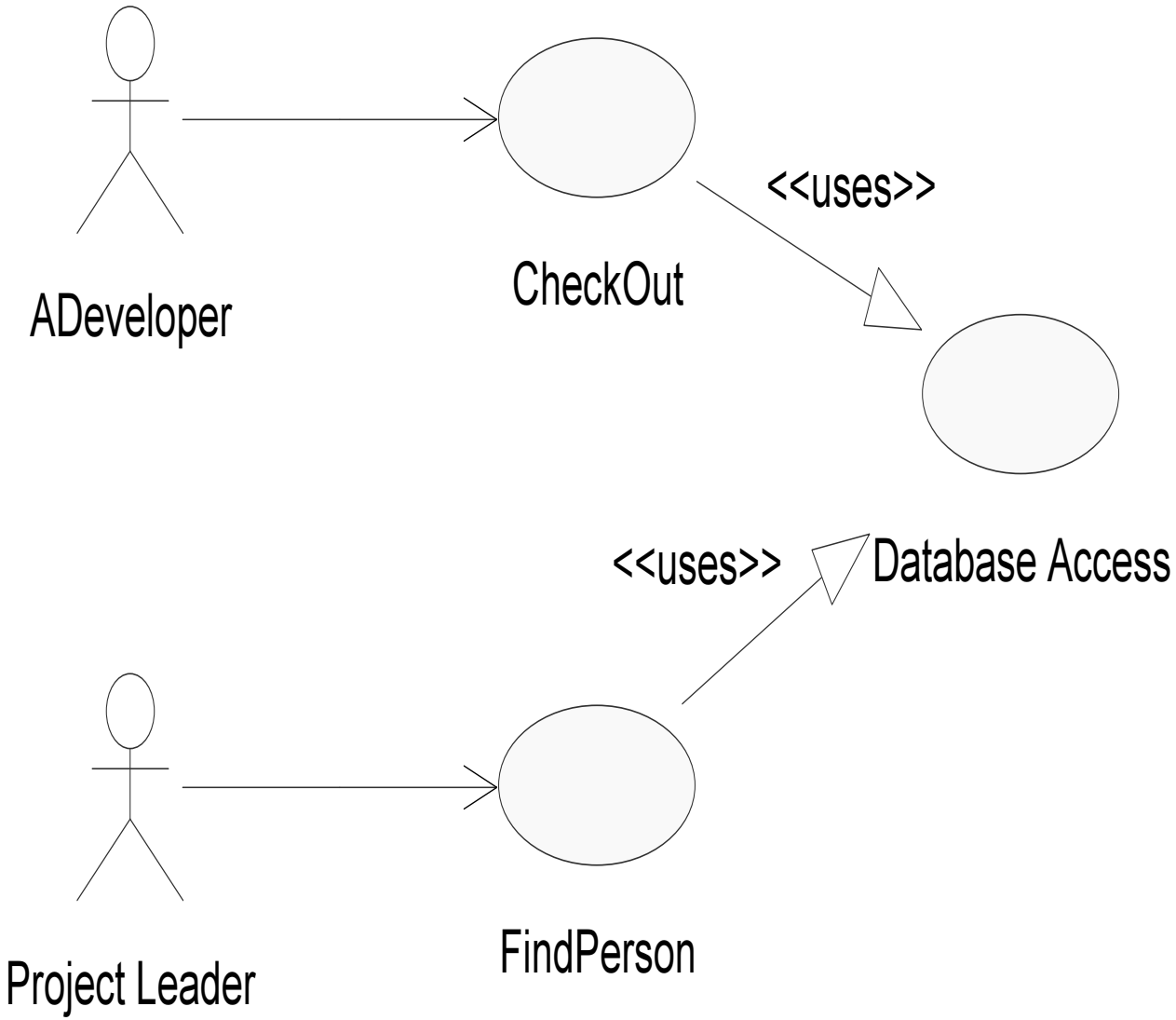


Diagram tříd - základní pojmy [Mul97, FS97]

- reprezentuje statickou strukturu systému pomocí typů objektů a jejich různých druhů statických relací.
- **Třída** je popis množiny objektů sdílející ty samé odpovědnosti, vlastnosti, operace, atributy, a sémantiku.
- **Asociace** reprezentují vztah mezi instancemi tříd.
  - Každá asociace má dvě **role**; každá role je jedním ze směrů asociace.
  - zdroj, cíl, multiplicita.
- **Atribut** [SM88] je abstrakce *jedné* charakteristiky vlastněnou všemi entitami, které samy byly abstrahovány do třídy.
- **Operace** jsou procesy, které třídy mohou provést (odpovídají metodám třídy).
- **Agregace** je druh asociace který vyjadřuje silnějšího vazbu mezi třídami
  ($\approx$ reference).
- **Kompozice** je silnější forma agregace
  ($\approx$ hodnota).

Class Diagram - dědičnost [Mul97]

- **Zobecnění** spočívá ve vyčlenění společných prků v rámci množiny tříd do jedné obecné třídy zvané **supertřída**.
- **Specializace** dovoluje zachytit speciální vlastnosti množiny objektů, které nejsou popsány dosud identifikovanými třídami (**podtřída**).

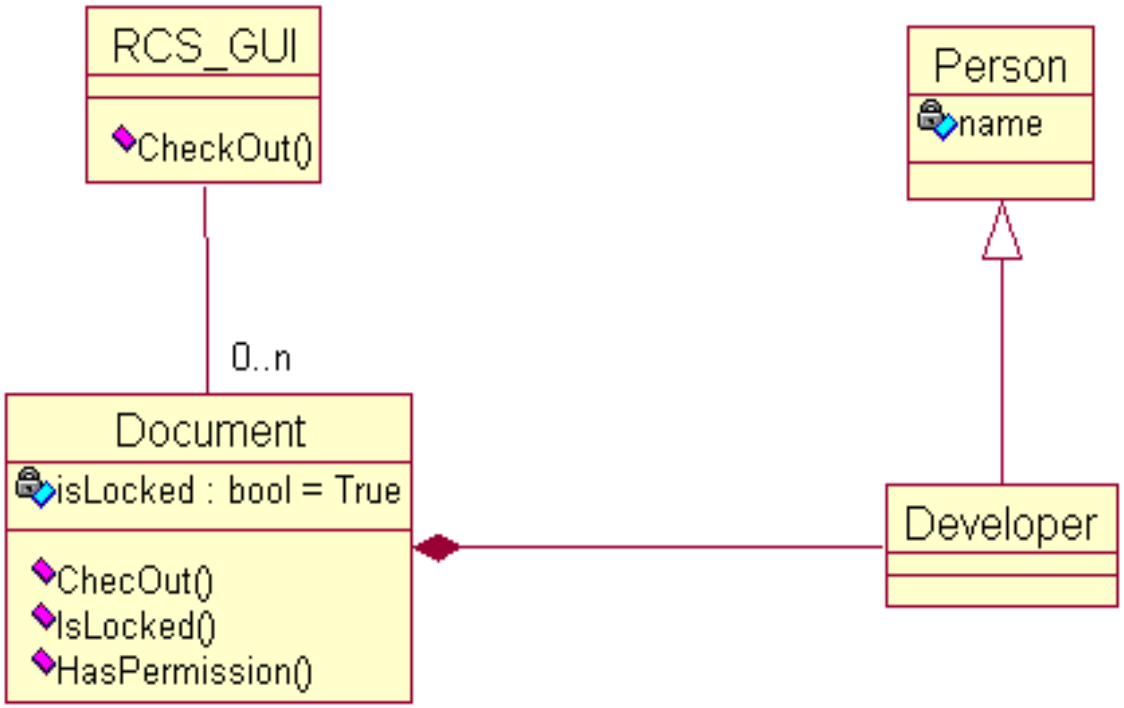
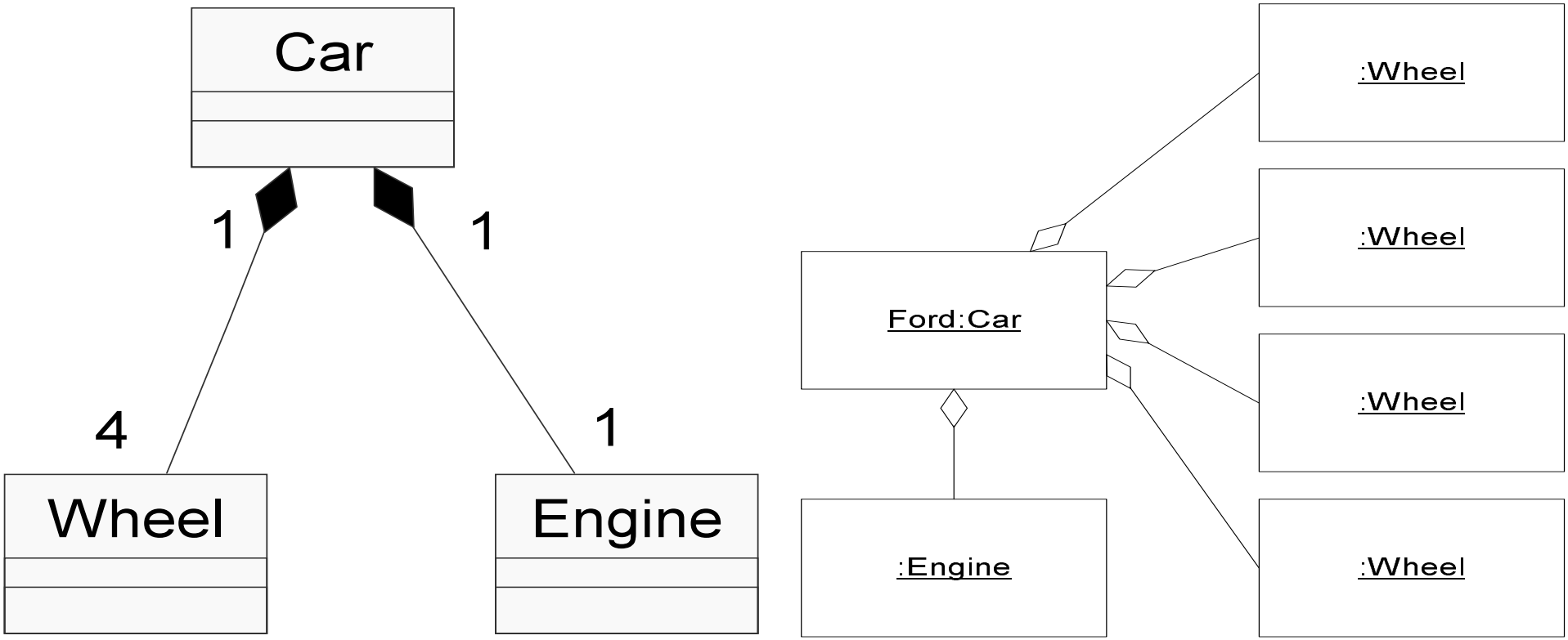


Diagram objektů [Mul97]

- vyjadřuje statickou strukturu systému pomocí objektů a jejich vztahů,
- je instancí diagramu tříd.



Sekvenční diagramy [Mul97]

- ilustruje interakci mezi objekty použitím časových struktur, které určují pořadí komunikace.
- vysílání:
  - **synchronní** při kterém vysílač je blokován a čeká na ukončení zpracování zprávy volaným objektem,
  - **asynchronní** při kterém odesílatel není blokován a může pokračovat ve zpracování své agendy.
- **Aktivace** koresponduje s dobou, po kterou objekt vykonává nějakou akci.

RCS CheckOut sekvence

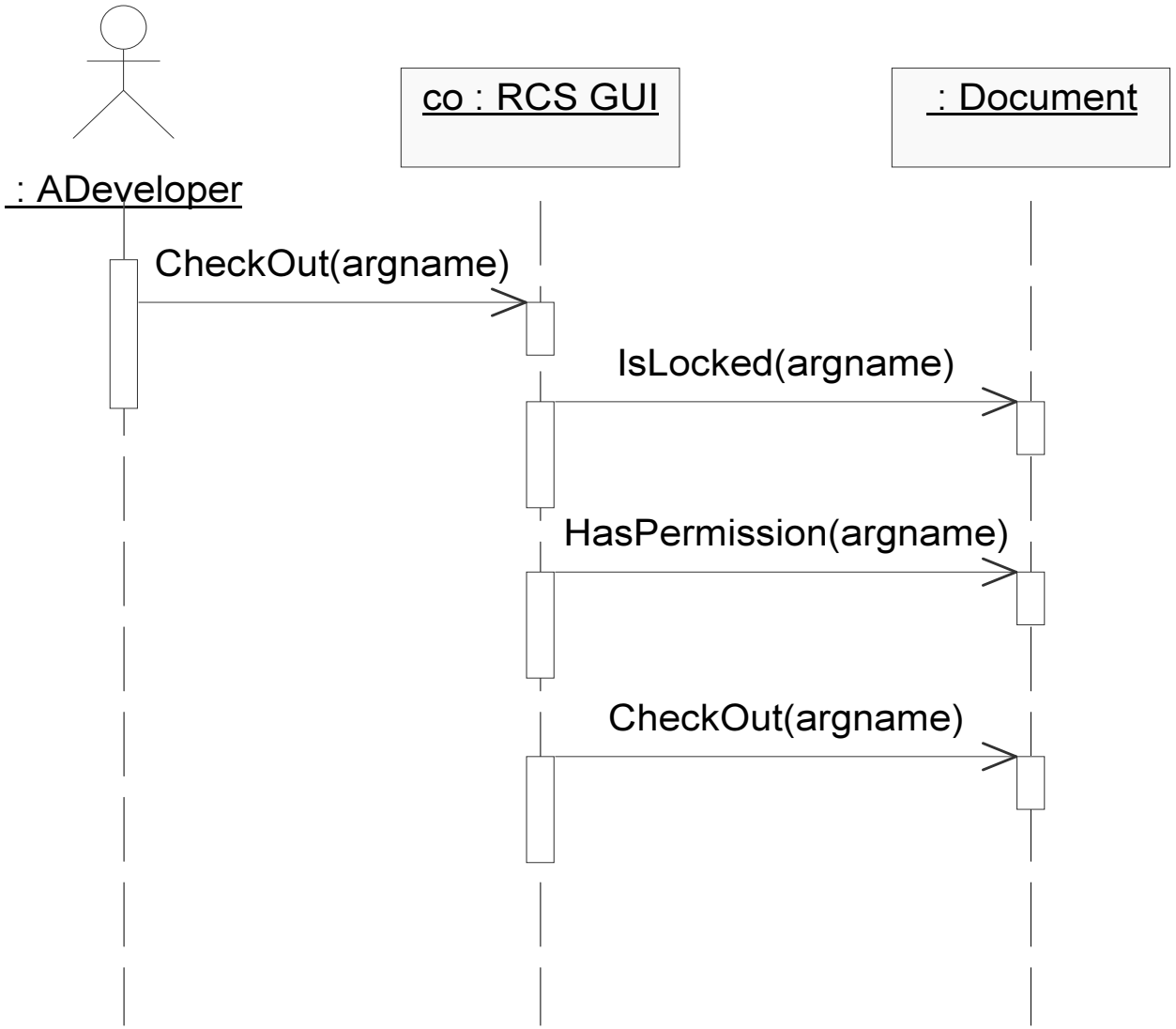


Diagram kolaborace [Mul97]

- ilustruje interakce mezi objekty použitím prostorové struktury, která představuje fyzické rozmístění,
- čas se nevyjadřuje explicitně,
- zprávy jsou očíslovány podle pořadí odeslání.

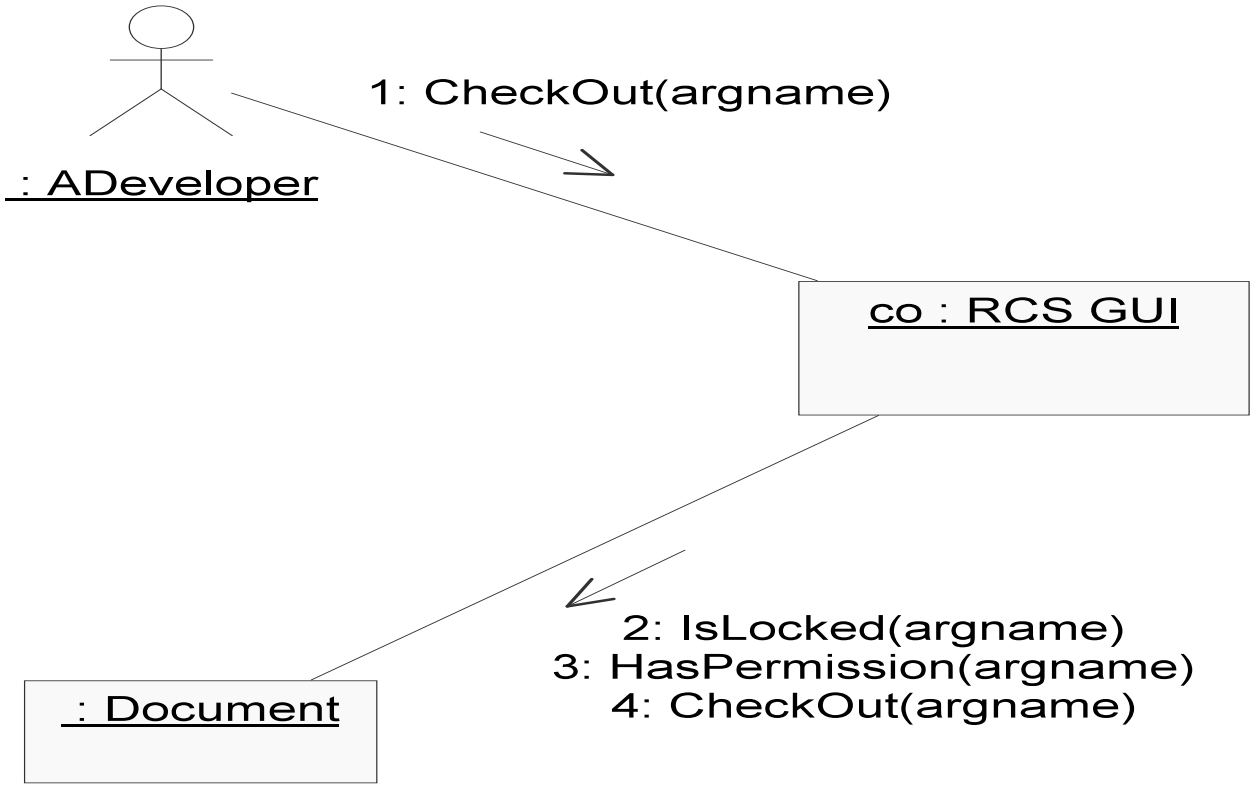
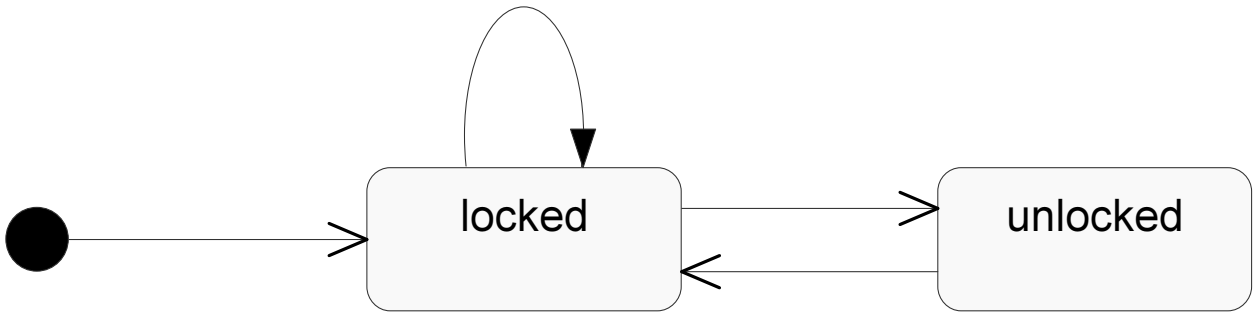
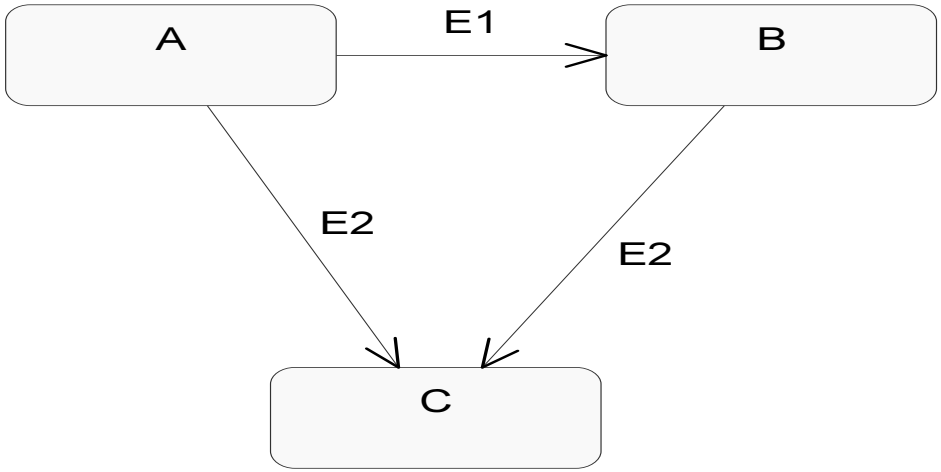


Diagram stavových schémat (statechart) [Mul97]

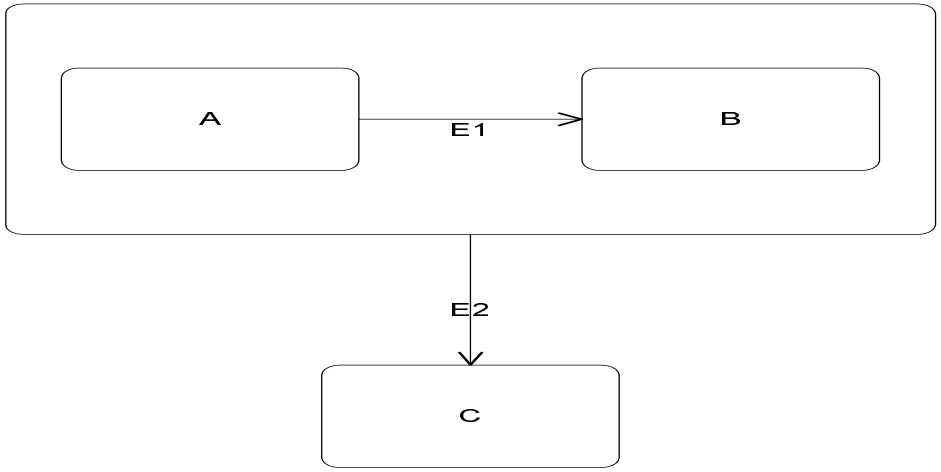
- zachycují chování tříd použitím statových automatů založených na pojmech stavu a přechodů,
- **stav**, **počáteční stav**, **koncové stavy**,
- Změna z jednoho stavu do jiného se provede tehdy, když je daný **přechod** iniciován **událostí**, která nastane v rámci dané domény problému.
- Stavová schémata jsou hierarchické stavové automaty.
  - agregace: je kompozice jednoho stavu z několika jiných nezávislých stavů (konjuktivní typ).
  - zobecnění: maskuje detaily,
    - obecnější stavy se nazývají **superstavy**,
    - speciálnější stavy se nazývají **podstavy**,



Příklad stavového schématu



(a) klasicky



(b) hierarchicky

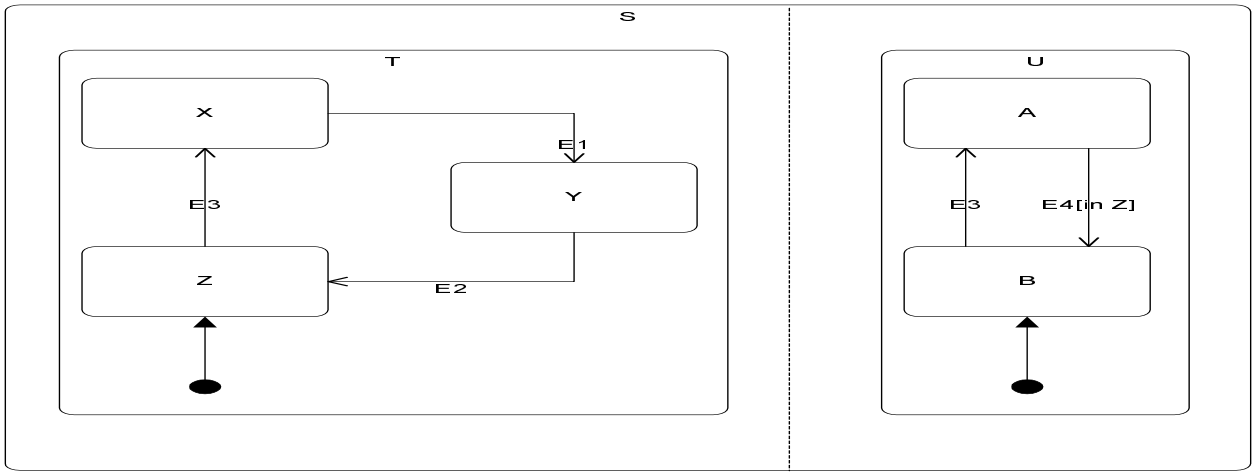


Diagram aktivit [Mul97, FS97]

- reprezentují chování operací užitím množiny akcí organizovaných do sekvencí kroků se sekvenčním či paralelním provedení větví řízení.
- **aktivita**:
  - úloha, která je potřeba udělat,
  - metoda třídy,
- **rozhodnutí**: **blokující podmínky (guard conditions)** řídí, které přechody se uskuteční.
- **synchronizační brána** smí být překročena, pokud byly iniciovány všechny vstupní přechody.

Diagram komponent [Mul97]

- popisuje softwarové komponenty aplikace v implementačním prostředí.

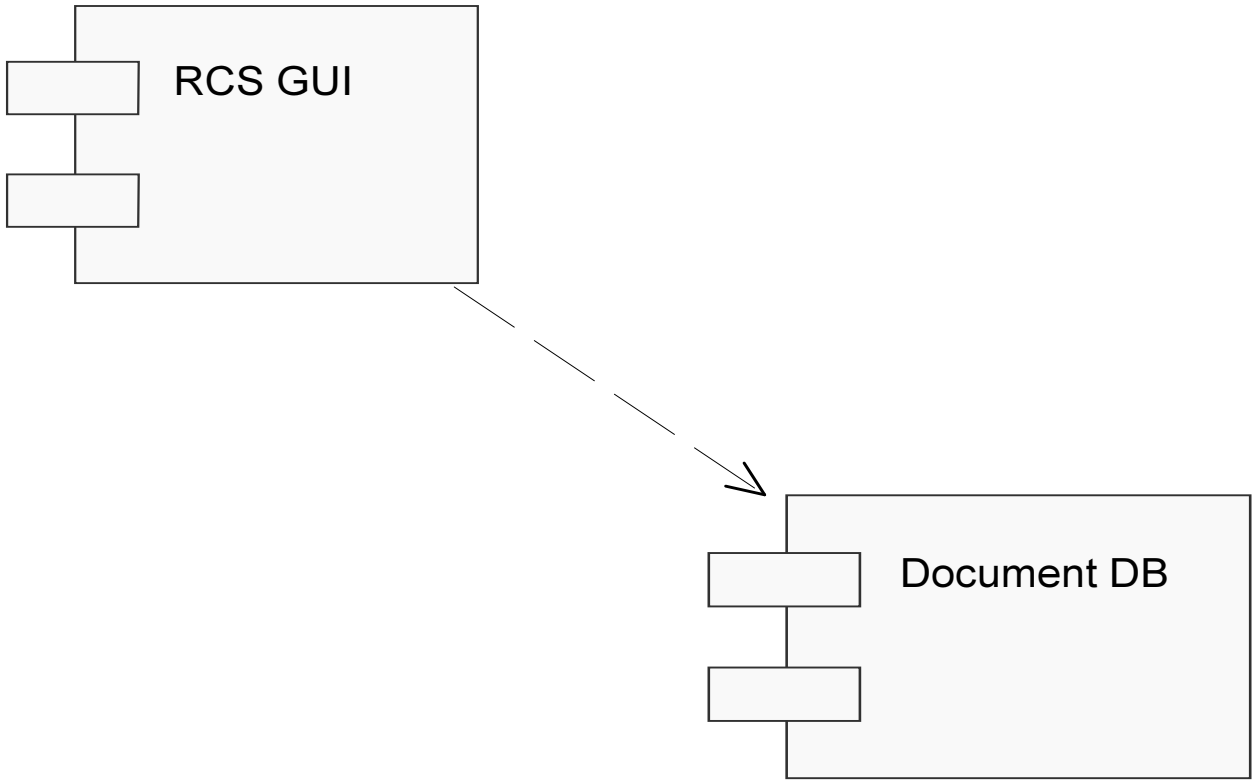


Diagram rozmístění [Mul97]

- ukazuje umístění softwarových komponent na hardwarových komponentách.