MASARYKOVA UNIVERZITA

Projekt vývoje informačního systému

Jaroslav Šmarda

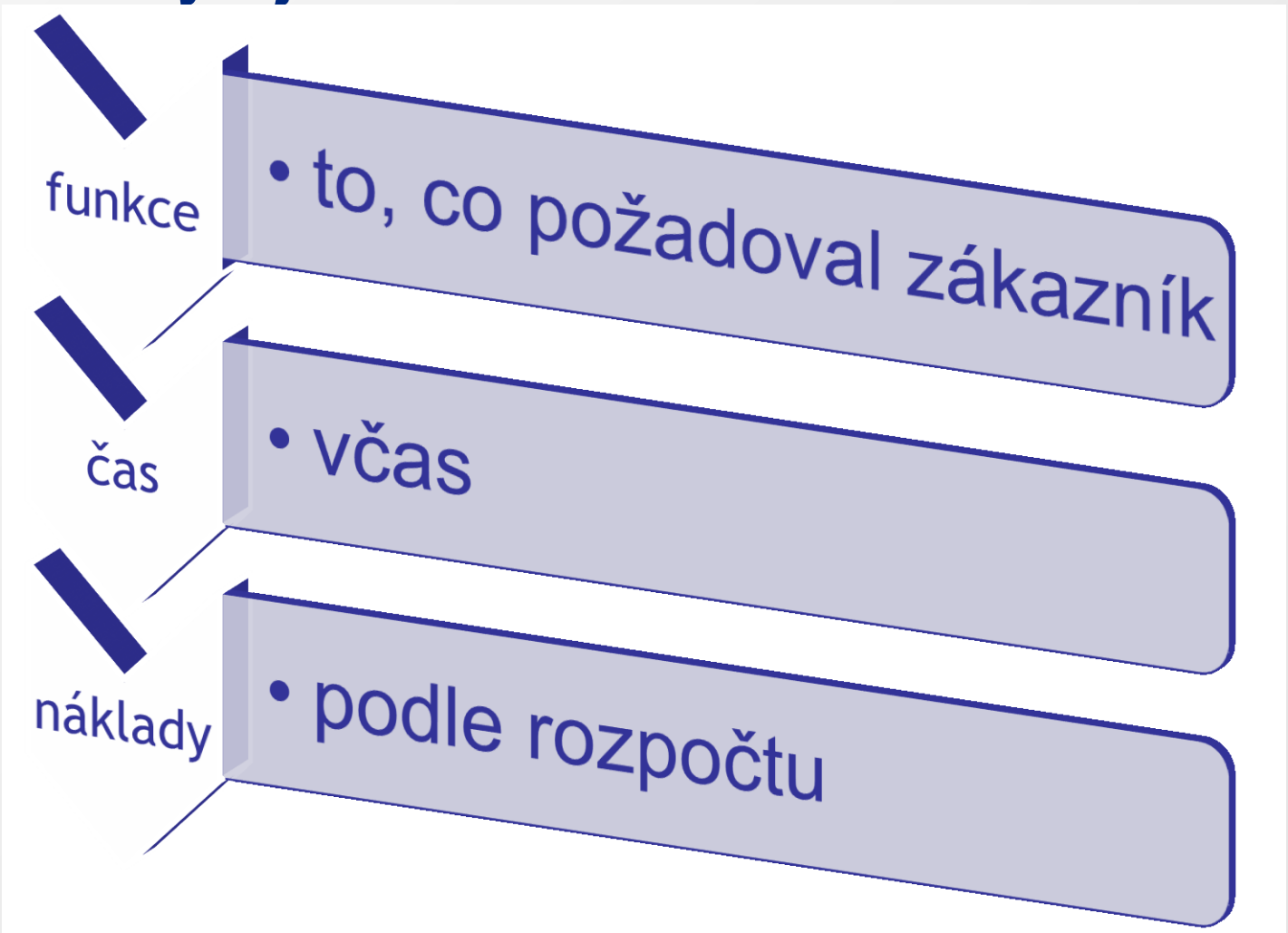
Projekt vývoje IS na příkladu jednoduché aplikace pro evidenci knih a časopisů

- Princip iteračního vývoj IS
- 1. krok – Analýza požadavků zákazníka
- 2. krok – Návrh
- 3. krok – Kódování
- 4. krok – Testování
- Praktická ukázka vývojových nástrojů

Projekt vývoje IS a zákazník

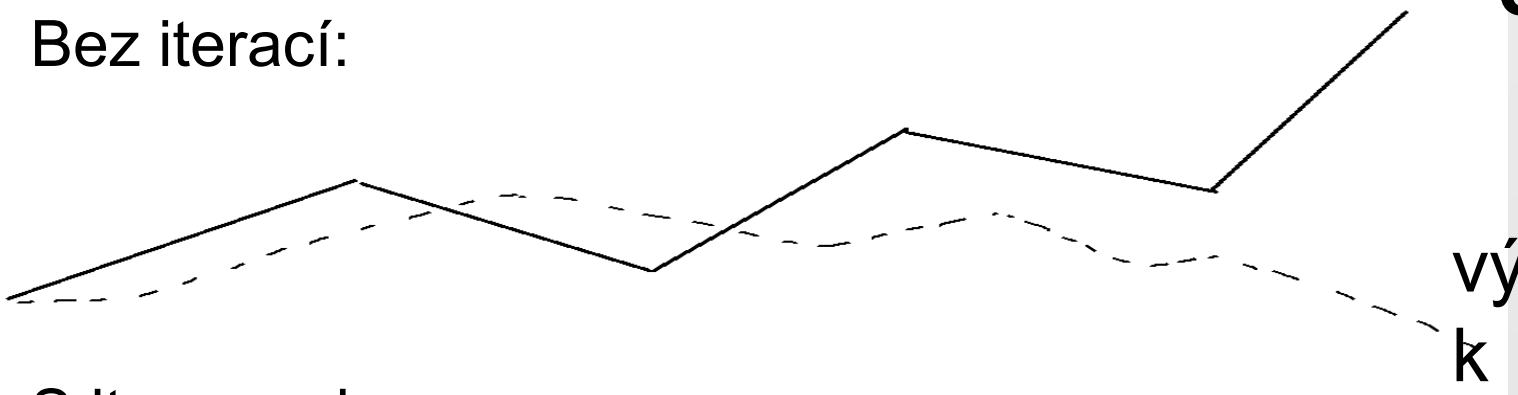
- Zákazník v centru vývoje
- Dvě základní otázky na projekt:

Výsledek vývoje IS

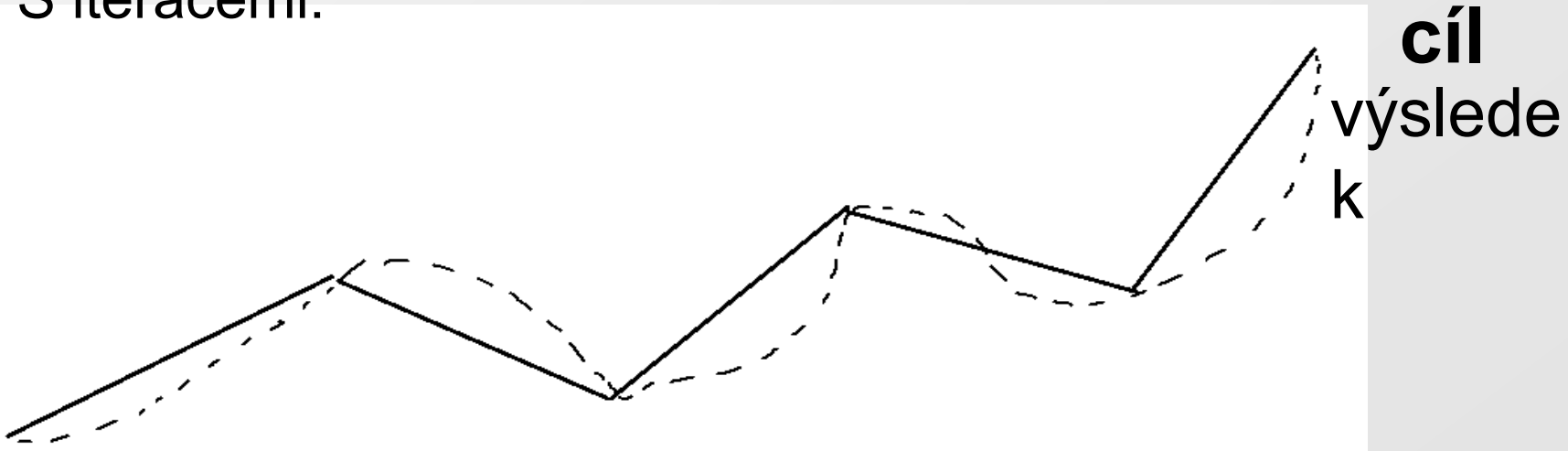


Iterace jako základ úspěšného vývoje

- Bez iterací:



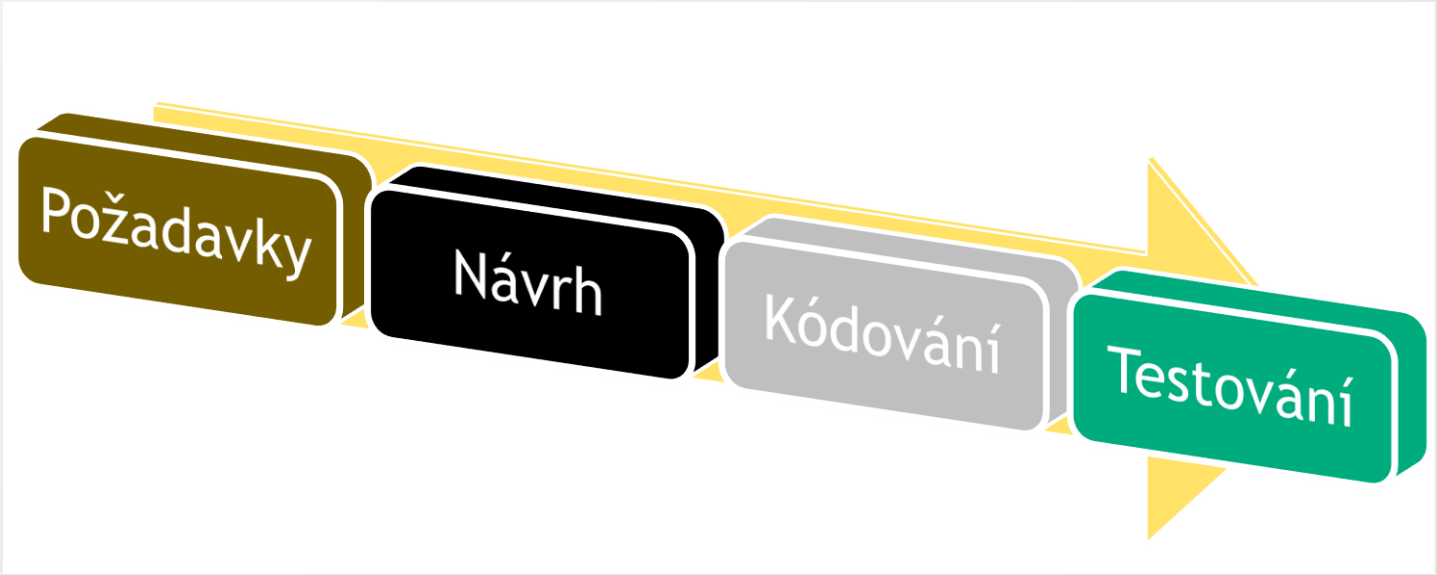
- S iteracemi:



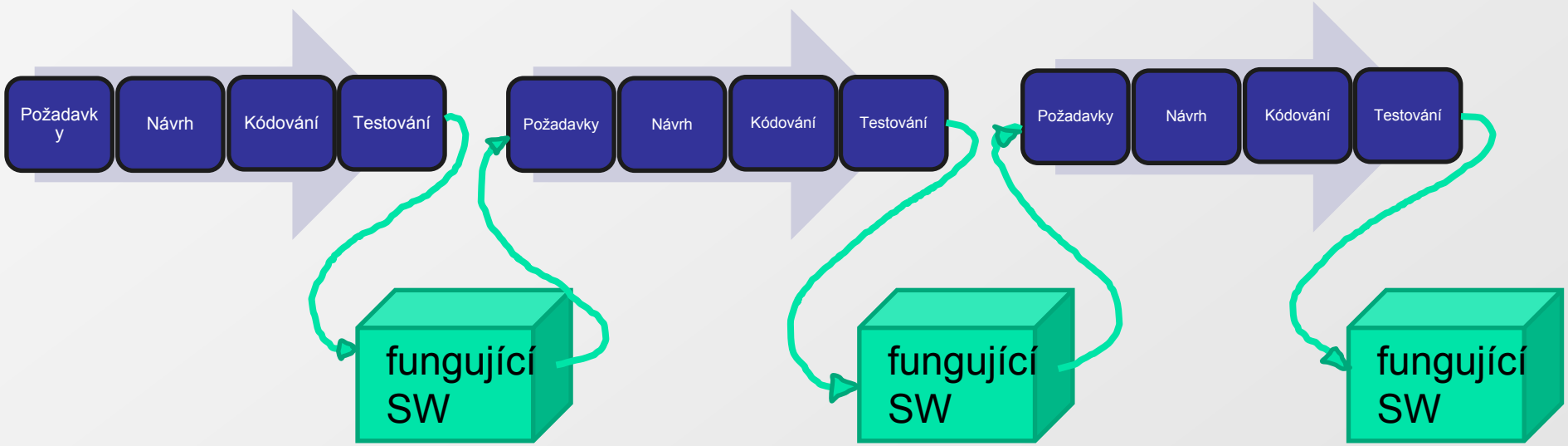
Iterace je víc než proces

4 kroky iterace

- Požadavky – analýza požadavků zákazníka
- Návrh řešení
- Kódování – programování
- Testování – ověřování



Každá iterace je mini-projekt



SW není hotový, pokud nebyl implementován u zákazníka

1. krok iterace – Požadavky zákazníka

Název: Vypůjči knihu
Popis: *Vytvořit záznam o výpůjčce knihy – jaká kniha, kdo, kdy půjčil, kdy vrátí*

Název: Vrať knihu
Popis: *Vytvořit záznam o výpůjčce knihy – jaká kniha, kdo, kdy vrátit*

Název: Prodluž výpůjčku
Popis: *Aktualizovat záznam o výpůjčce – údaj o datu vrácení a o kolikáté prodloužení se jedná*

Název: Vlož novou knihu
Popis: *Vložit nový záznam o nové knize*

Název: Vyřaď poškozenou knihu
Popis:

Název: Zruš výpůjčku ztracené knihy
Popis:

Název: Zobraz seznam vypůjčených knih
Popis:

Název: Zobraz seznam výpůjček po lhůtě pro vrácení
Popis:

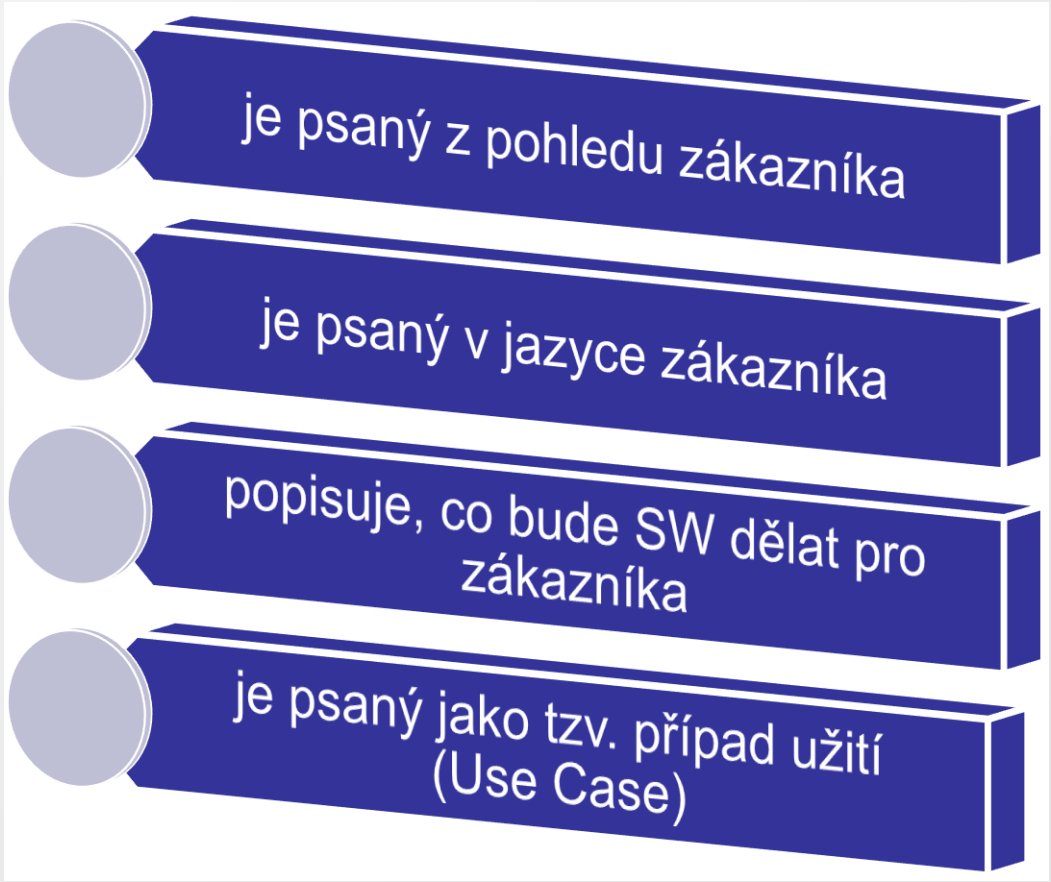
Název: Odešli upomínku na vrácení knihy
Popis:

Požadavky zákazníka

- Získat co nejvíce informací od zákazníka (brainstorming)
- Používané techniky:
  - Hraní rolí
    - vizualizace toho, co dělá SW
    - analytik dělá to, co SW, zákazník ho instruuje
  - Sledování
    - analytik sleduje zákazníka, co dělá, a hledá místa, kde pomůže SW

Požadavky zákazníka

- Správný požadavek:



Požadavek – případ užití (Use Case)

- Případ užití by **MĚL**:
  - popisovat jednu věc, kterou má SW dělat pro zákazníka
  - být psán v jazyce, kterému zákazník rozumí
  - být psán zákazníkem (zákazník to řídí, nemusí to psát)
  - být krátký

Požadavek – případ užití (Use Case)

- Případ užití by **NEMĚL**:
  - být dlouhou esejí
  - používat technické termíny neznámé zákazníkovi
  - zmiňovat specifické technologie

Požadavky – odhad doby trvání (případu užití)

- Případ užití:
  - definuje CO má SW dělat
  - nedefinuje JAK to má SW dělat
- Odhad trvání:
  - definuje KDY to má být hotovo

Průměr z více odhadů

Odhad doby trvání realizace požadavků v člověkodnech

Případ užití	Můj odhad	Odhad - Petr	Odhad - Jana
Vypůjčit knihu		3	2
Vrátit knihu		4	5
Prodloužit výpůjčku		3	3
Vložit novou knihu		5	6
Vložit čtenáře		4	3
Vyřadit poškozenou knihu		5	4
Zrušit výpůjčku ztracené knihy		6	5
Zobrazit seznam vypůjčených knih		4	6
Zobrazit seznam výpůjček po lhůtě pro vrácení		5	5
Odeslat upomínku na vrácení knihy		10	9

Odhad doby trvání realizace požadavku

- Zákazník **předpokládá**, že ….
- **Eliminovat** nebo **vyjasnit** všechny skryté předpoklady
- Každý přetrvávající **skrytý předpoklad** je **rizikem**

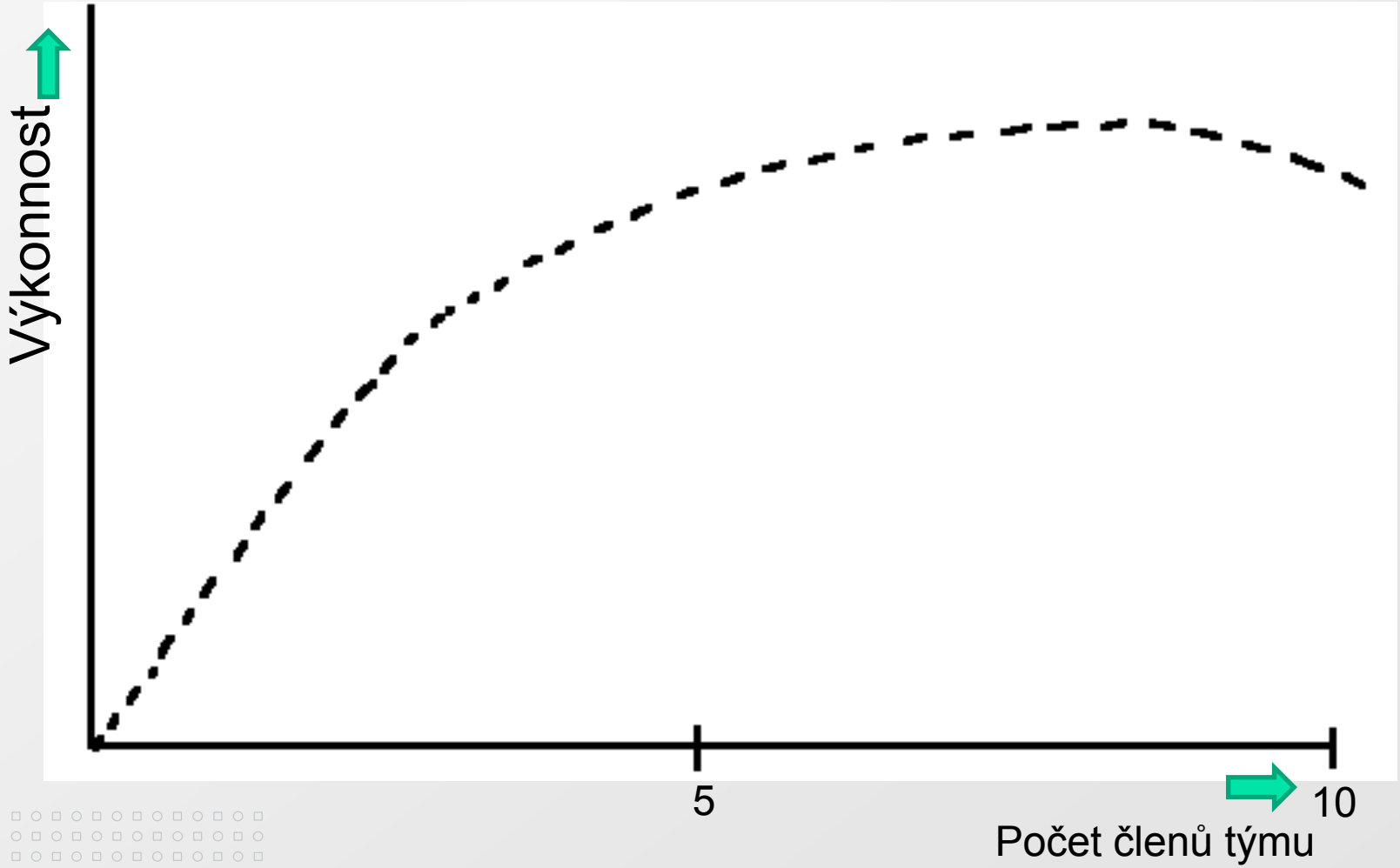
Odhad doby trvání realizace požadavků

Plánování celého projektu

- Stanovení priorit pro jednotlivé požadavky – se zákazníkem
- **Verze 1.0 – milník 1.0**
  - První verze distribuovaná zákazníkovi

Termín Verze 1.0 je ohrožen

Projektový tým a výkonnost



Iterace a plán Verze 1.0

1. iterace

Název: Vlož novou knihu Odhad: 5 dní Priorita: 10	Název: Vlož čtenáře Odhad: 3 dní Priorita: 10	Název: Vypůjči knihu Odhad: 4 dní Priorita: 10	Název: Vrať knihu Odhad: 3 dní Priorita: 10

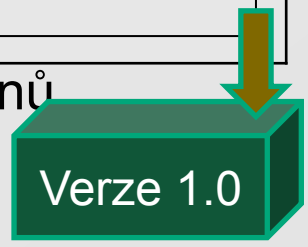
2. iterace

Název: Prodluž výpůjčku Odhad: 3 dní Priorita: 15	Název: Vyřaď poškozenou knihu Odhad: 3 dní Priorita: 20	Název: Zruš výpůjčku ztracené knihy Odhad: 6 dní Priorita: 20	Název: Zobraz seznam vypůjčených knih Odhad: 4 dní Priorita: 30

3. iterace

Název: Zobraz seznam výpůjček po lhůtě pro vrácení Odhad: 5 dní Priorita: 30	Název: Odešli upomínku na vrácení knihy Odhad: 9 dní Priorita: 40

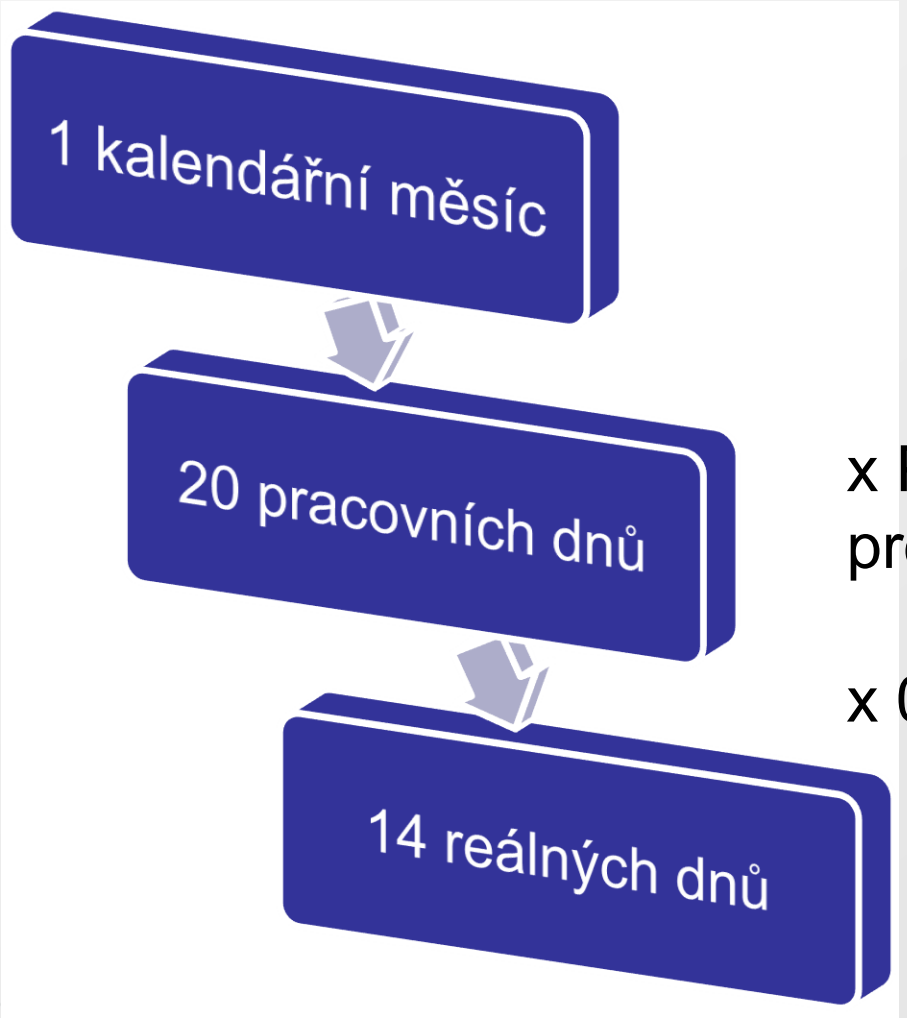
Délka iterace – 1 měsíc – zhruba 20 pracovních dnů



Produktivita projektu

- Produktivita projektu = poměr určující produktivní čas (menší než 1.0)
- Zvolíme počáteční produktivitu **0.7**
- Podle skutečnosti v dalších projektech produktivitu upravíme

Reálné dny pro plánování projektu



x Produktivita projektu

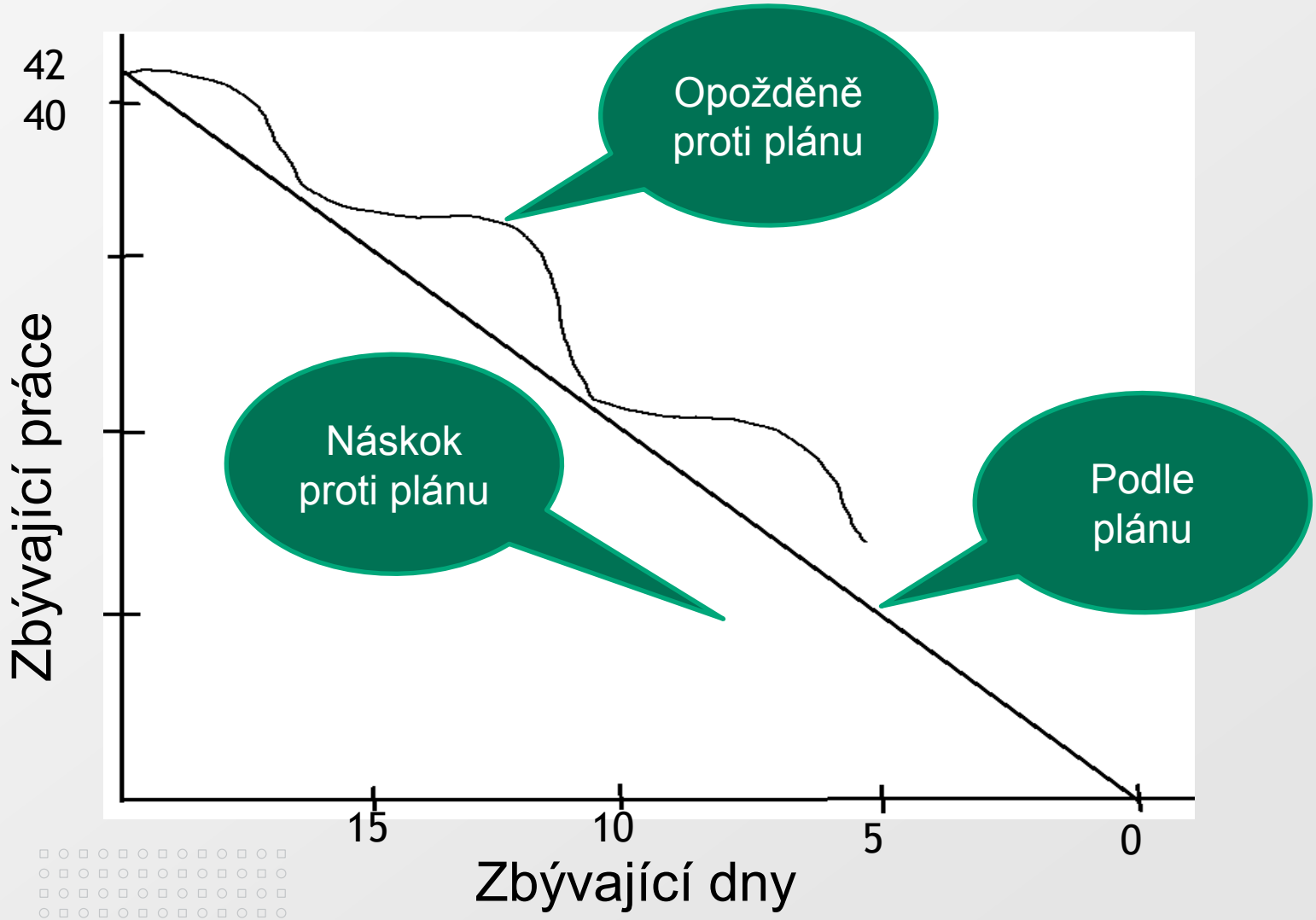
x 0.7

Tým a reálné plánování projektu

Počet člověkodnů pro plánování jedné iterace						
Počet členů týmu		Pracovní dny jedné iterace		Produktivita projektu		Počet člověkodnů pro plánování jedné iterace
3	X	20	x	0.7	=	42

Počet člověkodnů pro Verzi 1.0				
Počet iterací do Verze 1.0		Počet člověkodnů pro jednu iteraci		Počet člověkodnů pro Verzi 1.0
3	x	42	=	126

Přehled o stavu projektu



Rozdělení požadavků na úkoly

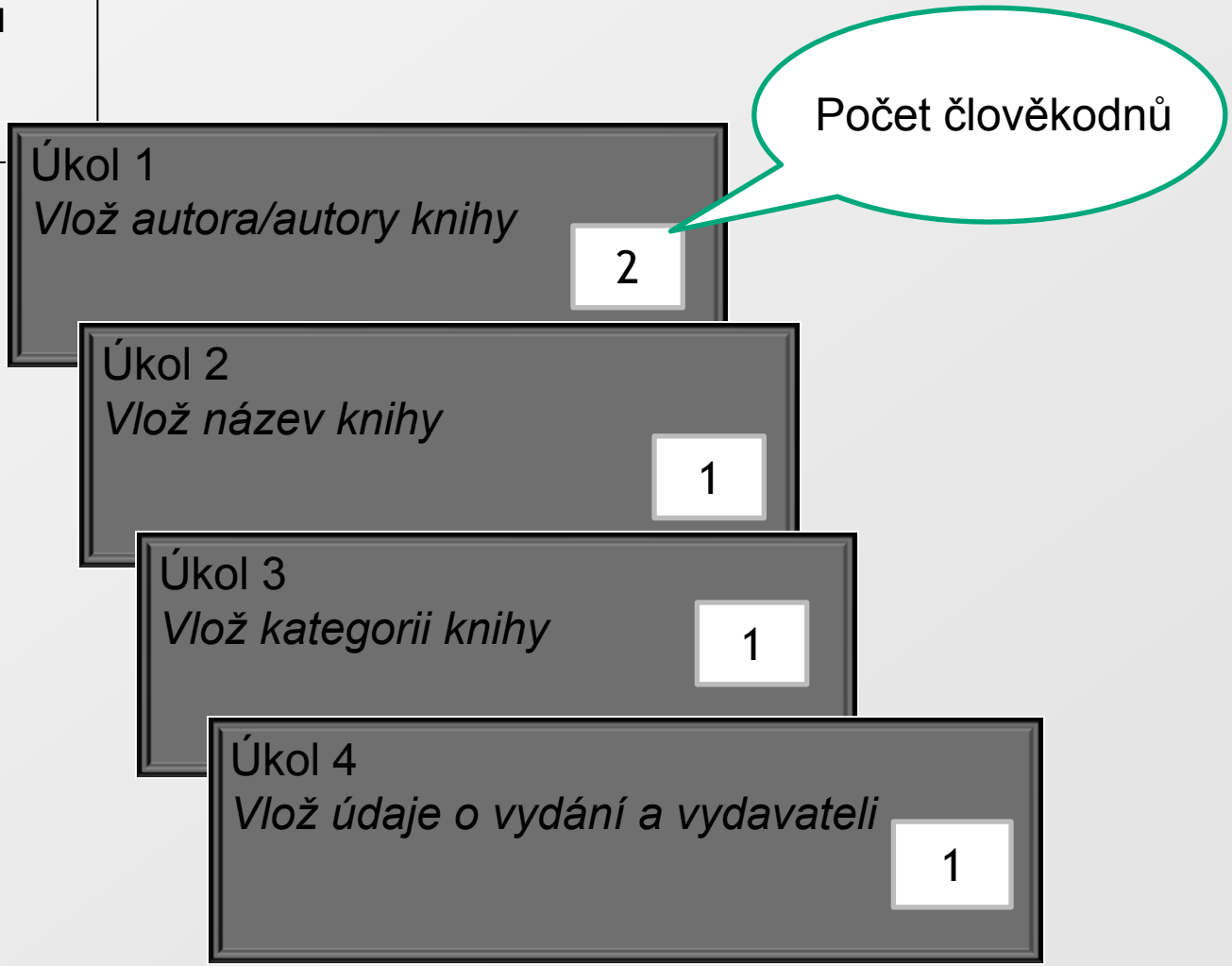
- Požadavky jsou jednotky projektu z pohledu zákazníka, ale pro zpřesnění plánu je potřeba je rozdělit
- **Požadavek** (případ užití) představuje skupinu **úkolů**

Rozdělení požadavků na úkoly

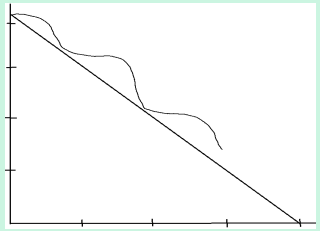
- Úkol je práce (část požadavku) pro jednoho vývojáře
- Úkol má:
  - název
  - popis
  - časový odhad v člověkodnech

Požadavek a úkoly

Název: Vlož novou knihu
Odhad: 5 dní
Priorita: 10



Stav plnění úkolů iterace

Případy užití	Rozpracováno	Dokončeno	
PU: Vlož knihu U1 U2	U2	U1	
PU: Vlož čtenáře U1 U2 U3	U1 U2		Odloženo (odložené PU - případy užití)
PU: Vypůjči knihu U1 U2 U3			
PU: Vrať knihu U1 U2			Dokončeno (dokončené PU)

Návrh – objektově orientovaný návrh

- Strukturování případu užití
- Objekty:
  - podstatná jména v uživatelském příběhu
  - Kniha, čtenář, výpůjčka, upomínka
- Atributy objektů:
  - vlastnosti objektů
  - Název knihy, autor knihy, údaje o vydání knihy, jméno čtenáře
- Metody objektů:
  - slovesa v uživatelském příběhu
  - Vypůjčit knihu, vrátit knihu

Návrh - objekty

- Objekt:
  - Černá skříňka, která s okolím komunikuje prostřednictvím svých metod
  - Zapouzdření (Encapsulation)

Objekt

- Kombinuje data a funkce do jediné soudržné jednotky
- Ukrývá svoje data za vrstvu funkcí

Objekt má

- Identitu objektu:
  - jedinečný objektový odkaz (Object Reference)
  - něco jako adresa v paměti
- Stav:
  - hodnoty atributů
- Chování:
  - operace, které popisují chování objektu
  - implementace (realizace) operace se nazývá **metoda**
  - metody zpravidla mění stav (atributy) objektu

Návrh – Princip jediné odpovědnosti objektu (Single Responsibility Principle)

- V textu jsou slovesa (metody) ve vztahu k více podstatným jménům (objektům)
  =>
  Otázka: Ke kterému objektu patří metoda?
- Odpovědnost je zde definována jako důvod ke změně

Návrh – Princip jediné odpovědnosti objektu (Single Responsibility Principle)

- Každý objekt musí mít právě jeden důvod ke změně
- Když dojde ke změně, víme, kam se podívat
- Jinak malá změna vyvolá spoustu změn v programu (ripple effect – vlnění)

Princip jediné odpovědnosti

- Objekt Kniha:
  - Metody:
    - Vložit knihu (Autoři, Název, Kategorie, Údaje o vydání a vydavateli, Klíčová slova)
    - Vyřadit knihu (Identifikace knihy)
- Objekt Výpůjčka:
  - Metody:
    - Vypůjčit knihu (Identifikace knihy, Čtenář, Datum vrácení)
    - Vrátit knihu (Identifikace výpůjčky)
    - Prodlužit výpůjčku (Identifikace výpůjčky, Datum vrácení)

Návrh objektů – má být suchý (DRY)

- Každá informace nebo chování jen na jednom místě
- Použití abstrakce, vytknutí obecného chování

- Výpůjčka knihy a výpůjčka skript NE (stejná funkcionalita)

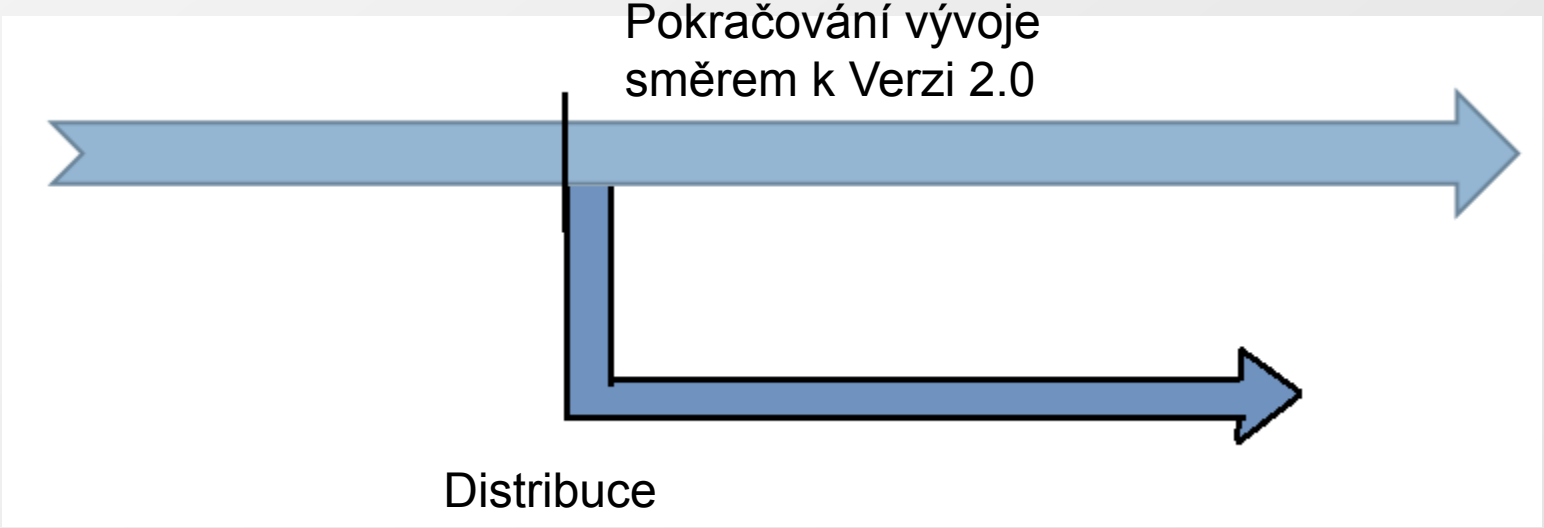
- Abstrakce Výpůjčka, která popisuje stejnou funkcionalitu pro knihu i skripta

Kódování

- Zdrojové kódy v programovacích jazycích např.:
  - Java, C++, C#, Objective C, Python

Kódování – Řízení konfigurace (CM – Configuration Management)

- Během vývoje více verzí aplikace
  - Vývojové větve (branch)



Kódování – Řízení konfigurace

- Kdy větvit vývoj?
  - Jedině v případě, že je to nezbytně nutné!
  - Byla distribuována Verze zákazníkovi, ale pokračuje vývoj
  - Je potřeba udělat radikální změny v kódu

Kódování – Řízení konfigurace

- Kdy nevětvit vývoj?
  - Při formálním rozdělení kódu do více zdrojových souborů
  - Pokud máte programátory, kteří nejsou schopni udržet zdrojový kód bez chyb

Kódování – sestavení (build) aplikace

Součásti projektu

- Adresáře se zdrojovými texty a testy
- Binární soubory – např. obrázky a ikony
- Knihovny, dll, jars
- Soubory s definicemi projektu, XML soubory, aplikační konfigurace

Proces sestavení

Funkční aplikace (.exe, .msi apod.)

Kódování – vývojové prostředí

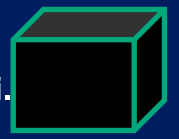
- Vývojové prostředí (IDE – Integrated Development Environment nebo SDK – Software Development Kit):
  - editor zdrojového kódu,
  - kompilátor,
  - program pro hledání chyb (debugger),
  - systém pro vizuální návrh grafického uživatelského rozhraní,
  - v případě objektově orientovaného programování také prohlížeč objektů (object browser)

Kódování – vývojové prostředí

- Příklady IDE:
  - Microsoft Visual Studio (Windows),
  - Xcode (Mac OS, iOS),
  - NetBeans,
  - Eclipse

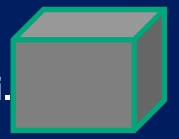
Testování

- 3 pohledy na vyvíjenou aplikaci:
  - Uživatelé vidí aplikaci jako černou skříňku (black box)
    - Zajímají se jen o funkcionalitu
  - Testeři vidí aplikaci jako šedou skříňku (grey box)
    - Sledují například, zda aplikace uvolnila všechny zdroje OS
  - Vývojáři vidí aplikaci jako bílou skříňku (white box)
    - Sledují i kvalitu zdrojového kódu, návrh objektů apod.



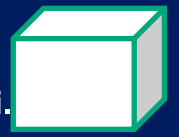
Testování aplikace jako černé skříňky

- Funkcionalita:
  - Jaké výstupy odpovídají vstupům?
- Ověření vstupních hodnot:
  - Je možno zadat 30. 2. 2011?
- Graficky korektní výstupy
  - např. tabulky
- Přechody mezi stavy
- Hraniční případy
  - 31.12.,
  - vstupní hodnoty mimo hranice (13. měsíc)



Testování aplikace jako šedé skříňky

- Podobné jako u černé skříňky, ale zaměřené víc systémově než uživatelsky
- Přihlašování do aplikace
- Komunikace s jinými systémy – předávané soubory
- Systémem přidaná kontrolní data – např. kontrolní součty v datech
- Kontrola systémového prostředí po skončení aplikace – jsou zrušeny pracovní soubory, jsou uzavřeny všechny procesy?



Testování aplikace jako bílé skříňky

- Ke každému programu další program jako testovací
- Testování všech větví programu
- Chyby a odpovídající hlášení
- Fungování podle dokumentace
- Správná Hlášení systémových omezení – vyčerpána paměť

Testy řízený vývoj aplikací

- Nejprve test, pak teprve program
- Často objektově orientovaný vývoj
- Nejprve test prázdného objektu, pak teprve implementace metod

Distribuce Verze 1.0

- Uzavřeno testování jako bílé skříňky
  - programátoři
- Uzavřeno testování jako šedé skříňky
  - tester
- Uzavřeno testování jako černé skříňky
  - zadavatel požadavků – případů užití
- Finální sestavení a distribuce Verze 1.0 zákazníkovi
  - pro Windows soubor .msi – Microsoft Installer nyní Windows Installer
- Zahájení iterací pro Verzi 2.0