SOFTWAROVÉ INŽENÝRSTVÍ

SI

Řízení IT projektů

Ing. Ondřej Macek

2013/14

ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

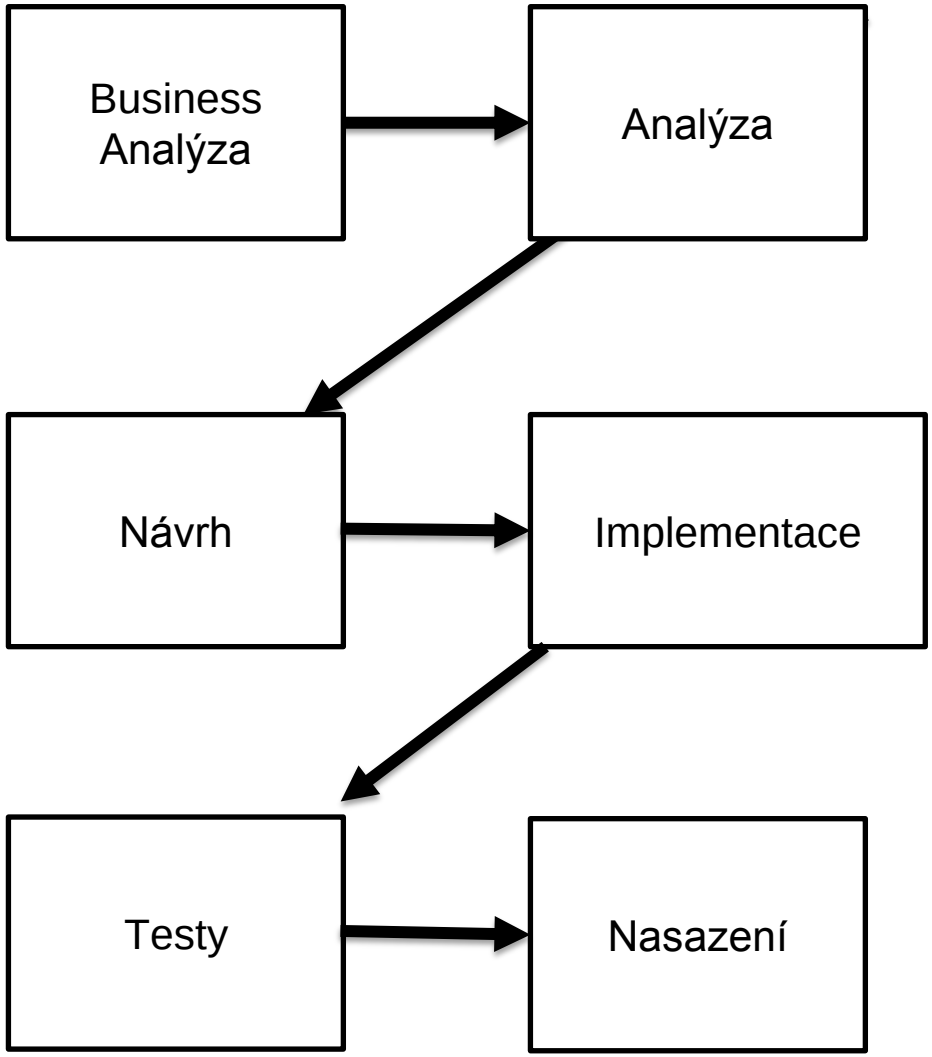
Jak vypadal vývoj SW?

- Bylo třeba specifikovat zadání, to se naprogramovalo a pak se dostal výsledek – podobně jako např. ve strojírenství
- Zákazník nerozuměl tvorbě programů (magie) – dnes si všichni myslí, že rozumí
- Přelom 81 – první PC, takže bylo třeba začít myslet jinak. Jde to těžko!
- Někdy kolem 78 byla konference NATO – o softwarovém inženýrství – projekty často chybují, nejsou včas, ...

- V roce 2000 bylo 125 aplikací pro řízení projektů a 25 technik – a to ještě nebyli agilní metodiky

Tradiční řízení

Lineární přístup k řízení projektů



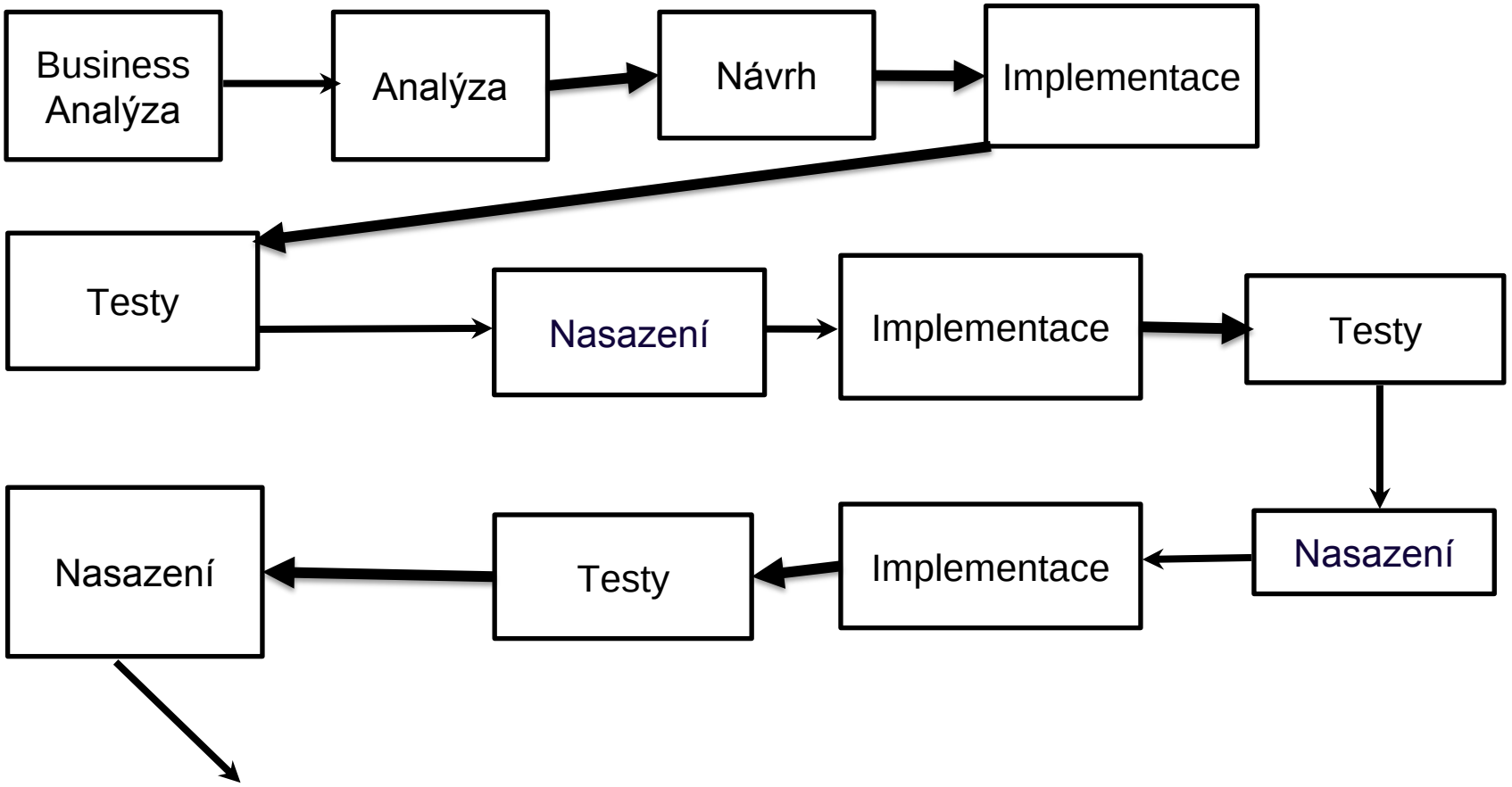
LPMLC

- Silné stránky?
  - Už od začátku je jasný plán => víme co bude
  - snadné řízení zdrojů (paralelismus projektů)
  - Zvládne to i začátečník/průměrný zaměstnanec (rozdíl oproti agilním)
  - Není třeba aby tým byl na jednom místě – analýza u nás, implementace v Malajsii; dá se snadno předat externímu dodavateli

- Slabé stránky?
  - Špatně reaguje na změnu
  - Stojí hodně peněz – veškeré chyby se projeví až při předání (akceptačním testu) a pak je drahé chyby opravit
  - Dlouho trvá než zákazník vidí nějaký výstup
  - Vyžaduje detailní plán a specifikaci – každé přeplánování je drahé
  - Sleduje procesy, které není možné změnit
  - Není zaměřen na přínos klientovi – je zaměřen na dodání v nějaký čas – dodání podle specifikace není často totožné s tím co zákazníkovi pomůže – potřeby vs.přání

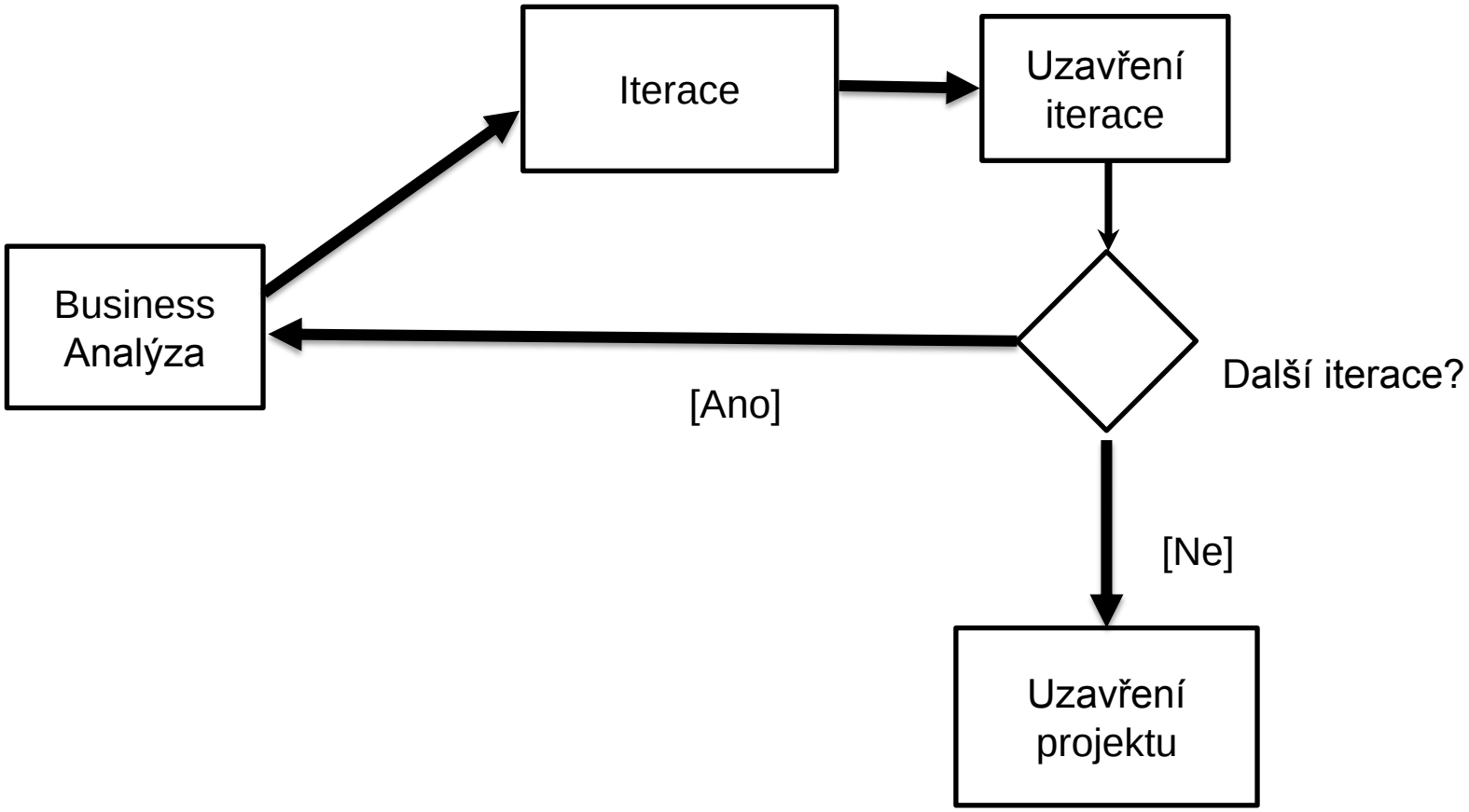
Pokročilejší přístupy

Inkrementální přístup

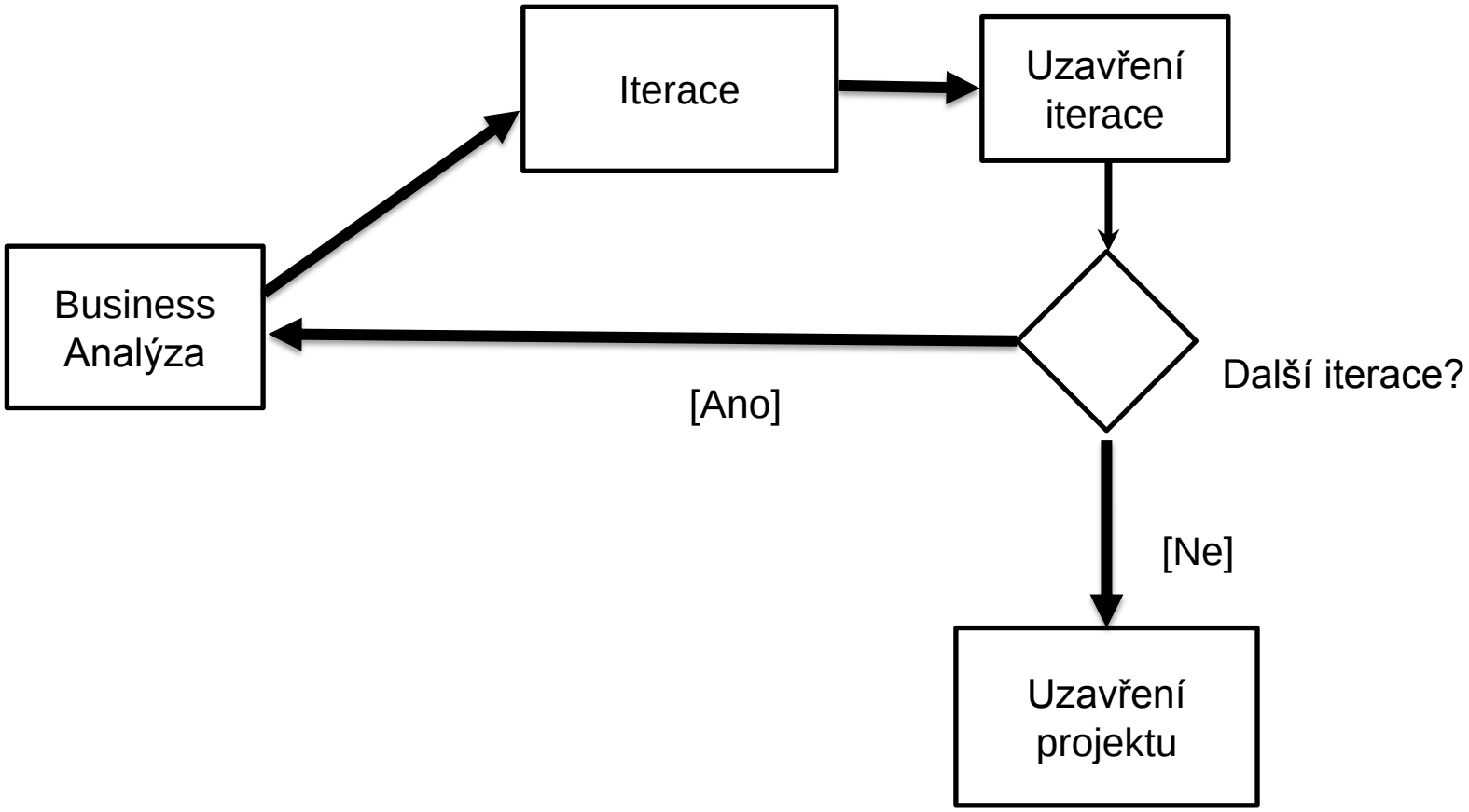



Iterační řízení

Iterační přístup



Co je iterace?



Iterační přístup

Charakter
- Častý sběr požadavků na změnu, ale známe řešení
- Tým v jedné lokalitě
- Iterace 2-4 týdny
- Část řešení není známa – nejsou detaily -> změny
- High level scope/detailní scope pro každou iteraci
- Zpětná vazba s klientem
- Jednoduché/lehké zapojení klienta

Nevýhody
- Zapojení klienta
- Týmy v jedné lokalitě
- Není vždy jasné jak to dopadne

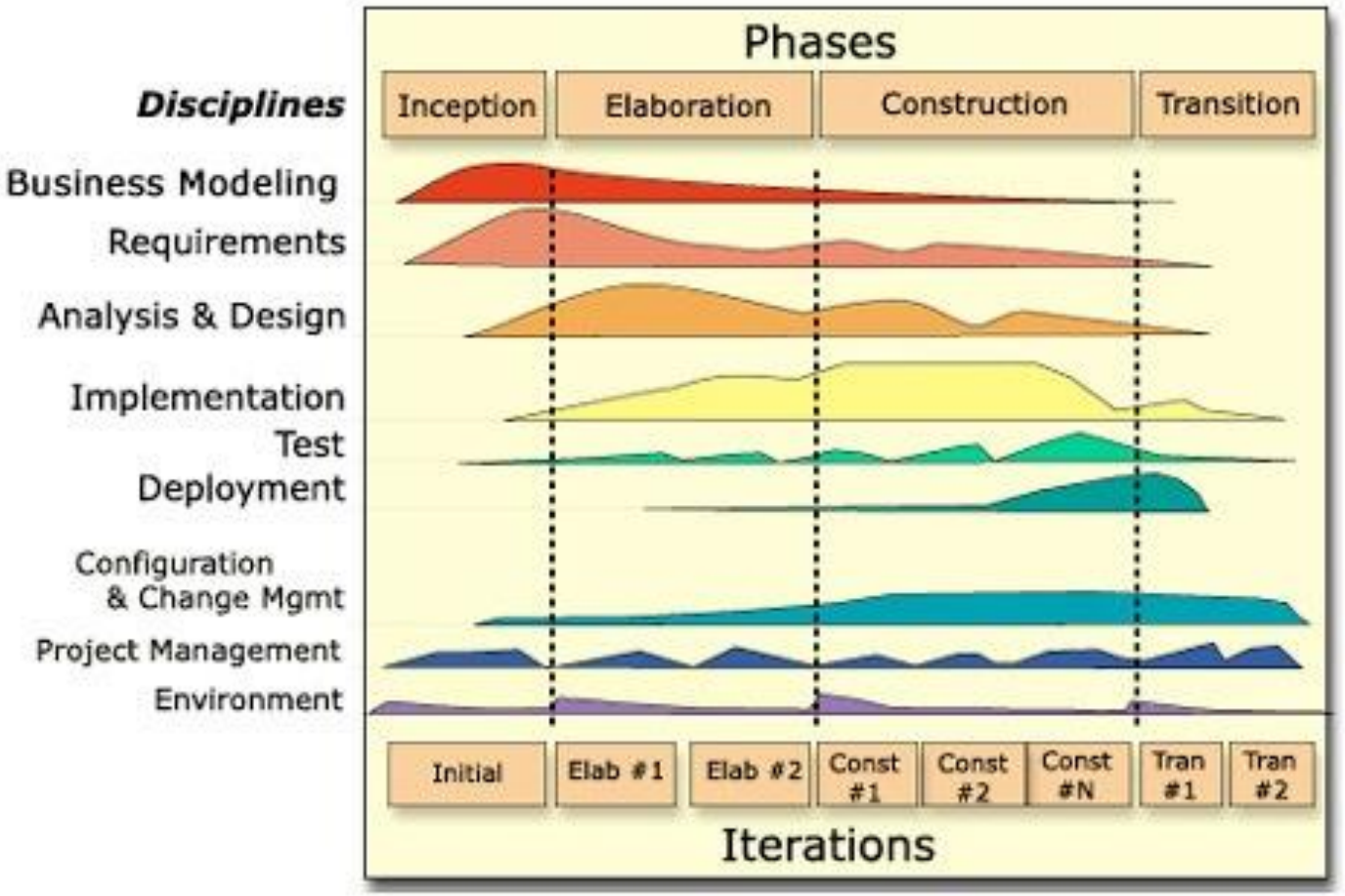
Potřeby
- Malý tým
- Zkušení pracovníci


Unified Process

Unified Process - Životní cyklus

- Postupuje ve fázích
- Fáze se skládá z iterací
- Po dokončení každé iterace/fáze se provede zhodnocení
- Další iterace/fáze projektu začne jen v případě splnění kritérií.

FÁZE RUP


Phases
Disciplines
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration & Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

ZAHÁJENÍ (INCEPCE)

- identifikace problému, který má být vyřešen
- formulace čeho má být dosaženo – specifikace zadání
- je třeba porozumět doméně
- určení priorit

Vstupy

Stávající systém, úvodní požadavky na systém,…

Výstupy

Vize, Plán, Základní požadavky, (Prototyp)

Milník

Dohoda mezi zákazníkem a dodavatelem na rozsahu, ceně a harmonogramu

PŘÍPRAVA (ELABOACE)

- sestavení projekčního týmu
- nastavení prostředí pro vývoj
- zpřesňování představy/požadavků
- základ architektury a prototypu
- plán projektu

Výstup

Prototyp, Use case (80%), Návrh(10%), Datový model, Plán testů, Plán projektu,

Milník

Architektura, Use case, Plán testů, Podrobnější plán projektu

- dokončení návrhu
- první verze SW
- návrh testů
- realizace testů
- vznik dokumentace (návrhové modely, model nasazení…)

Výstupy

První verze, testy, uživatelská dokumentace

Milník

Funkční verze

PŘEDÁNÍ (TRANSITION)

- předání finální verze koncovým uživatelům
- beta testování systému
- nemělo by docházet k zásadním změnám funkcionality
- řeší se instalace, konfigurace a odlaďování

Výstupy

Konečná funkční verze splňující všechny požadavky zadání, Manuály

Milník

Předání

Unified process

- Vyžaduje konzultace s klienty
- Základem analýzy a implementace jsou use-cases
- Na architekturu je třeba zaměřit se co nejdříve
- Testy probíhají na všech úrovních


AGILNÍ PRINCIPY

Agilní přístup - motivace

- Metodiky vývoje SW jsou nevhodné
- Špatná komunikace vede ke krachu projektu
- Na testy už nezbývá čas
- Vytváření „zbytečnou“ dokumentaci

Agilní přístup - motivace

- Spokojený zákazník
- Spokojený programátor

Agilní

- způsob řízení týmu
- způsob vývoje SW
- způsob myšlení

Agilní manifest

- **Člověk a komunikace** je nad **procesy a nástroje**

Agilní manifest

- **Člověk a komunikace** je nad **procesy a nástroje**
- **Fungující software** je nad **vyčerpávající dokumentaci**

Agilní manifest

- **Člověk a komunikace** je nad **procesy a nástroje**
- **Fungující software** je nad **vyčerpávající dokumentaci**
- **Spolupráce se zákazníkem** je nad **přesně sjednanou smlouvou**

Agilní manifest

- **Člověk a komunikace** je nad **procesy a nástroje**
- **Fungující software** je nad **vyčerpávající dokumentaci**
- **Spolupráce se zákazníkem** je nad **přesně sjednanou smlouvou**
- **Reakce na změnu** je nad **přesné dodržování plánu (**Ale zas má agile pravidlo – nezasahovat do iterace. Proto krátké iterace)

Role projektového manažera

Extrémní programování XP

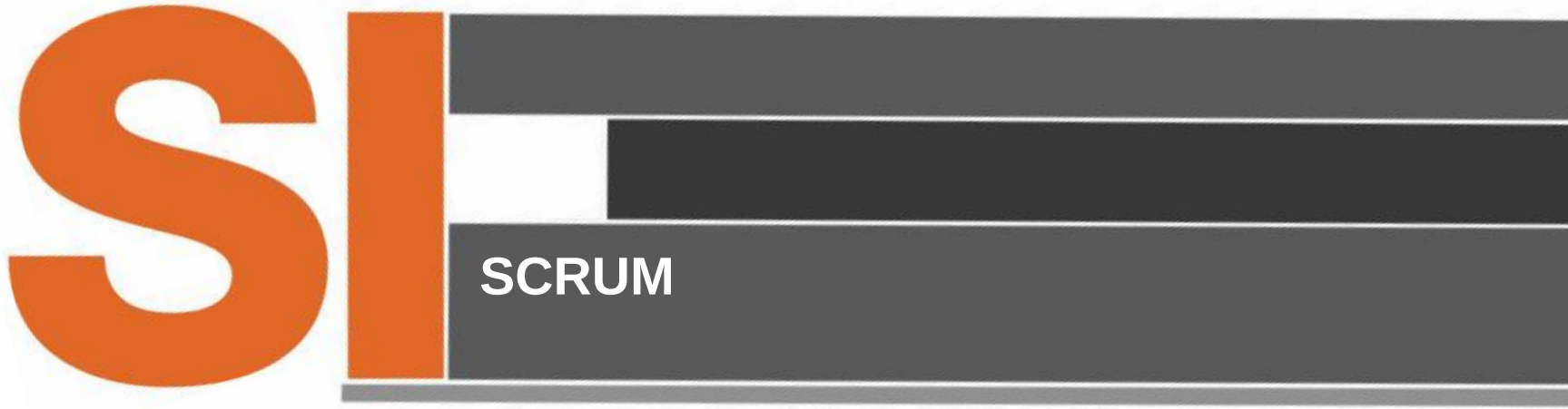
- Projekt se skládá z **iterací**
- Plánování je zaměřeno na **časté dodání malých releasů** (iterace = release)
- Každý tým má vyčleněný **vlastní pracovní prost**
- Pracovní den začíná **stand up meetingem**
- Reporting sleduje **rychlost projektu**
- Popis funkce **pomocí User stories**

- Nástěnka/tabule – základní prostředek motivace/komunikace/modelování
- Refactoring – změna kódu- bez změny chování programu; zvyšování čitelnosti, snižování složitosti =>zlepšení udržovatelnosti; zlepšení rozšiřitelnosti
- Reverse engineering - hlavně za účelem vytvoření dokumentace, protože dokumentace neexistuje
- Buildy – ověření funkčnosti ⇔ prototypování

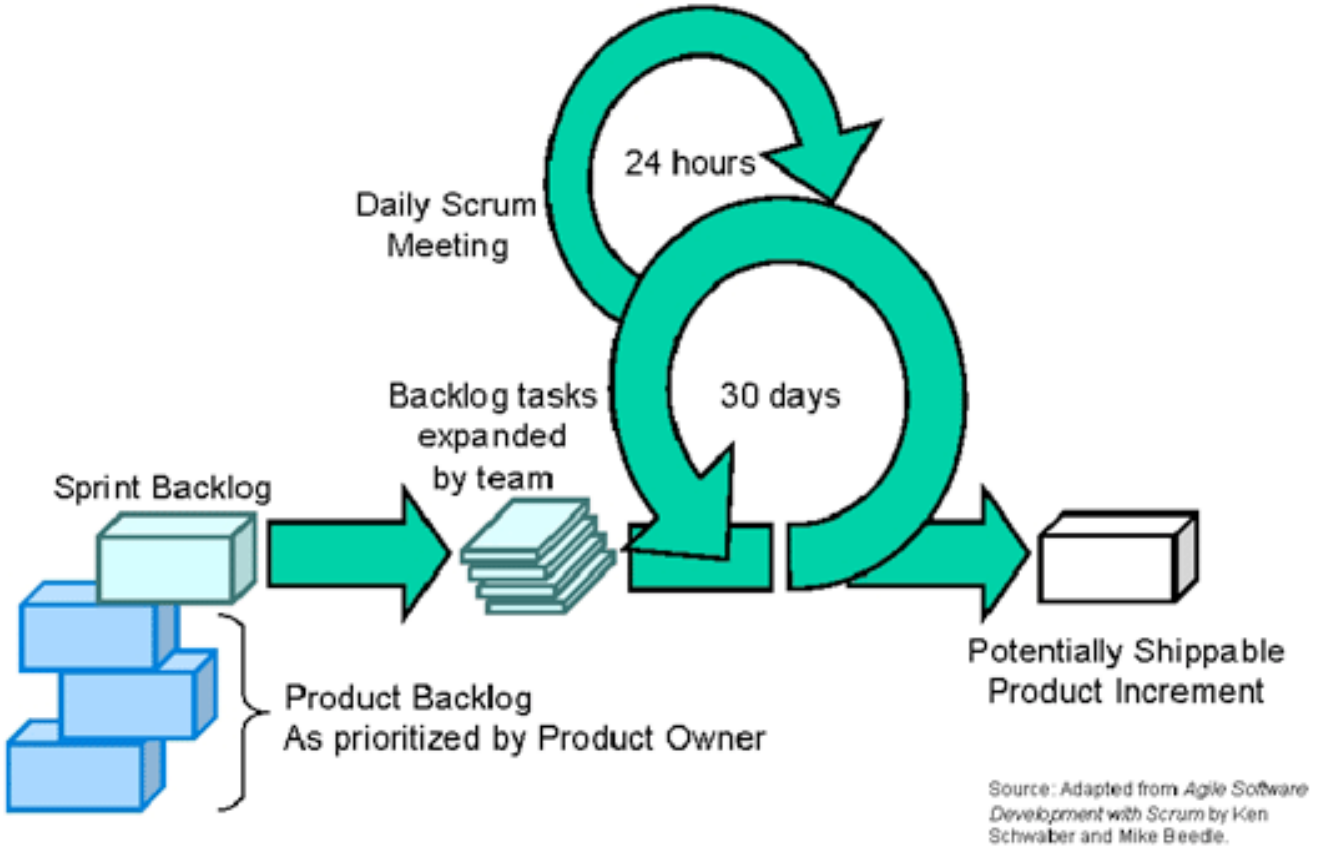
XP

- Návrh je **jednoduchý**
- Použití **metafor k popisu funkcí**
- **Vydávání malých verzí**
- Zaměření na **okamžitou přidanou hodnotu**
- **Refactoring**
- **Udržitelné tempo**
- **Zákazník je vždy k dispozici**
- **Coding standards**
- **Test driven development**
- **Pair programming**
- **Continuous integration**
- **Sdílený kód**

- **Pro každý kód existuje unit test**
- **Před každým release** musí projít **všechny testy**
- Pokud je objeven **bug => je napsán unit** a akceptační test
- Akceptační testy probíhají často a jejich úspěšnost je zveřejňována


SCRUM

Životní cyklus



Requirements are allowed to change (and change is encouraged) **but only outside the sprint**. **Once the team starts on a sprint, it remains maniacally focused on the goal** of that sprint.

SCRUM
dokumenty/výstupy

- Sprint/Product Backlog
- Burndown charts
- Increment

Zúčastněné strany

- Product owner
- Scrum master
- Team

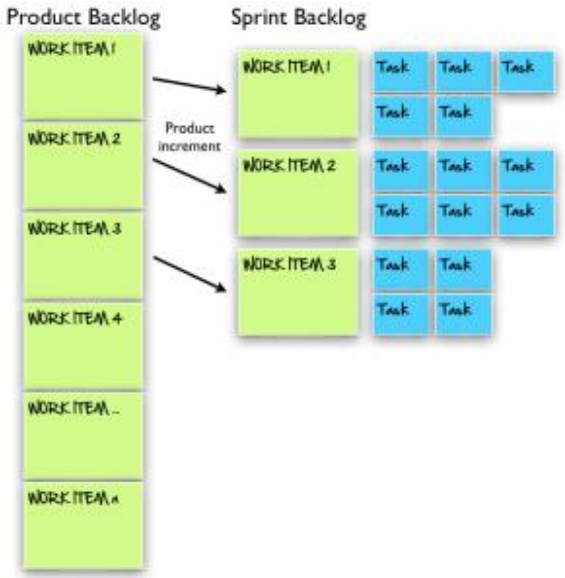
Project/Sprint Backlog

Project

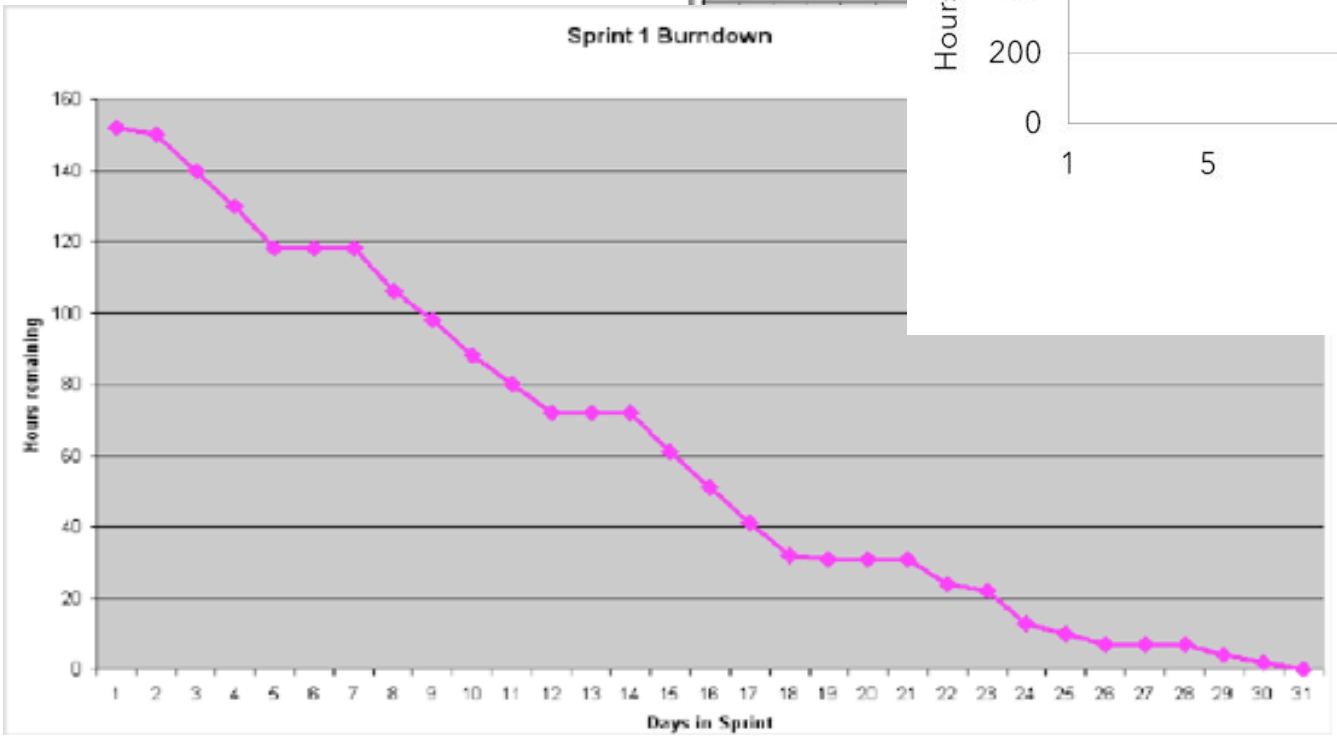
Sprint	Name	Estimate (days)	Remaining (days)
1	Manažer zadává úkoly zaměstnancům	1	1
	Zaměstnanec předá úkol pro otestování	2	2

Sprint

Name	Estimate (hours)	1. 12.	2. 12.	3.12.
Manažer zadává úkoly zaměstnancům	8	7	3	0



Burndown graph

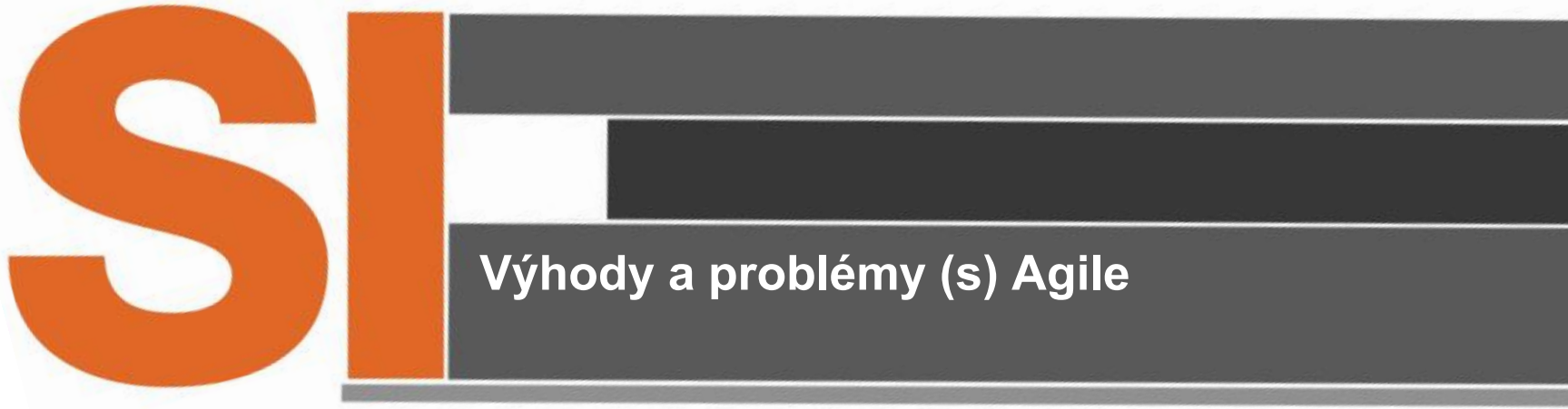


SCRUM Meetings

- Planning meeting
- Daily meetings
- Sprint review
- Sprint retrospective meeting

SCRUM „done“

- Kompletní funkčnost
- Kompletní testy
- Připraveno na nasazení



Výhody a problémy (s) Agile

- Vize
- Týmový práce & spolupráce
- Jednoduchá pravidla
- Sdílení informací
- Lehké řízení
- Učení se

Kde se Agilní přístup doporučuje

- Zkušení programátoři
- Očekává se hodně změn
- Malý tým

Co Agilní přístup potřebuje

co je doporučováno

- Zkušené programátory
- Někoho s předchozí zkušeností s Agile
- Komunikační schopnosti

Kde se Agilní přístup nedoporučuje

- Velké programátorské týmy
- Dislokované týmy
- Nutnost fixace projektového trojúhelníku

Co Agilní programování není

- Anarchie
- Ignorování dokumentace
- Cowboy programming
- Striktní sada pravidel

Problémy s aplikací agilního přístupu

- Strach
- Neznalost
- Firmy jsou řízeny procesně
- Manažeři
- Programátoři

Strach z neznámého, ze ztráty kontroly (ze strany firmy); strach z ostatních, pair programmingu, odpovědnosti atd. (ze strany vývojářů)

Neznalost – představa agile = chaos

Firmy jsou řízeny podle procesů a jsou i podle procesů hodnoceny (ISO, CMM) – jak do toho zapojit agile?

Manažeři jsou nastaveni na procesní řízení (naučili je to), chtějí/jsou zvyklí rozhodovat, u agile by i manažer měl být kodér (z 50%)

Programátoři
– individuality: mají představu „nepostradatelnosit“ nepřijímají kritiku, nechtějí spolupracovat
- asociálové: nechtějí pracovat v týmu, bojí se lidí/ukáže se, že že nejsou dobří apod.

Problémy s aplikací agilního přístupu

- Aplikace jen některých doporučení
- Nepřítomnost zákazníka
- Spoléhá na ústní podání
- Přečtení knížky ≠ zavedení do praxe
- Malý důraz na architekturu v počátečních fázích

Doporučené čtení

- http://soch.cz/blog/
- https://docs.google.com/spreadsheet/ccc?key=0Ag91X5gS0piQcDhvTUw1QzlNLS11d29mbWhsMW5wZXc#gid=0
- http://www.extremeprogramming.org/
- http://www.scrumalliance.org/
- http://www.methodsandtools.com/archive/archive.php?id=18