XINF1

Jaroslav Žáček
jaroslav.zacek@osu.cz

Tutoriály

* 24.9.

* 15.10.

* Na tutoriálech stihneme vysvětlit věci, které nejsou jasné ze skript a zaměříme se aplikaci znalosti pro předmět XRPR1

* Přečtěte si skripta dříve, než týden před zkouškou.

* Pište na e-mail kdykoli něco nebude jasné, nečekejte až na tutoriál.

* Slajdy z přednášek jsou dostupné na osobních stránkách.

První námět k přemýšlení

* Co budete vyvíjet za informační systém v předmětu XRPR1

* Bude tento systém k něčemu dobrý?

* S kým budete vyvíjet nový informační systém?

Architektura IS

* Klíčový prvek řízení IS – z něj vycházejí detailní analytické i plánovací charakteristiky celého IS. Architektura musí respektovat strategii podniku, podnikové cíle a cíle IS.

* Musí být jednoduchá a srozumitelná, je to jakýsi skelet, na který se navěšují další funkce systému.

* 3 vrstvy v architektuře IS:

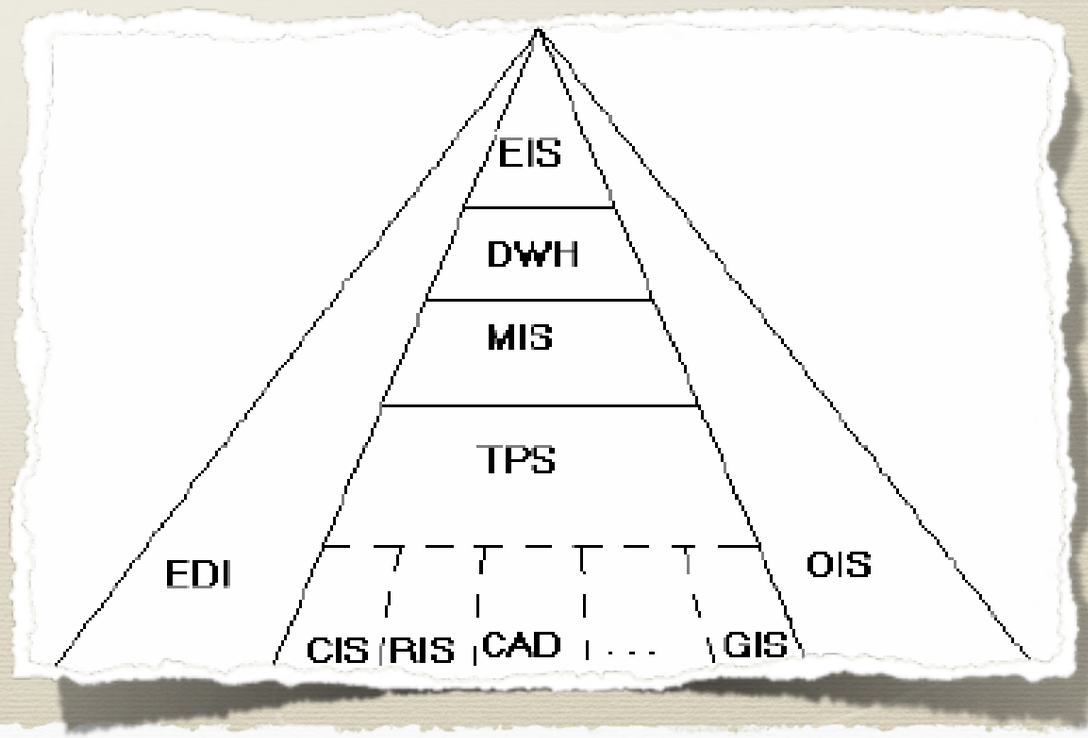
    * Vrstva prostředí – ekonomické prostředí, legislativa, organizační struktura, personální kapacity a jejich kvalifikace.

    * Vrstva aplikační – provozované a řešené projekty, jejich dokumentace, funkční a datové specifikace, organizační pravidla jejich řešení a provozu, aplikační SW.

    * Vrstva technologická – návrh a provoz počítačových sítí, vymezení jednotlivých komponent IT (ZSW, technické prostředky včetně jejich vazeb a vnitřní struktury).

Globální

- EIS (Executive IS) – podpora vrcholového řízeníDWH (Data warehouse) – podpora řízení na základě analýz rozsáhlých dat.
- MIS (Management IS) – podpora taktické a operativní úrovně řízení (účetnictví, nákup, prodej, sklad, ...).
- TPS (Transaction processing system) – spojený s typem provozu v rámci dané organizace (dílenské, skladové, podniků, rezervační systémy dopravních společností).
- CIS (Customer IS) – zajišťuje bezprostřední styk se zákazníkem.
- GIS (Geographic IS)
- CAD (Computer aided design),
- CAM (Comp aided manufacturing)
- OIS (Office IS),
- EDI (Electronic data interchange)



Dílčí

* Funkční - funkční struktura
* Procesní - zachycuje procesy, diagramy toků dat, síťový diagram
* Datová - interní a externí, návrh entit, datových souborů
* Softwarová - ASW, ZSW, systémový SW
* Technická - všechny prostředky ICT
* Organizační - organizační struktura a organizační jednotky
* Personální - profesní struktury

Informační systém

* Informační systém organizace je systém informačních technologií, dat a lidí, jehož cílem je efektivní podpora informačních a rozhodovacích procesů na všech úrovních řízení organizace (firmy). Vývoj a provoz IS jsou ovlivňovány řadou aspektů.
* Podporují podnikové procesy organizace:
    * objednávky,
    * fakturace,
    * nákup, prodej,
    * skladové hospodářství

Metodika

* Projektování software – proces tvorby nového SW a jeho uvedení do provozu. Proces je řízen a má určitá pravidla a doporučení, kterými se při vývoji řídíme => metodiku

* Metodika říká kdo, kdy, co a proč má dělat během vývoje a provozu SW.

* Metodika je doporučený souhrn principů, konceptů, dokumentů, metod, technik a nástrojů pro tvůrce softwarových (informačních) systémů, který pokrývá celý životní cyklus informačních systémů

    * Např. RUP, DSDM, Jackson Structured programming

Metoda, technika, nástroj

* Metoda říká, co je třeba dělat v určité fázi nebo činnosti vývoje a provozu.
    * např. testování software
* Technika – určuje, jak dělat danou činnost, vymezuje pro činnost přesná pravidla. Předurčuje způsob uvažování a vyjadřování, často spjata s konkrétním nástrojem.
    * OOP, datové modelování, vytváření prototypů
* Nástroj – prostředek k uskutečnění určité činnosti v procesu vývoje a provozu SW.
    * CASE nástroje, automatizované testování

Procesní framework

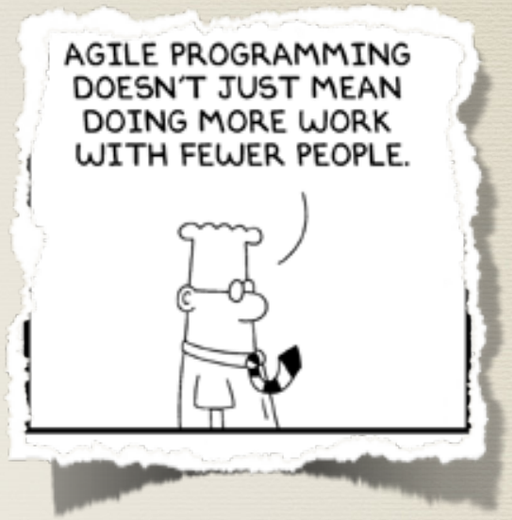
* Není to konkrétní metodika!
* Předdefinovaná množina rolí, artefaktů, aktivit
    * Např. dokumenty, modely,
* Vybere se jen to, co je pro projekt potřebné
* Vznikly většinou z minulých zkušeností
* RUP - framework pro tvorbu SW
* ITIL - framework pro provoz a údržbu stávajícího SW

Agilní metody

* The Agile Manifesto (http://agilemanifesto.org/)
* Jednotlivci a součinnost nad procesy a nástroji
* Fungující software nad komplexní dokumentací
* Spolupráce se zákazníkem nad vyjednávání o kontraktu
* Odpověď na změnu nad následováním plánu
* Klíčový bod: iterativní a inkrementální vývoj s prioritou (založeny na rizicích a přínosech pro byznys) úkolů!

Agilní metody

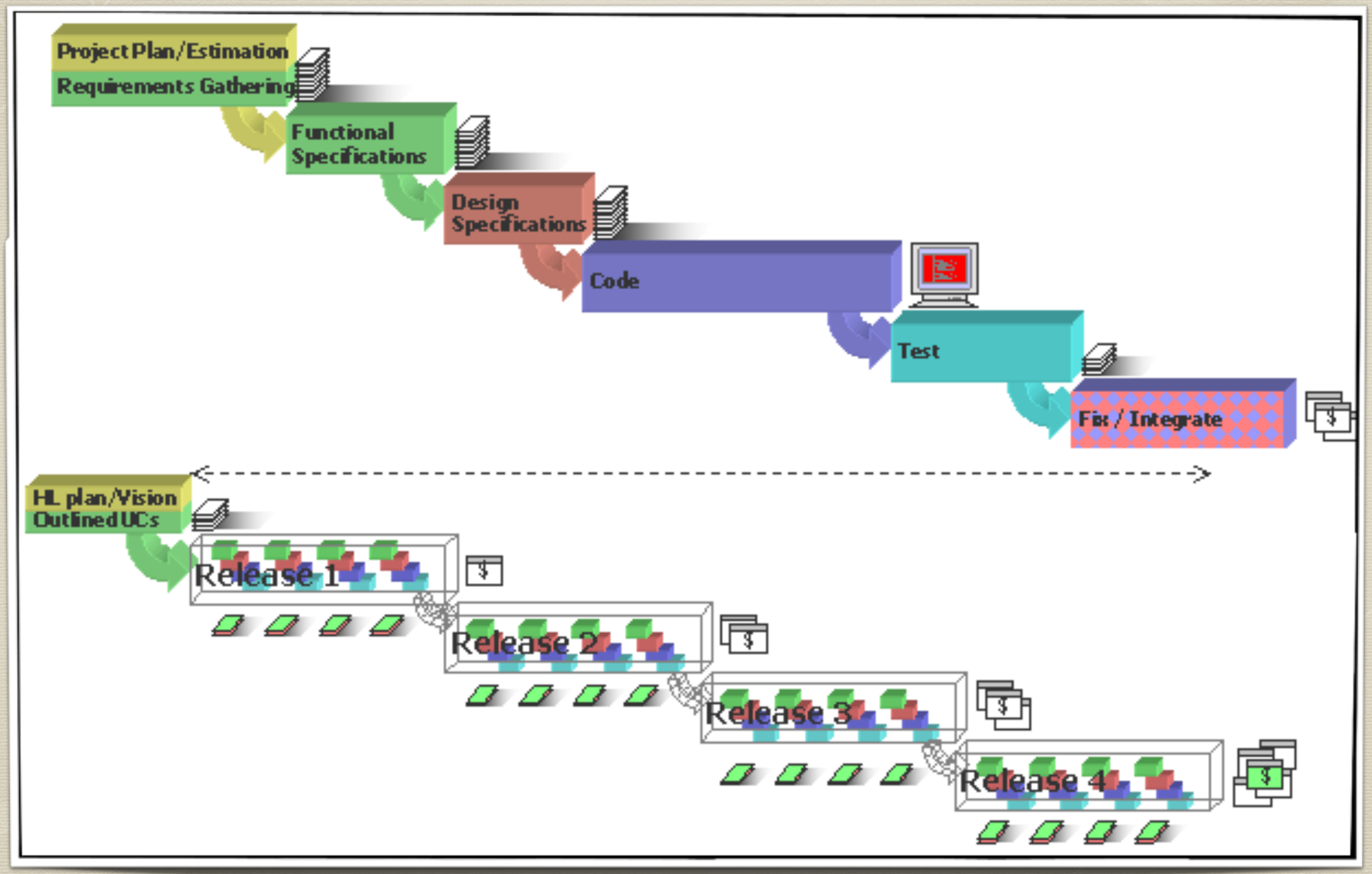
* Extreme Programming - XP
* Scrum
* Crystal Clear
* Lean Development (můžete znát z Toyoty)
* Feature-Driven Development (FDD)
* Test-Driven Development / Behavior-Driven Development
* ...



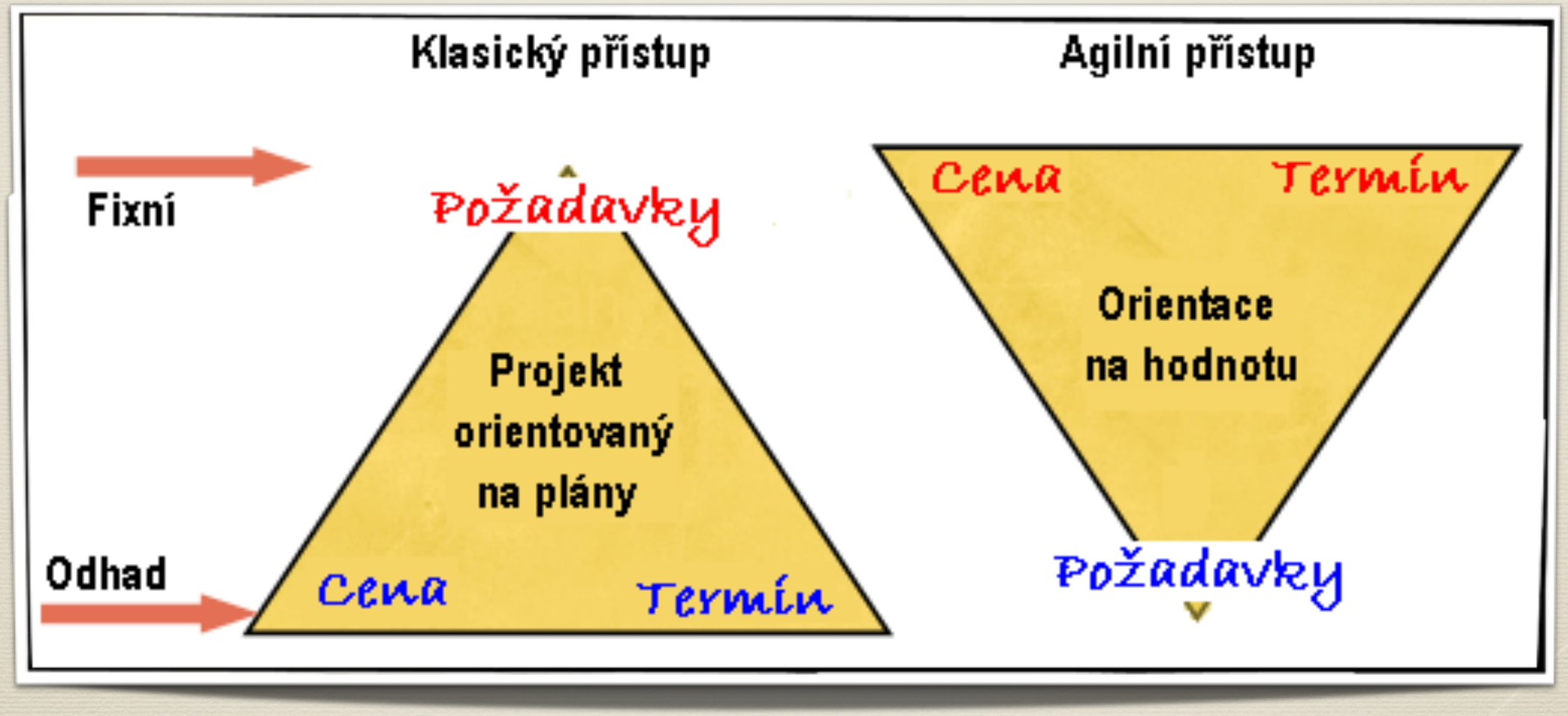
Vodopád vs. Agile

Srovnání

Vodopádové principy	**Iterativní (agilní principy)**
Zaměřen na procesy, předpokládá jejich opakovatelnost.	Zaměřen na lidi – motivace, komunikace prvořadá.
Pevné, podrobné plány definovány na úvod, kdy je spousta nejasností.	Pro celý projekt pouze road map. Detailní plány jen iterace (kratší úseky, 2 měsíce).
Rizika jsou často překvapení, přináší problémy.	Řízen riziky – nejrizikovější věci řešíme nejdříve.
Integrace a testování až na konci.	Průběžná integrace a testování.
Změny nejsou vítány.	Počítá se změnami, přijímá je.
Často zaměřen na tvorbu dokumentů bez přidané hodnoty a jejich revize.	Zaměřen na fungující SW (hodnota pro zákazníka).
Buildy a testy až na konci, často přeskočeno nefunkční testování.	Automatizované buildy a testy.
Za kvalitu odpovědní pouze testeři, QA manažeři nebo často nikdo.	Všichni (celý tým) odpovědní za kvalitu produktu.


Project Plan/Estimation
Requirements Gathering
Functional Specifications
Design Specifications
Code
Test
Fix / Integrate
HL plan/Vision
Outlined UCs
Release 1
Release 2
Release 3
Release 4

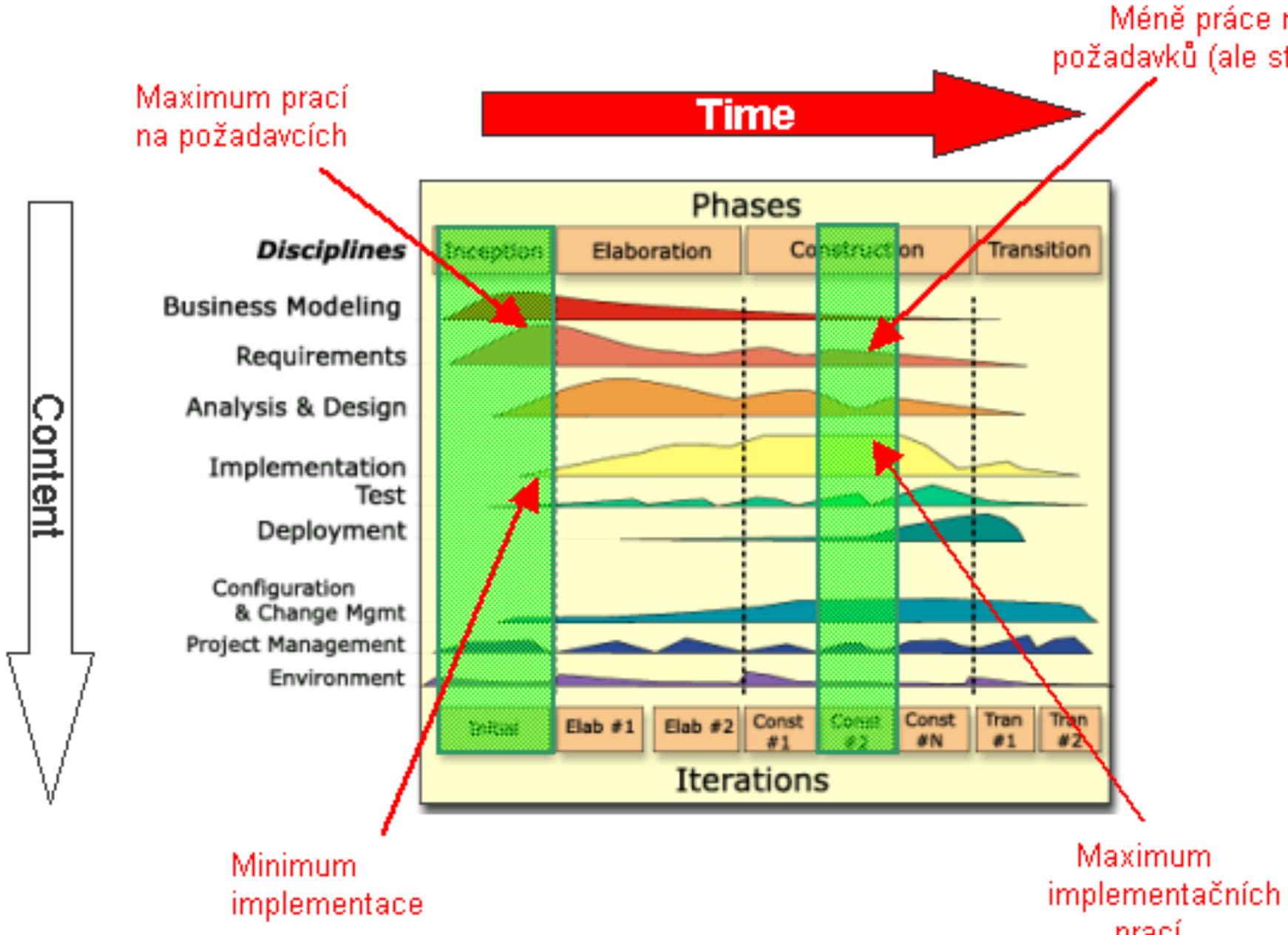
Železný trojúhelník



Rational Unified Process (RUP)

* Komplexní nástroj pro řízení vývoje IS
* UC driven (řízen pomocí UC) - UC je součástí vize projektu
* Zaměřen na rizika (nejrizikovější věci dělám nejdříve)
* Iterativní (každá iterace produkuje spustitelný a otestovaný build)
* Kooperace (analytik, designer, programátor, tester těsně spolupracují)
* Orientován na architekturu

RUP fáze a disciplíny



Iterace

* Každá iterace produkuje spustitelný a otestovaný build obsahující nově implementované funkčnosti (scénáře) – z důvodu zpětné vazby od uživatele

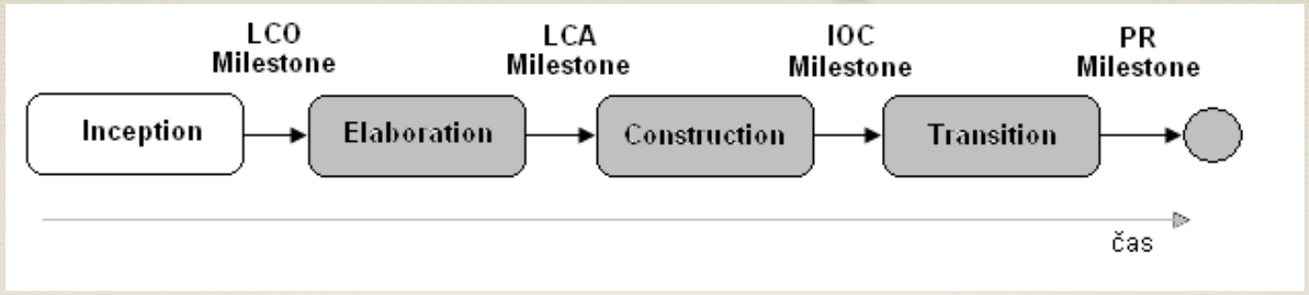
* Každá iterace má definovaný přesný cíl, který se snažíme naplnit (paralelní analýza, návrh, implementace a testování vybrané nové funkčnosti – jejich scénářů).

* Iterace je miniprojekt, má svůj začátek, konec a pevně definovaný časový rozsah (2-6 týdnů) – což se již předvídá a detailně plánuje lépe, než například 2 roky.

* V průběhu 1 iterace provádíme všechny disciplíny! Tj. definice požadavků, analýza a návrh, implementace, integrace a testování!!!

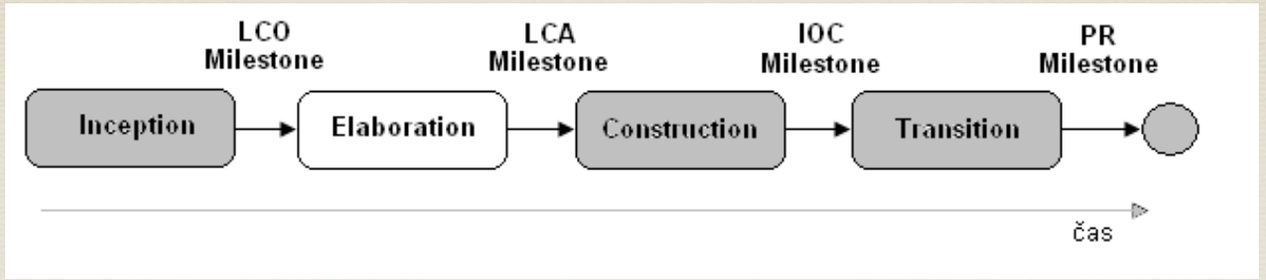
* Zřetězením iterací nabalujeme jednotlivé funkčnosti až do výsledného produktu.

Fáze Inception



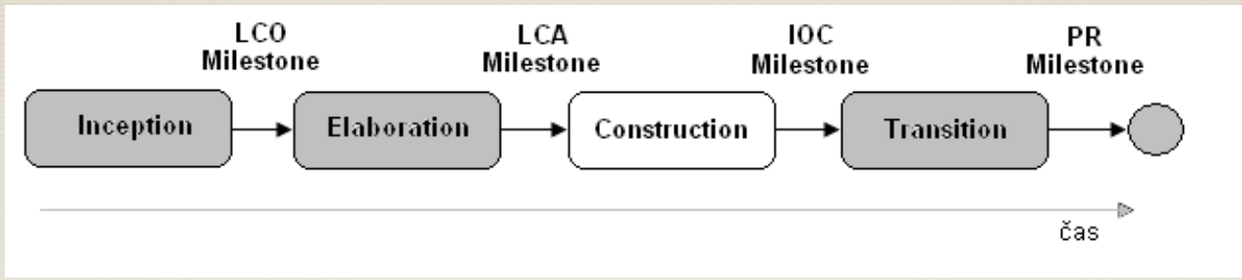
* Pochopení problematiky, nastínění funkčností systému (základní UC) – Vize
* Identifikace rizik + akce na jejich snížení / odstranění
* Návrh možného řešení (architektura)
* Složení týmu, financování
* Definice procesu, výběr a nastavení nástrojů

Fáze Elaboration



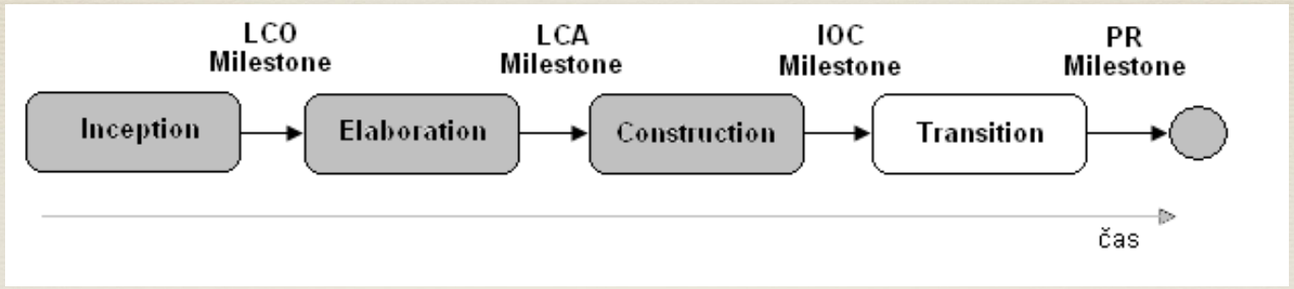
* Snížení technických rizik
* Návrh implementace a testování architektury – na základě klíčových potřeb (asi 20-30% UC)
* Výsledek fáze: stabilní, otestovaná architektura
* Architektura – rozhraní subsystémů, komunikační mechanismy, odchytávání výjimek, ukládání dat, ...

Fáze Construction



* Detailní analýza, návrh, implementace a testování zbývajících 70-80% UC v několika iteracích
* Využíváme mechanismy definované architekturou
* Výsledek fáze: beta-release
* Všechny disciplíny (analýza, návrh, vývoj, testování) jsou v každé iteraci prováděny paralelně

Fáze Transition



* Beta testování
* Školení (podpory, uživatelů)
* Příprava dat a prostředí (souběžný provoz verzí, transformace dat), instalace SW na klientské stanice
* Další aktivity (marketing, distribuce, prodej, CASE study, white papers, zprávy pro tisk)
* Lessons learnt - pro zlepšení procesu

Jak to udělat špatně

* Špatná komunikace mezi členy týmu
* Částečně udělaná práce
* Extra rysy aplikace (zákazník nikdy nepoužije)
* Znovu se učení
* Předávání práce, dokumentů, znalosti
* Přepínání mezi jednotlivými úkoly (context switching)
* Zpoždění (čekání na schválení, fronty)
* Defekty

Jak to udělat dobře

* TDD
* Iterative development (feedback, možnost zahrnout v průběhu vývoje změny)
* UC driven (reuse TC, tracebility)
* Continuous integration (automatické sestavení + automatický testing)
* Mále dokumentace + dokumentace návrhu kódem
* Implementace pouze potřebných požadavků (klíčové potřeby – 20% UC)
* Posunout rozhodnutí