Vývoj informačních systémů

Jak vyvíjet v týmu

2021-22

Co je potřeba a co je podstatné?

- Lidé a jejich spolupráce
- Plány, pravidla, procesy, řízení
- Dokumentace
- Techniky a technologie
- Dlouhý čas
- **Cílem je produkt (software) a jeho kvalita měřená více faktory**

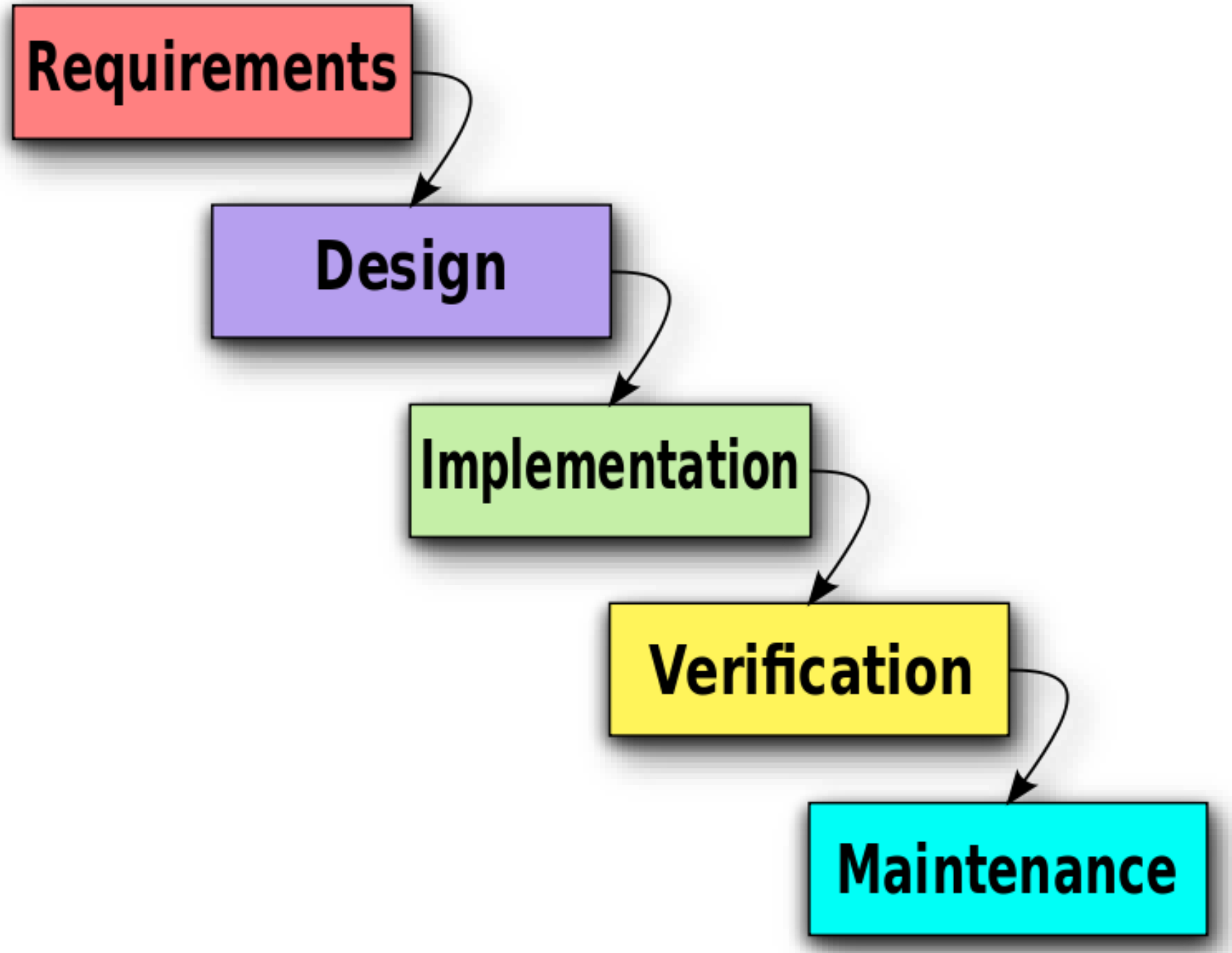
Od zadání k produktu...

- Základem jsou lidé, jejich kvalita a kvalita jejich výstupů.
- Je nutné umět organizovat práci a koordinovat lidi v týmu. *Je otázkou, s jakou mírou administrativní zátěže.*
- Je nutná dokumentace, kdo nedokumentuje, neváží si vlastní práce. *Je otázkou, v jakém množství.*
- Je nutný plán (čas a náklady). *Je otázkou, zda je možné odhadnout všechno dopředu.*
- Jsou nutná dobrá rozhodnutí na začátku (technologie, architektura). *Je otázkou, zdá máme vždy dostatek informací.*

Vodopádový model (1970)

- Původně sedm navazujících (nepřekrývajících se) fází: *Specifikace požadavků, Návrh, Implementace, Integrace, Testování, Ladění, Instalace, Údržba*.

- Vodopádový model vyžaduje, aby se k následující fázi přikročilo pouze tehdy, pokud je ta předcházející kompletní a perfektně připravená.


Requirements
Design
Implementation
Verification
Maintenance

Vlastnosti modelu

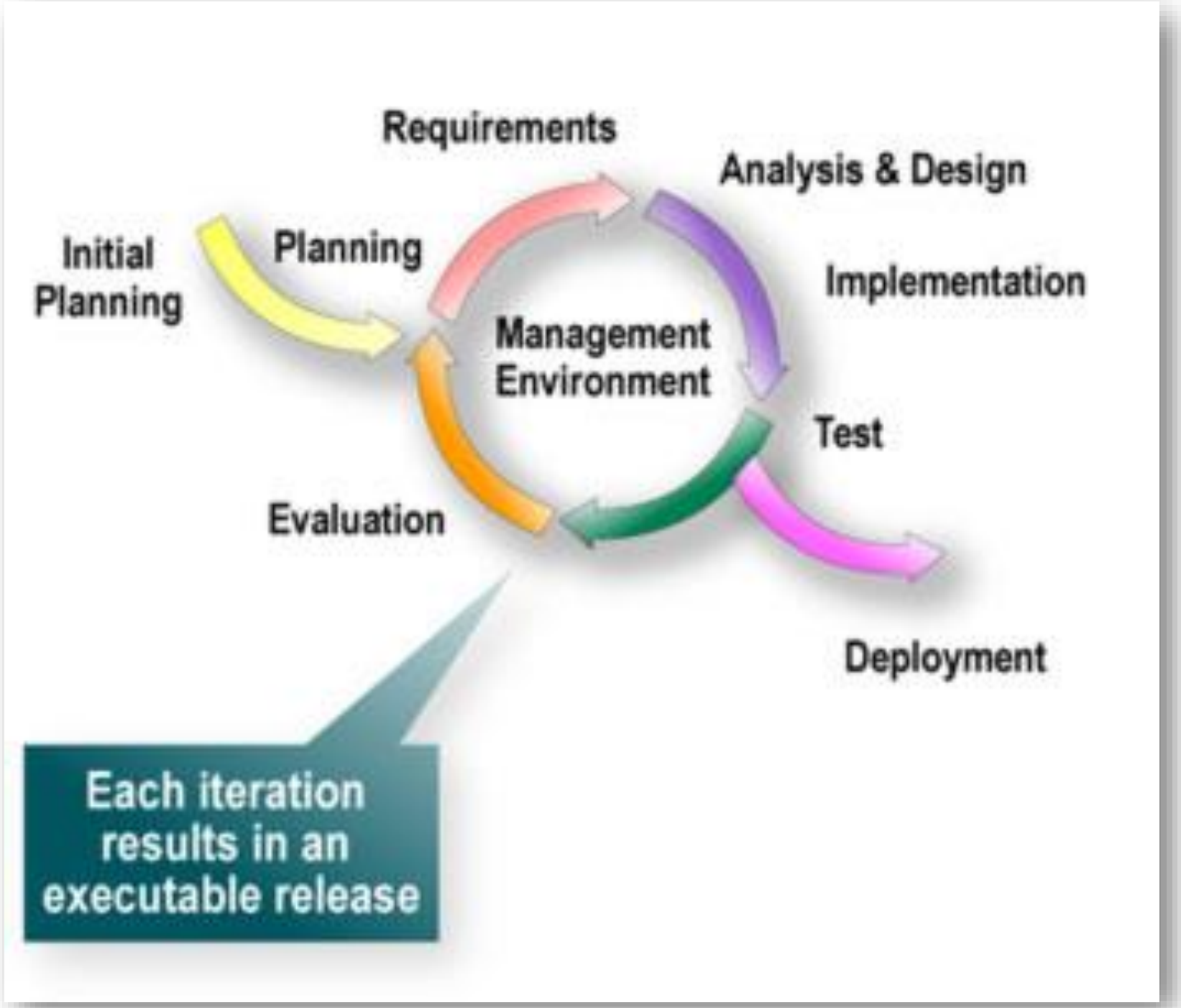
- Výhody (dvakrát měř a jednou řež)
  - Včasné odhalení chyb (vede k úsporám).
  - Vysoký důraz na dokumentování (bezproblémová záměna lidí).
  - Jednoduchost pro řízení (stabilita projektu).
- Nevýhody
  - Je nemožné dovést jednu fázi životního cyklu softwarového produktu k dokonalosti předtím, než se přejde k další fázi.
  - Trvá se na rozhodnutích, která se mohou později ukázat jako nesprávná.

Aspekty práce v týmu

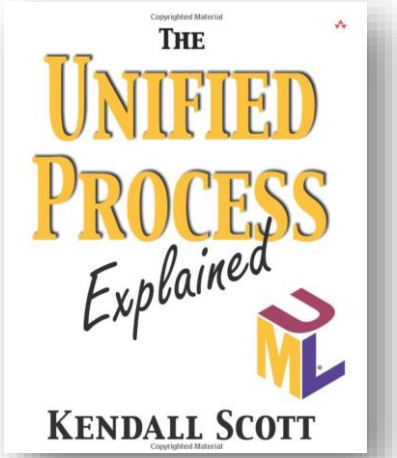
- Stačí dva a už se musí na lecčems dohodnout.
- Je rozdíl pracovat ve velkém a malém týmu.
- Robustní x agilní (čilý, aktivní, horlivý) přístup k projektu.
- Vždy musíme definovat role a odpovědnosti, ze kterých vyplývá způsob řešení úkolů.

Iterativní a inkrementální vývoj

- Základní princip odlišující současné přístupy od vodopádového modelu.
- *Iterativní návrh* je metodologie založená na opakovaném procesu analýzy, návrhu, implementace (prototypování), testování a redefinici produktu.
- *Přírůstkový (inkrementální) model* je založen na principu postupně budovaného produktu po přírůstcích založených na iterativním návrhu.
- Vývoj kombinuje vlastnosti vodopádového modelu s vlastnostmi iterativního prototypování.


Requirements
Analysis & Design
Initial Planning
Planning
Implementation
Management Environment
Test
Evaluation
Deployment
Each iteration results in an executable release

UP: Unified Process



- Ivar Jacobson (1999, Unified Software Development Process).
- Dva klíčové principy: Iterative and incremental development.
- Tři klíčové charakteristiky: Use-case-driven, Architecture-centric, Risk-focused.
- Čtyři fáze: Inception, Elaboration, Construction, Transition.
- UML, dokumenty.

Iterativní vývoj

- Iterativní vývoj rozděluje proces vývoje SW na menší části. Každá část, nazývaná *iterace*, představuje celý proces vývoje.
  - Každá iterace obsahuje kroky plánování, návrhu, vývoje a testování.
  - Na konci každé iterace je funkční produkt.
  - Od iterace k iteraci se přidává funkčnost.

Inkrementální vývoj

- Inkrementální vývoj je metoda, kdy jsou různé části systému vyvíjeny v různé době a různě rychle.
  - Části jsou spojovány v okamžiku dokončení.
  - Inkrementální vývoj souvisí s iterativním vývojem tak, že se vývojářské týmy mohou vracet k hotovým částem systému a mohou postupně zlepšovat jejich funkčnost.

Use case driven

- Spojení *use-case-driven* odkazuje na skutečnost, že projektový tým používá use cases (případy užití) ve všech fázích vývojových prací, od počátečního sběru požadavků až po tvorbu zdrojového kódu systému.
  - Seznam use case můžeme chápat jako specifikaci systému s níž se může porovnávat implementovaný systém nebo jeho část.
    - verifikace – testování funkčních požadavků (use-case je zdroj pro tzv. test-case).
    - validace – akceptování systému (jak funkční, tak technické požadavky).
  - Během vývoje může být jednotlivé use case upravovány a jejich seznam doplňován.

Architecture-centric

- Přístup k vývoji softwaru, v němž je popis architektury jádrem a ústředním bodem celého vývojového procesu.
  - Použití tohoto přístupu umožňuje jednak architekturu udržovat stabilní (minimálně na abstraktní úovni) a jednak ji během vývoje zpřesňovat (např. v závislosti na konkrétních technologických rozhodnutích).
  - V architecture-centric přístupu předpokládáme pro vývoj nového systému nejen vývoj nových komponent, ale také použití již vyvinutých komponent.

Risk-focused

- Vývojářský tým by se měl zaměřit na řešení nejkritičtějších rizik (nejistot) v rané fázi životního cyklu projektu.

- Výstupy jednotlivých iterací by měly být vybrány tak, aby bylo zajištěno, že největší rizika budou řešena jako první.

- Podcenění a odklad rizik a nejistot na později není dobrý nápad a může vést k neúspěchu projektu.

Inception fáze - zahájení

- Hlavním cílem inception fáze je prokázat realizovatelnost navrhovaného systému.
  - Definování rozsahu systému (tj. co do něj patří a co ne), definování klíčových požadavků.
  - Návrh první verze architektury.
  - Identifikace kritických rizik a určení, kdy a jak je bude projekt řešit
  - Zdůvodnění, že se projekt vyplatí realizovat a odhad nákladů.
  - První verze harmonogramu.

Elaboration fáze - rozpracování

- Hlavním cílem elaboration fáze je potvrdit schopnost vybudovat nový systém s ohledem na definovaný finanční a časový rámec.
  - Zachycení a rozpracování většiny funkčních požadavků.
  - Zpřesnění architektury na technickou úroveň.
  - Zpřesnění významných rizik a návrh jejich řešení.
  - Příprava plánu projektu, který obsahuje dostatečně podrobné informace pro další fázi projektu (konstrukce).

Construction fáze - konstrukce

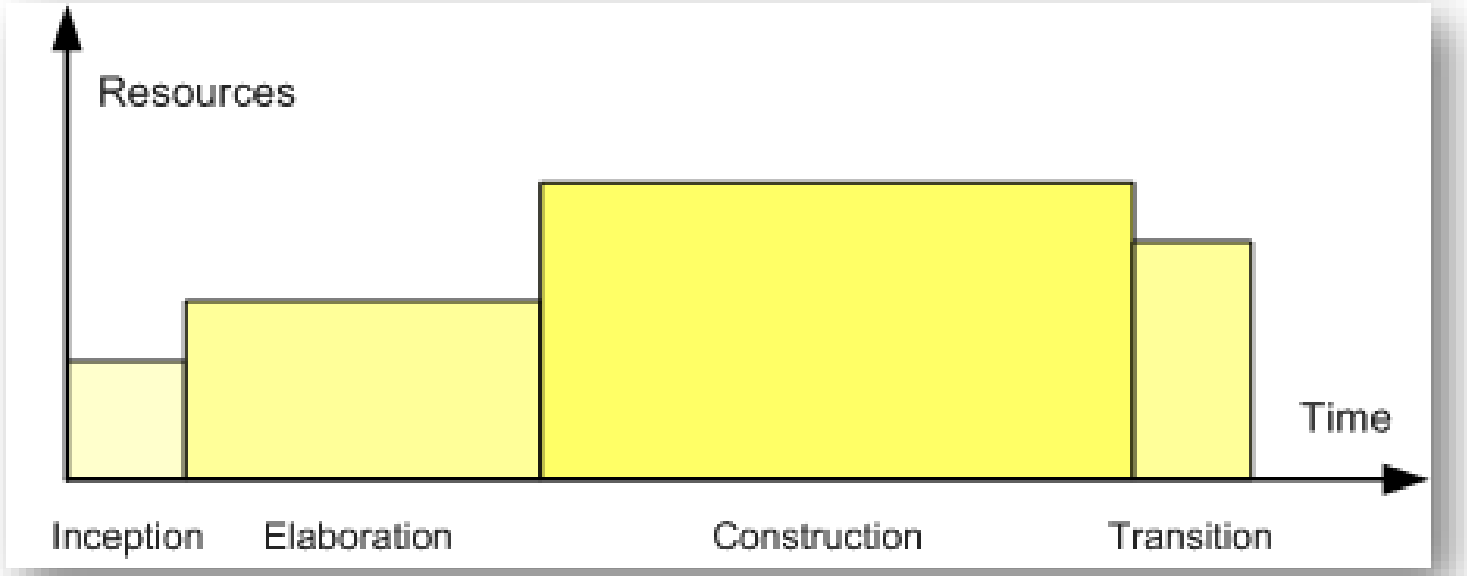
- Hlavním cílem construction fáze je vytvořit systém, který bude schopen úspěšně fungovat v beta prostředí u zákazníka.
  - Projektový tým plní úkoly, které jsou součástí iterativního a inkrementálního postupu budování systému.
  - Dílčí milníky jsou funkční části systému (přírustky) průběžně akceptované zákazníkem.
  - Hlavním milníkem fáze konstrukce je první (beta) verze systému nasazená do provozu v technickém prostředí zákazníka.

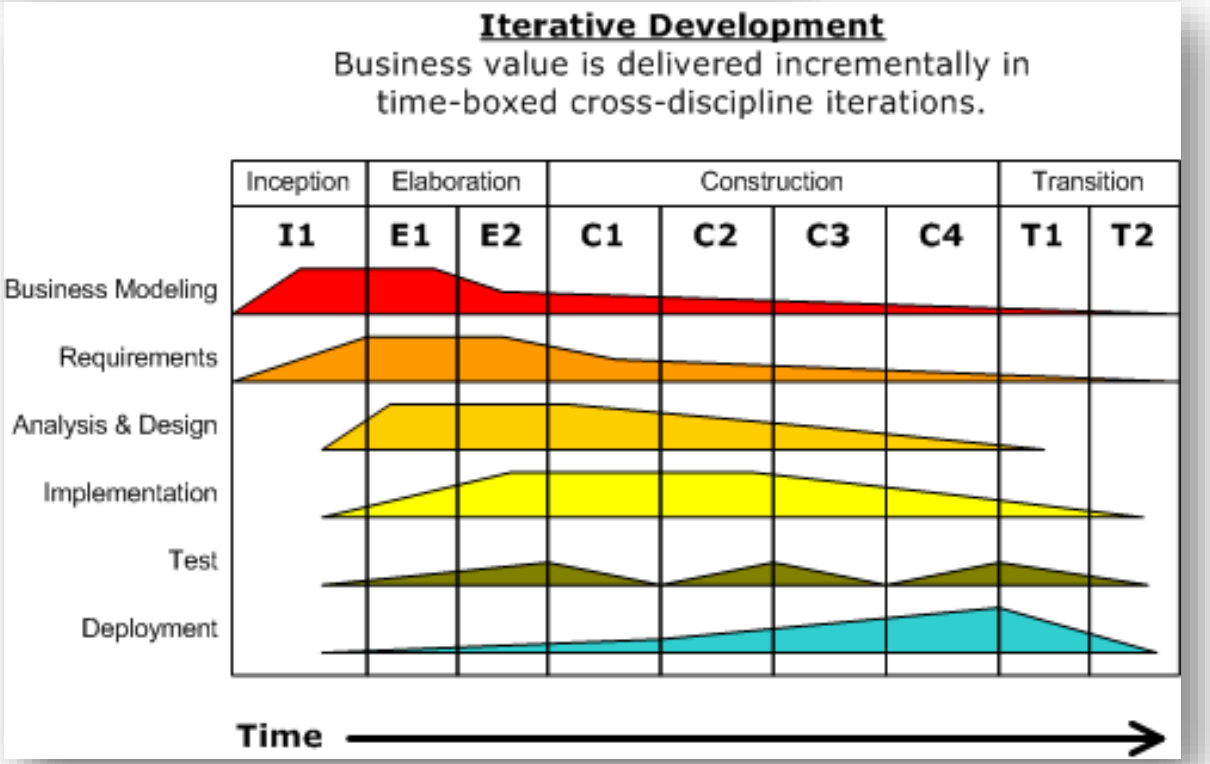
Transition fáze – nasazení do provozu

- Hlavním cílem transition fáze je nasazení plně funkčního systému do prostředí zákazníka.

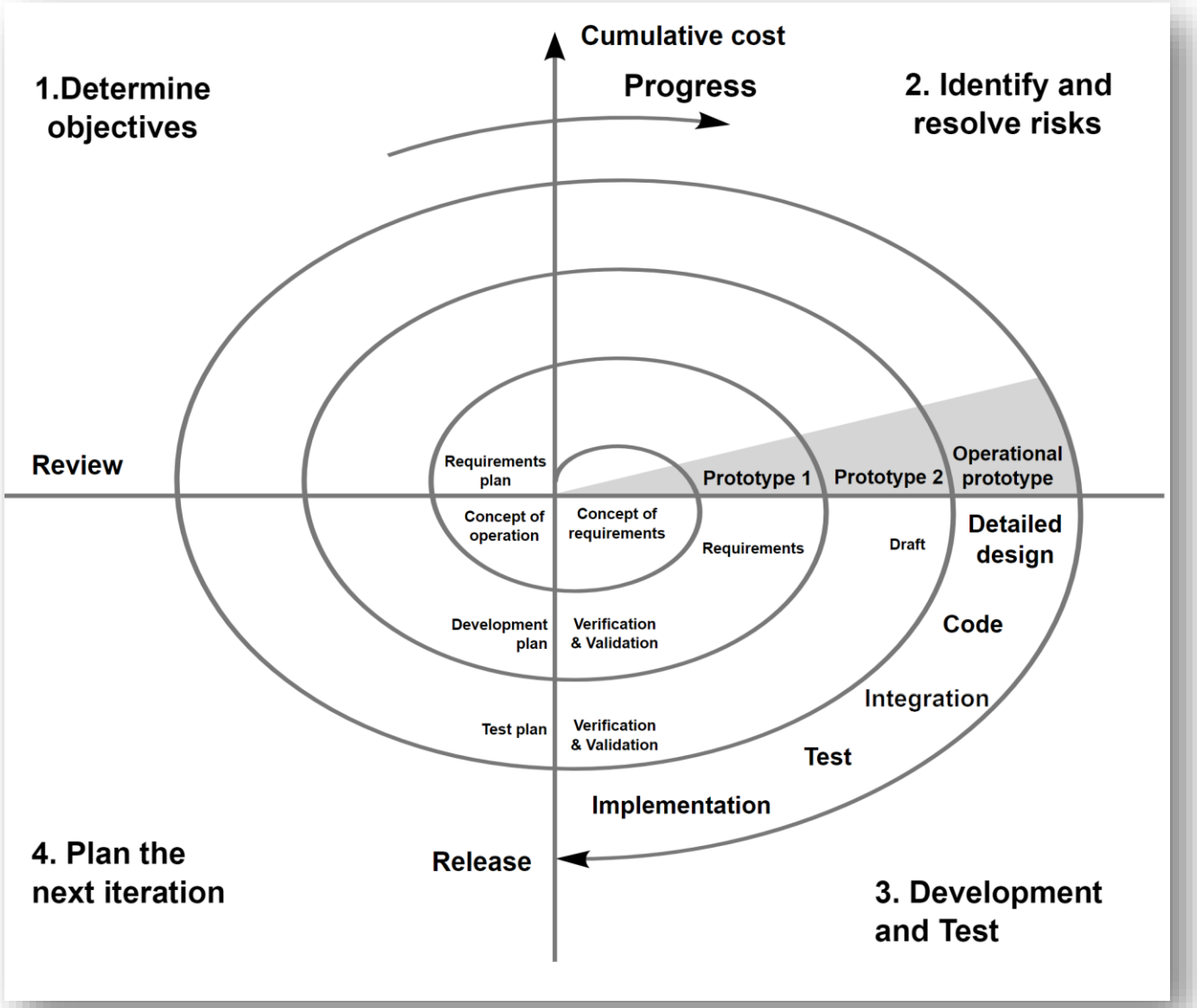
- Během fáze nasazení se projektový tým soustředí na řešení zákazníkem oznámených problémů a úpravu systému tak, aby byly odstraněny dříve nezjištěné vady.

- Hlavní milník této fáze je tzv. release.


Resources
Time
Inception
Elaboration
Construction
Transition


Iterative Development
Business value is delivered incrementally in time-boxed cross-discipline iterations.
Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Time

Spiral model (risk-focused)



Příklady

- Rational Unified Process (RUP)
- Microsoft Solutions Framework (MSF)
- Oracle Unified Method (OUM)
- Open Unified Process (OpenUP)

OpenUP

- https://download.eclipse.org/technology/epf/OpenUP/published/openup_published_1.5.1.5_20121212/openup/index.htm

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Manifesto for Agile Software Development (2001)

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

12 principů

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Working software is the principal measure of progress
5. Sustainable development, able to maintain a constant pace
6. Close, daily co-operation between business-people and developers
7. Face-to-face conversation is the best form of communication (co-location)
8. Projects are built around motivated individuals, who should be trusted
9. Continuous attention to technical excellence and good design
10. Simplicity
11. Self-organizing teams
12. Regular adaptation to changing circumstances

XP: Extrémní programování

- Kent Beck
- Extreme Programming Explained (1999)
- Všechno, co je správné, se dělá naplno. Nelze z ničeho ustoupit…
- Máte ovládací panel s otočným knoflíkem, od jedné do deseti, pro každou osvědčenou metodu.
  - Extrémní programování nastává, pokud všechny knoflíky otočíte na desítku…

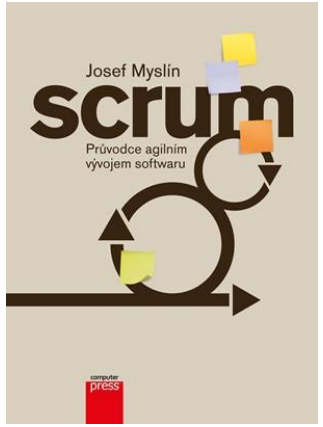
Obecné principy

- Komunikace
- Jednoduchost
- Zpětná vazba
- Odvaha

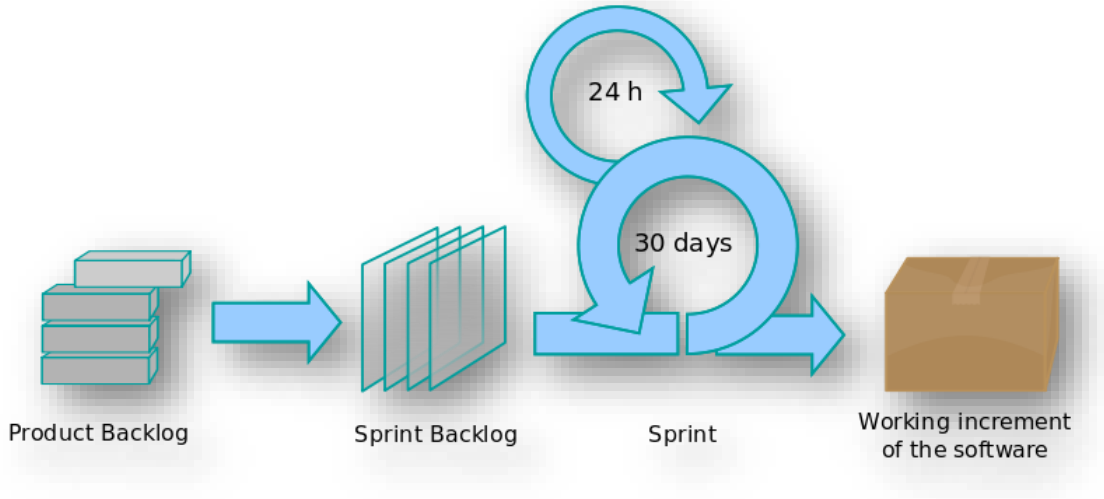
Praktiky

- Business praktiky (Plánování hry, Zákazník na pracovišti, Vydávání malých verzí, Metafora)
- Týmové praktiky (Párové programování, Společné vlastnictví kódu, Standardy kódu, Udržitelné tempo)
- Programovací praktiky (Neustálá integrace, Jednoduchý návrh, Refaktorování kódu, Testování)

SCRUM

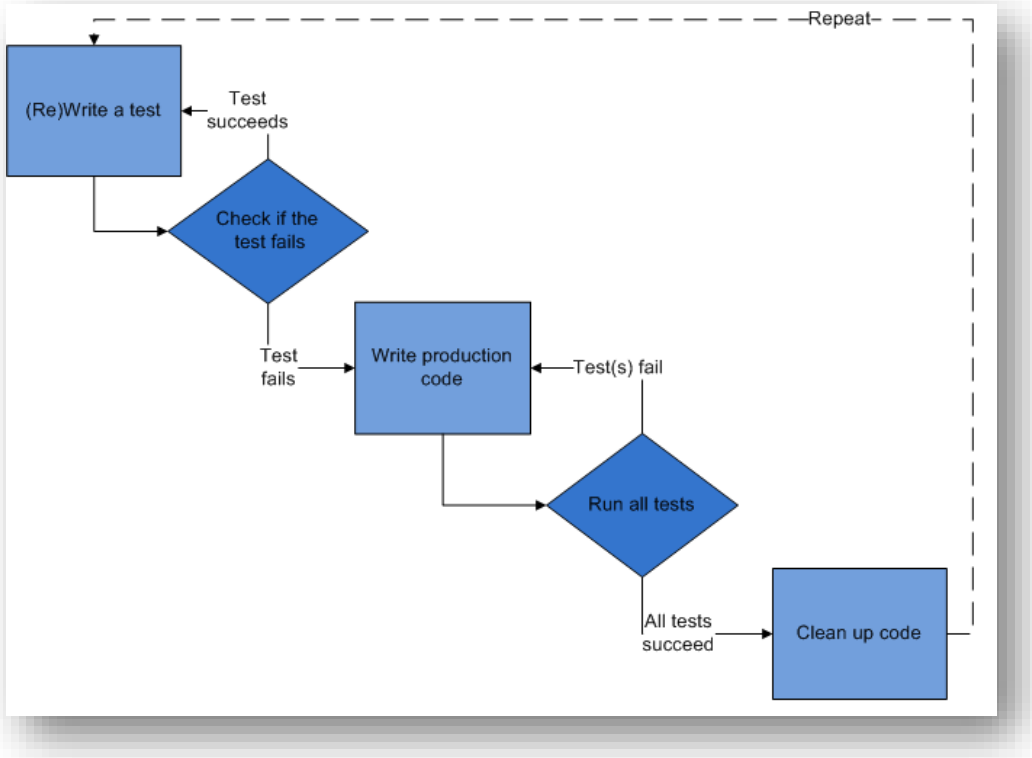


- 1995, Sutherland and Schwaber
- Product owner x vývojový tým x SCRUM master
- Backlog (user stories)
- Rizika (nejistoty)
- Sprinty
- Schůzky (denně)
- Flexibilita

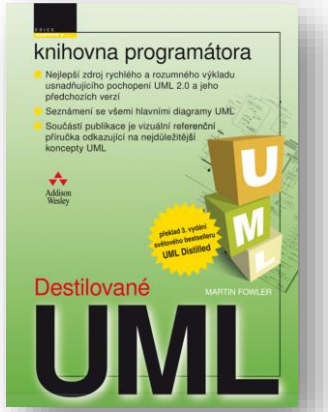


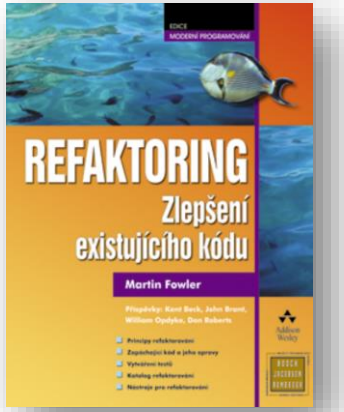
TDD: Programování řízené testy

- Napsat test
- Spustit testy a ujistit se, že všechny neprojdou
- Napsat vlastní kód
- Kód automatickými testy prochází
- Refaktoring
- Opakování

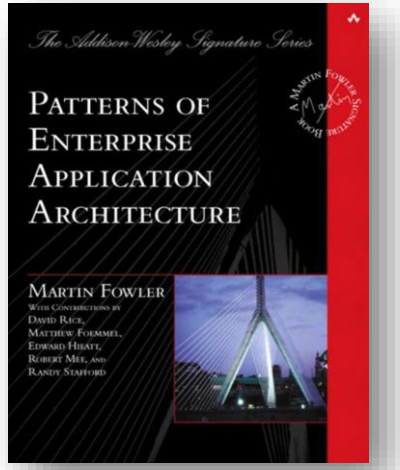


Shrnutí





- Nejdůležitější jsou lidé.
- Vývoj software nefunguje bez metodiky zaměřené na
  - odpovědnosti,
  - soustředění se na požadavky, architekturu a nejistoty,
  - průběžně revidované odpovědi na otázky:
    - CO-JAK-KDE-KDO-KDY-PROČ,
  - testování, zda realizovaný software splňuje definované požadavky.
- Musí existovat minimální projektová dokumentace.
- V důsledku je jediným měřítkem kvality realizace projektu funkční software s dobrým zdrojovým kódem.



Kontrolní otázky

1. Popište, co se rozumí vodopádovým modelem a jaké jsou jeho výhody a nevýhody.

2. Popište, co se rozumí iterativním a inkrementálním vývojem. Uveďte příklad.

3. Co se rozumí UP (unified process), na jakých principech, charakteristikách a fázích je postaven? Uveďte příklady.

4. Jaké jsou čtyři základní charakteristiky agilního softwarového vývoje, které ho odlišují od vodopádového a dalších robustních přístupů?

5. Na jakých principech a praktikách je založeno tzv. extrémní programování?

6. Které nejdůležitější charakteristiky má SCRUM?

7. Popište proces typický pro tzv. testy řízený softwarový vývoj. Uveďte příklad.