
# Agilní metodiky vývoje SW

# Agilní metodiky

- Agile = hbitý, čilý, bystrý, svižný.
- Manifesto for Agile Software Development (Manifest agilního programování), Utah, únor 2001
- www.agilealliance.org

# Manifest agilního programování

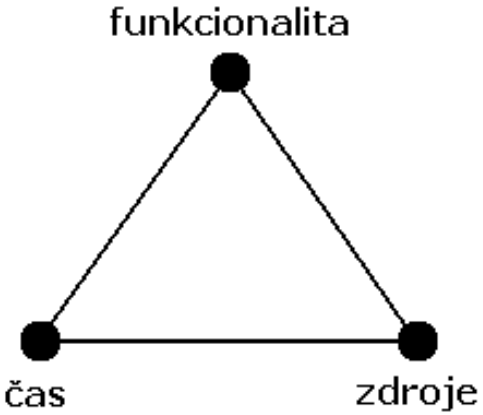
Umožnit změnu je mnohem efektivnější, než se jí snažit zabránit.

Je třeba být připraven reagovat na nepředvídatelné události, protože ty nepochybně nastanou.

Takže:

- Individuality a interakce mají přednost před procesy a nástroji.
- Fungující software má přednost před obsáhlou dokumentací.
- Spolupráce se zákazníkem má přednost před sjednáváním smluv.
- Reakce na změnu má přednost před plněním plánu.

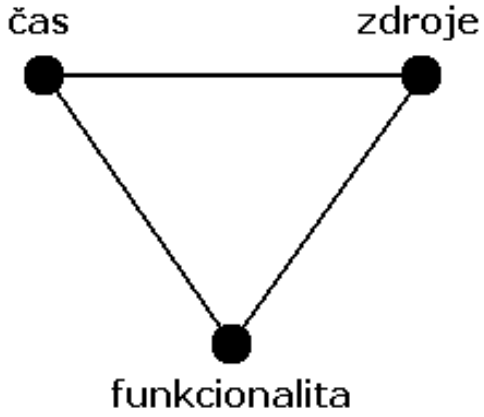
# Tradiční versus agilní přístup




tradiční přístup

**fixní**

**proměnné**




agilní přístup

# Agilní přístup

Společnými rysy všech agilních metodik jsou

- Iterativní a inkrementální vývoj s krátkými iteracemi
- Komunikace mezi zákazníkem a vývojovým týmem
- Průběžné automatizované testování

# Iterativní a inkrementální vývoj s krátkými iteracemi

Vývoj probíhá v krátkých fázích, takže celková funkcionalita je dodávána po částech.

Zákazník tak má možnost průběžně sledovat vývoj, muže se k němu vyjádřit a oponovat změny.

Zákazník má na konci jistotu, že nedostane něco, co neočekával.

# Komunikace mezi zákazníkem a vývojovým týmem

V ideálním případě je zákazník přímo součástí vývojového týmu, má možnost okamžitě vidět průběžné výsledky a reagovat na ně.

Zákazník se účastní sestavování návrhu, spolurozhoduje o testech a poskytuje zpětnou vazbu pro vývojáře.

# Průběžné automatizované testování

Díky krátkým iteracím se aplikace mění velice rychle, proto je nutné pro zajištění co nejvyšší kvality ověřovat její funkčnost průběžně.

Testy by měly být automatizované, předem sestavené a měly by být napsány ještě před samotnou implementací testované části.

Při každé změně musí být aplikována kompletní sada testů, aby nebyla porušena integrace jednotlivých částí.

# Konkrétní metodiky

- Extreme Programming (XP)
  Kent Beck, Ward Cunningham, Ron Jeffries (www.xprogramming.com)
- Feature-Driven Development (FDD)
  Jeff de Luca, Peter Coad (www.featuredrivendevelopment.com)
- SCRUM Development Process
  Ken Schwaber, Mike Beedle (www.controlchaos.com)
- Test-Driven Development (TDD)
  Kent Beck (www.testdriven.com)

# Extreme Programming

Hodí se pro menší projekty a malé týmy, vyvíjející software podle zadání, které je nejasné nebo se rychle mění.

Jediným exaktním, jednoznačným, změřitelným, ověřitelným a nezpochybnitelným zdrojem informací je zdrojový kód.

Používá běžné principy a postupy, které dotahuje do extrému.

Například:

- jestliže se osvědčují revize kódu, bude se neustále revidovat (myšlenka párového programování).
- pokud se vyplácí testování, bude se nepřetržitě testovat, a to i u zákazníka (testování jednotek, funkcionality, akceptační testy).
- osvědčuje-li se návrh, stane se součástí každodenní činnosti (refaktorizace).

Principy XP:

- Rychlá zpětná vazba**,** která spočívá v rychlém zjištění stavu systému po provedené akci, vyhodnocení akce a uložení výsledku vyhodnocení co nejdříve zpět do systému.
- Předpoklad jednoduchosti, představuje v mnoha ohledech nejobtížnější princip,protože je v protikladu s tradičním pojetím programování, kdy se vše plánuje a navrhuje do budoucna tak, aby to bylo znovu použito. XP naopak předpokládá u řešitelského týmu schopnost přidat složitost tam, kde to bude v budoucnu účelné.

## Principy XP:

- Přírůstková změna, vychází z předpokladu, že velké změny provedené najednou nefungují. Veškeré změny v projektu se proto provádějí pomocí malých přírůstků.
- Využití změny, vychází z předpokladu, že nejlepší strategie je ta, která si zachová co nejvíce možností řešení nejnaléhavějších problémů projektu.
- Kvalitní práce, která představuje fixní proměnnou ze čtyř proměnných pro posouzení projektu (šíře zadání, náklady, čas a kvalita) s hodnotou vynikající, při horší hodnotě členy týmu práce nebude bavit a projekt může skončit neúspěchem.

# Feature-Driven Development

Vývoj po malých kouscích (vlastnostech, rysech), což jsou elementární části funkcionality přinášející nějakou hodnotu uživateli.

Vývoj probíhá v pěti fázích, první tri jsou sekvenční, poslední dvě pak iterativní.

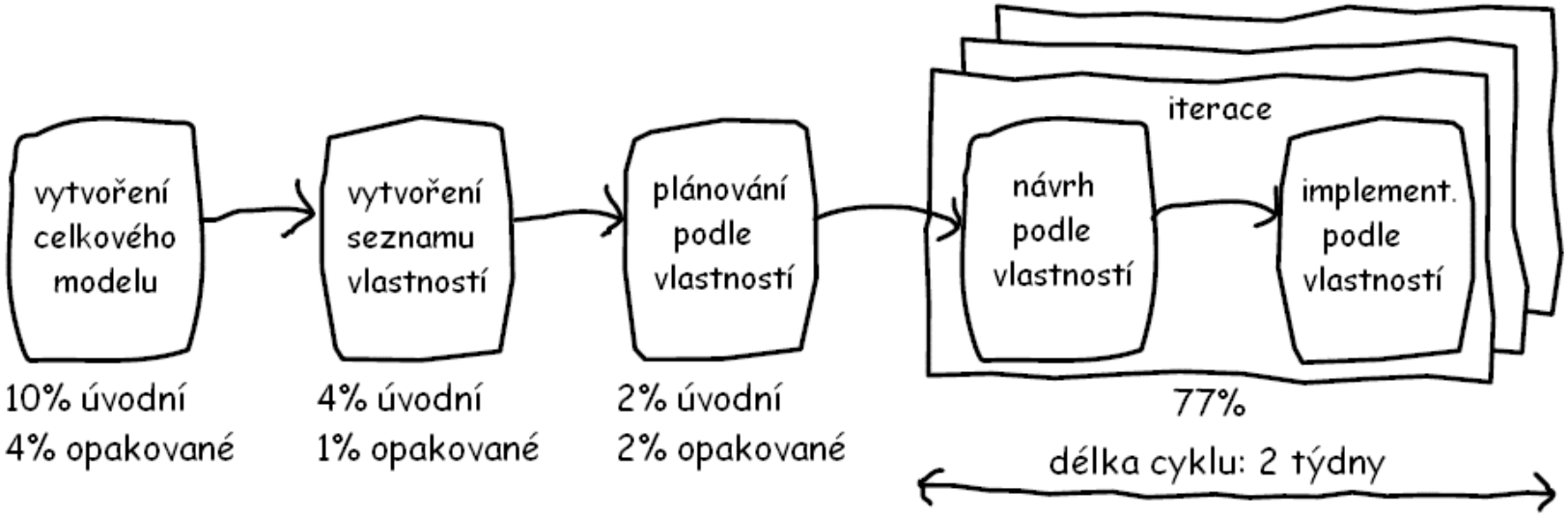
Iterace trvají zpravidla 2 týdny. Začíná se vytvořením modelu, ten se převede do seznamu vlastností, které se postupně implementují.

Měří pokrok ve vývoji projektu, FDD detailně plánuje a kontroluje vývojový proces.

Zaměření na dodávání fungujících přírůstku každé dva týdny.

# Feature-Driven Development

## Fáze metodiky z pohledu rozdělení času:



# SCRUM Development Process

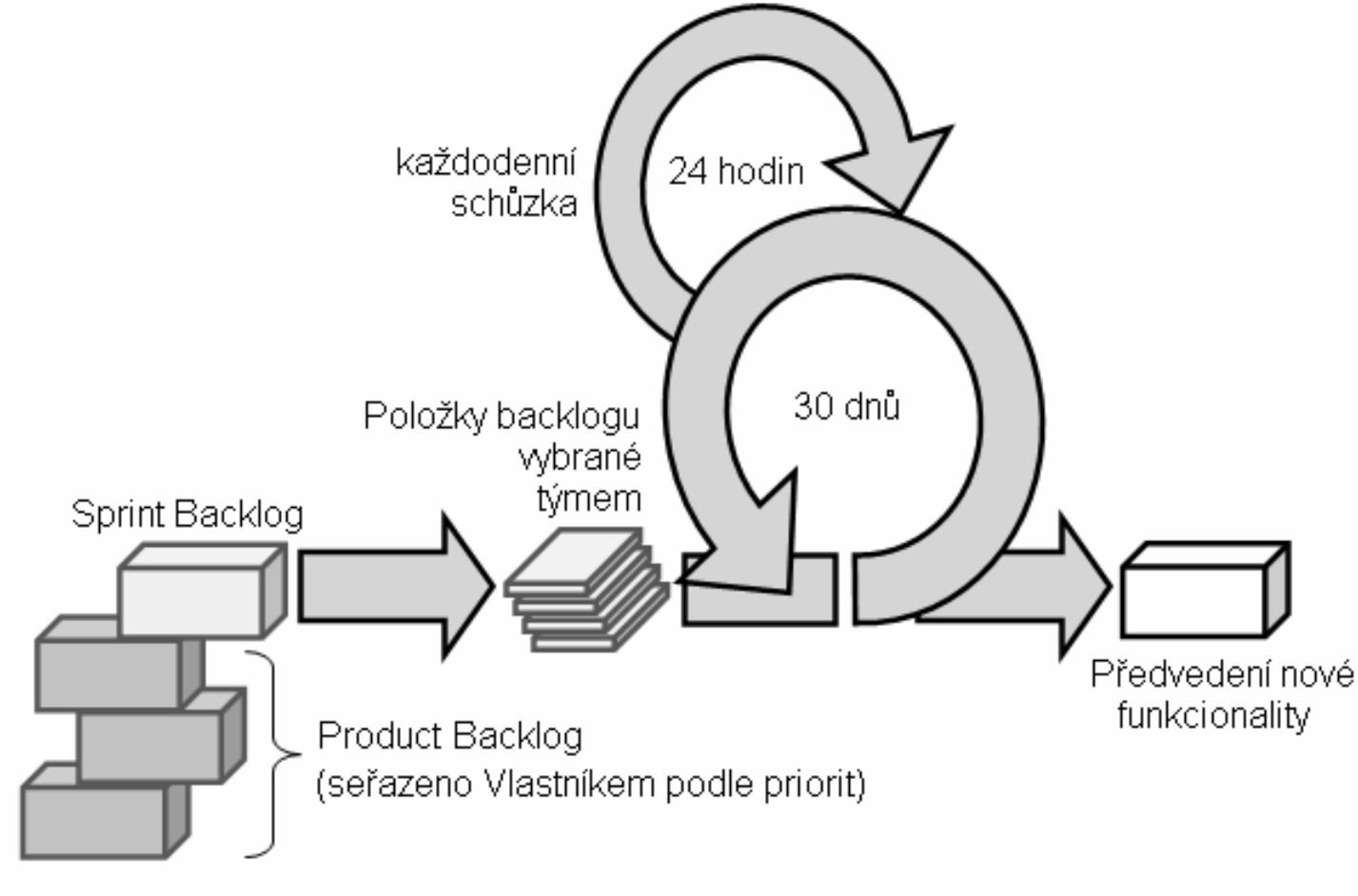
Principem je opět iterativní vývoj definující 3-8 fází (tzv. sprintu), z nichž každá trvá přibližně měsíc.

Nedefinuje žádné konkrétní procesy, pouze zavádí pravidelné každodenní schůzky vývojového týmu (scrum meetings), kde si jednotliví členové sdělují, které položky byly od minulé

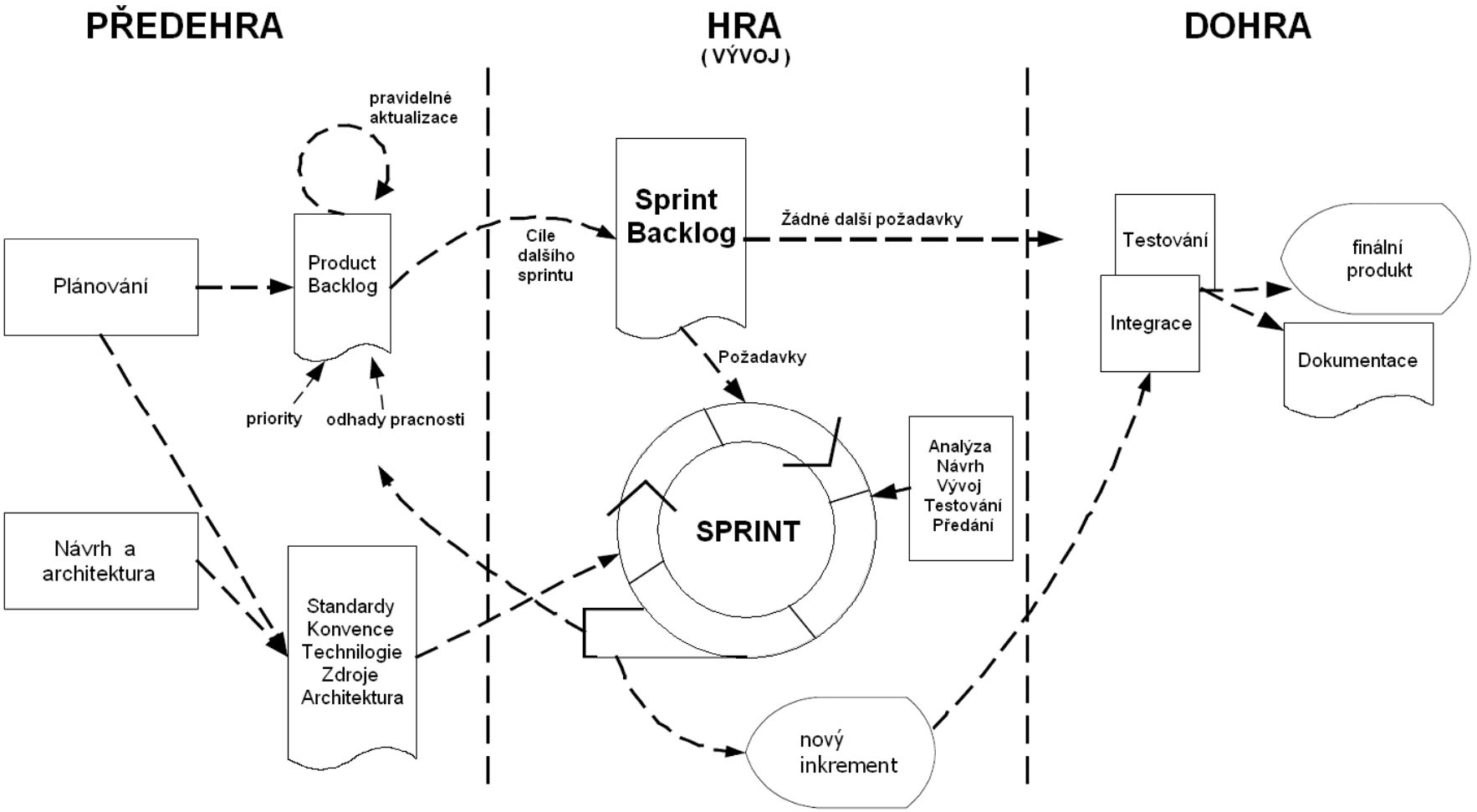
schůzky dokončeny, které nové úkoly nyní přijdou na řadu a jaké překážky stojí vývoji v cestě a musí se řešit.

Každý sprint je zakončen předvedením funkční demo-aplikace zákazníkovi, který poskytne zpětnou vazbu.

# SCRUM Development Process



# SCRUM Development Process



# Test-Driven Development

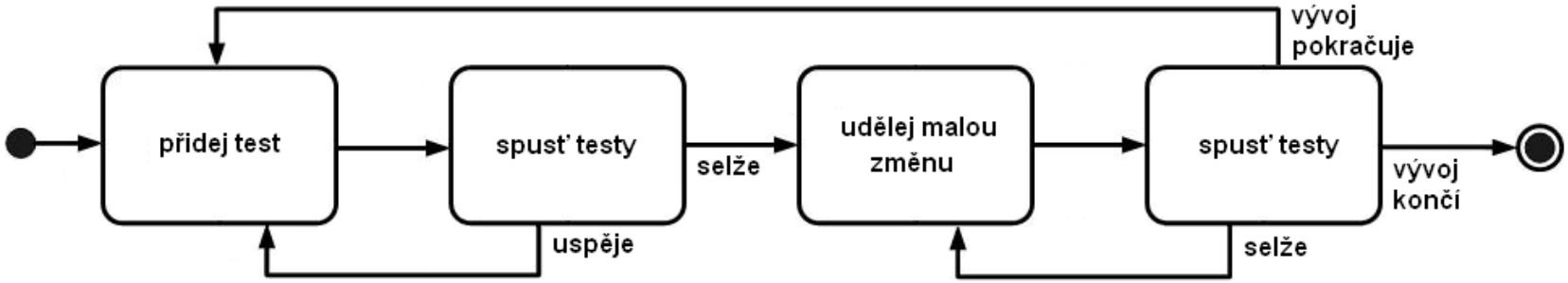
Nezabývá se tvorbou specifikací, plánu a dokumentace, to si každý tým musí zvolit sám podle toho, jak mu to vyhovuje.

Doporučuje přistoupit k testům jako k hlavní fázi celého vývojového procesu.

Základním pravidlem je psát testy dříve než samotný kód a implementovat jen přesně ty části kódu, které projdou testem.

# Test-Driven Development

## Postup metodiky:



# Srovnání metodik z pohledu životního cyklu SW