Agilní metodiky a techniky

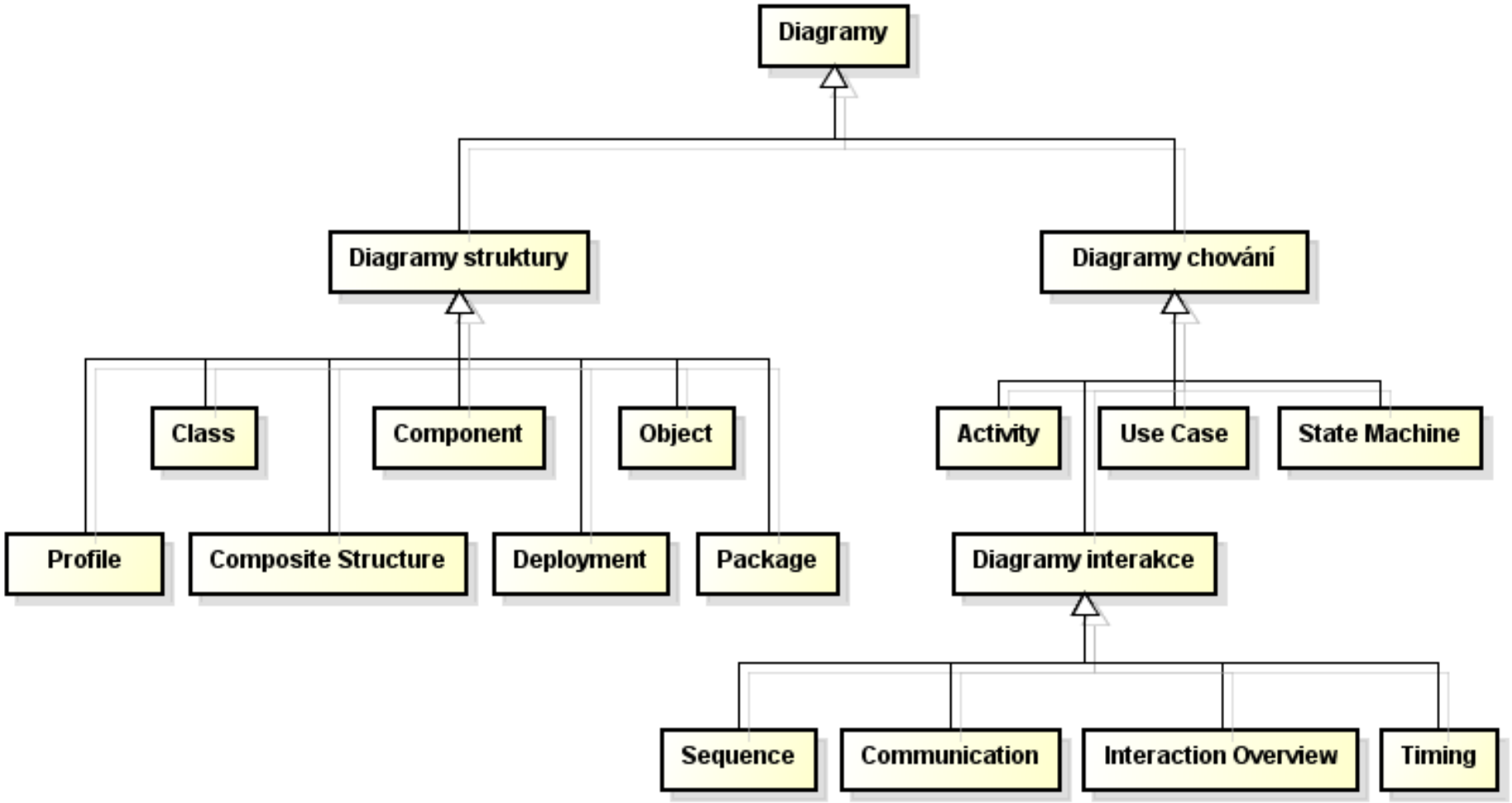
analýza a vývoj IS

Využití UML

- UML jako náčrt systému
- UML jako plán vývoje
- UML jako programovací jazyk

- Příklad: Analýza - chyby v zákoně viz http://blog.geospy.org/tagged/Anal%C3%BDza

Současný stav UML (14 diagramů)



Význam analýzy požadavků na IS

- jednoznačně identifikovat a popsat určitý pojem zákazníka,
- objasnění všech nejasných a odstranění neúčelných pojmů,
- způsob, jak se seznámit s aplikací nově zúčastněné osoby,
- existují jednotné informace o tom, jak má aplikace fungovat,
- v aplikaci je možno jednoznačně identifikovat odchylky od požadavků zákazníka = chyby,
- při rozšiřování aplikace víme na co navazujeme. Je tak možné stanovit systém, jak rozšíření/změnové požadavky zapracovat.

Význam analýzy požadavků na IS

- informace pro identifikaci problémů v aplikaci a jejich odstranění.
- popis systému pro její opětné vytvoření.

Agilní metodiky a techniky

Agilní metodiky jsou skupiny metod původně určených pro vyvíjení SW založené na iterativním a inkrementálním vývoji. Umožňují rychlý vývoj softwaru a zároveň dokáží reagovat na změnu požadavků v průběhu vývojového cyklu. Podle těchto metodik se správnost systému ověří jedině pomocí rychlého vývoje, předložení zákazníkovi a následných úprav dle zpětné vazby.

Historie agilních metodik

Techniky užívané agilními metodikami se často používaly už dříve, ale pojem se začal používat až v únoru 2001. V Utahu se tehdy sešli odborníci z oblasti softwarového inženýrství a vývoje softwaru, aby diskutovali o odlehčených metodách vývoje. Sepsali *Manifest agilního programování*, kde definují přístup k vývoji nyní známý jako agilní programování.

Historie agilních metodik

Až do nástupu odlehčených metodik se používaly těžké, rigorózní metodiky. Ty jsou někdy kritizovány pro svůj vodopádový model vývoje a pro to, že vedoucí projektu svým blízkým dohledem omezuje vývojáře v práci. Kromě toho obtížně reagují na změny.

V polovině 90. let se začaly objevovat odlehčené metodiky. Ty se podle jejich tvůrců navrací k vývojovým praktikám ze samotných počátků softwarového vývoje. Mezi tyto odlehčené metodiky patřil **Scrum, Crystal Clear, Extrémní programování či Vývoj řízený vlastnostmi.** Od publikování Manifestu agilního programování se tyto metodiky nazývají agilními.

Agilní metodika a technika

- **Agilní metodika** je způsob rozvržení a ověřování práce. Cílem Agilní metodiky bývá lepší organizace práce. Odlišné Agilní metodiky vedou k odlišným produktům, protože považují jiné aspekty produktu za klíčové. Liší se i přístup ke změně zadání. Možnost změny zadání je ale určující podmínkou.
- **Agilní technika** je naproti tomu konkrétní postup, který se většinou týká samotných vývojářů. Cílem těchto technik bývá zvýšení produktivity práce, odstranění chyb, kvalitnější software nebo přesnější dodržení specifikace. Agilní techniky jsou od metodik oddělitelné. Cílem Agilní techniky bývá kvalitnější produkt.

Rozdíly mezi agilní metodikou a technikou

- Agilní metodika se zaměřuje na organizaci práce bez ohledu na to, zda se ve firmě vyvíjí software, nebo se jedná o jinou oblast, kde lze agilní řízení uplatnit.
- V agilní metodice je kladen důraz na možnost změny, agilní technika nic takového nevyžaduje.
- Agilní metodika se týká nejen vývojářů, ale i managementu.
- Agilní technika je konkrétní činnost (především vývojáře, může se ale týkat i například klienta).

Přínos agilní metodiky a techniky

- Je předem známo, kolik práce je tým schopen udělat (v časovém rozsahu jedné iterace). To proto, že z minulosti je známo, jaká je rychlost v jednotlivých iteracích. Protože se zároveň dělají časové odhady na co nejkratší úkoly, je možné řídit rychlost týmu.
- Změny zadání za běhu (což je například pro klasický Waterfall model naprosto neřešitelný problém)
- Zrychlení doručování produktu od vývojáře k zákazníkovi (protože klientovi produkt doručujete po každém běhu – zákazník se navíc často účastní běhu firmy)

Přínos agilní metodiky a techniky

- Garance kvality (pomocí automatizovaných testů)
- Odstranění třecích ploch mezi produktem a vývojáři.
- Omezení rigidních procesů ve prospěch komunikace (jakkoliv Agile proti dobře nastaveným procesům nejdou – pouze omezují ty procesy, které brání efektivitě, komunikaci a agilnímu přístupu k práci)

Priority agilního programování

- Lidé a jejich spolupráce před procesy a nástroji
- Fungující software před obsáhlou dokumentací
- Spolupráce se zákazníkem před sjednáváním smluv
- Reakce na změnu před dodržováním plánu

Principy agilního programování

1. Nejvyšší prioritou je uspokojit zákazníka skrz rychlé a průběžné dodávání kvalitního software.
2. Změnové požadavky jsou vítány, dokonce i v průběhu vývoje. Agilní procesy je zpracují tak, aby zákazníkovi přinášely konkurenční výhody.
3. Dodávejte fungující software často, v intervalech týdnů až měsíců. Upřednostňujte kratší intervaly dodání.
4. Lidé z businessu a vývojáři musí spolupracovat každý den během celého projektu.
5. Pro práci na projektu vybírejte motivované jedince. Dejte jim prostředí a podporu, kterou potřebují, a důvěřujte jim, že práci dokončí.
6. Nejúčinnější metoda sdílení informací vývojářskému týmu (i uvnitř tohoto týmu) je osobní setkání.

Principy agilního programování

7. Fungující software je hlavním měřítkem postupu vývoje.
8. Agilní procesy podporují udržitelný vývoj. Sponzoři, vývojáři i uživatelé by měli být schopní dodržovat stálý výkon dokud je třeba.
9. Průběžná pozornost věnovaná technické dokonalosti a dobrému návrhu posiluje agilní přístup.
10. Základem je jednoduchost – umění co nejvíce práce vůbec nedělat.
11. Nejlepší architektury, požadavky a návrhy vznikají v týmech, které se samy organizují.
12. Tým v pravidelných intervalech vyhodnocuje svou práci a upravuje své postupy tak, aby byl co nejefektivnější.

Fáze celého projektu v agilních metodikách

1. **Nultá iterace** – první krátká analýza a naprogramování nějaké základní činnosti. V agilním týmu nejde příliš o to, co se v této části bude implementovat. Podmínkou je, aby na konci byl hotový kousek aplikace, který se dá předvést (tedy i klientovi).
2. **Analýza změny** (výběr toho, co se bude implementovat, analýza a designování změn).
3. Samotná **implementace požadované vlastnosti** (či vlastností).

Fáze celého projektu v agilních metodikách

4. **Předvedení klientovi**.
5. **Pokud není produkt hotov**, zpět na bod 2.
6. Pokud ano, následuje **údržba, rozvoj** (rovněž v iteracích).

Body 2–4 se označují jako **jedna iterace**. Tyto body se opakují tak dlouho, dokud není vývoj ukončen (úspěchem, odložením projektu, proto, že dojdou peníze, změnou priorit klienta).

SCRUM – metoda řízení agilního vývoje

- V programování **SCRUM** (česky mlýn, skrumáž) je iterativní a inkrementální metodologie agilního vývoje softwaru používaná na řízení produktového vývoje.
- Definuje flexibilní, holistickou strategii produktového vývoje, kde vývojový team pracuje jako jednotka na dosažení společného cíle", zpochybňuje předpoklady "tradičního, sekvenčního přístupu" k vývoji produktu, a umožňuje týmům se samoorganizovat podpořením fyzické kolokace nebo blízké online spolupráce všech členů týmu, stejně jako denní ústní komunikaci všech členů týmu a disciplín v projektu.

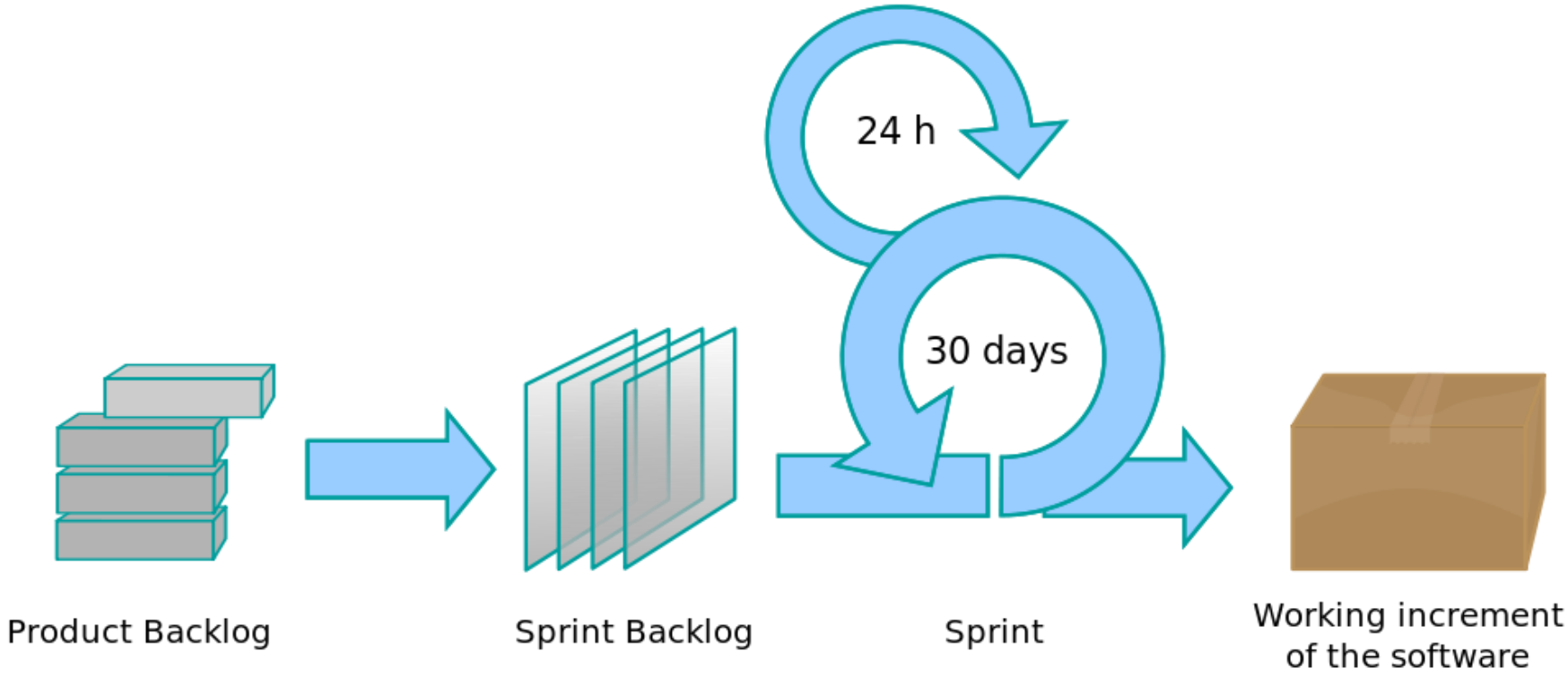
SCRUM – metoda řízení agilního vývoje

- Klíčový princip SCRUMU je jeho pochopení, že během projektu mohou zákazníci změnit názor o tom, co chtějí a potřebují (často zvané "souhrn požadavků") a že nepředvídané úkoly nelze jednoduše řešit tradičním předvídáním a plánováním.
- Scrum používá empirický přístup, podle kterého problém nelze zcela pochopit nebo definovat a proto se soustředí na maximální schopnost týmu rychle dodat a reagovat na nové požadavky.

SCRUM – metoda řízení agilního vývoje

- Základem SCRUMU je **Sprint** - periodická činnost (s periodou obvykle uváděnou v týdnech), která je neustále kontrolována (např. každých 24 hod, 48 hod).
- Vstupními parametry Sprintu jsou tzv. **Backlog itemy** (seznam návrhů na vylepšení stávajícího produktu, příp. chyby) a výstupním parametrem je **Produkt** s novou funkcionalitou, příp. bez zjištěných chyb.

SCRUM proces



Role v SCRUM

- **Product Owner** - osoba zodpovědná za výsledný produkt, přiřazuje priority jednotlivým backlog itemům (vytváří Selected Backlog)
- **Scrum Master** - osoba, jejíž úkolem je zajistit hladký průběh sprintu, organizovat schůzky, řešit případné administrativní problémy
- **Team -** programátoři

- **Stakeholder(s)** – uživatelé produktu, manažeři, osoby, kterých se problém dotýká, ale produkt (systém či SW) nevytvářejí

Vlastník produktu (product owner)

- Vlastník produktu reprezentuje zainteresované subjekty a je hlasem zákazníka. Je odpovědný za ujištění, že team do byznysu přidá hodnotu. Vlastník produktu píše články pro zákazníky (typicky zkušenosti uživatelů), hodnotí a přiřazuje jim priority, a přidává je do produktového backlogu.
- Scrum týmy mají mít jednoho vlastníka produktu, tato role se nemá spojit se scrum masterem. Product owner má být na obchodní straně projektu, a nikdy nemá interagovat s členy týmu o technických aspektech vývoje. Tato role je stejná jako role customer representative (reprezentant klientů) v jiných agilních frameworcích.

Role product ownera

- demonstrovat řešeni pro klíčové stakeholdery, kteří nebyly na iteračním demu
- ohlašovat uvedení nových verzí
- komunikovat stav týmu
- organizovat milníkové přehledy
- vzdělávat v procesu vývoje
- dohadovat priority, rozsah, financování a rozvrh
- ujistit se, že produktové testy jsou viditelné, transparentní a jasné

Vývojový team

- Je odpovědný za dodání potenciálně použitelných inkrementů (potentially shippable increments - PSIs) produktu na konci každého sprintu (cíl sprintu).
- Team je složen z 3-9 jednotlivců, kteří dělají aktuální práci (analýza, design, vývoj, test, technická komunikace, dokument, atd.). Vývojové týmy jsou vícefunkční, se všemi dovednostmi vytvořit produktový inkrement.
- Vývojový team v skrumu je sebeorganizující, i když tady může být nějaký stupeň interakce s projektovým managementem (project management offices - PMOs).

Scrum master

- Scrum je usnadněný scrum masterem (mistrem), který je odpovědný za odstranění překážek teamu na dodání produktových cílů.
- Scrum master není tradiční team leader nebo projektový manažer, ale koná jako prostředník mezi teamem a jakýmikoli negativními vlivy.
- Scrum master zajišťuje, že scrum proces je použit jako bylo plánováno a členové týmu dodržují dohodnuté procesy. Často organizuje schůzky a povzbuzuje tým k zlepšení. Tato role se někdy označuje jako "team facilitator" na zdůraznění duální funkce.

Eventy (události)

- **Sprint** (nebo iterace) je základní jednotka vývoje ve scrumu. Sprint je časově omezená snaha, tedy je omezen na specifický čas. Doba je určena dopředu pro každý sprint a obvykle je to jeden týden až jeden měsíc, nejčastěji 2 týdny.
- Každý sprint začíná eventem **plánování sprintu**, cílem kterého je definovat úkoly sprintu, kde je definována práce sprintu a odhadnut závazek pro cíl sprintu.
- Každý sprint končí **recenzí sprintu** (sprint review) a retrospektivou, kde je reportován progres pro stakeholdery a definují se úlohy na zlepšení pro další sprinty.
- Scrum zdůrazňuje zpracovaný produkt na konci sprintu, který je opravdu hotový, v případě software to může znamenat, že software byl integrován, kompletně testován, zdokumentován a potenciálně může být dodán.

Plánováni sprintu

Na začátku sprintu, tým organizuje *sprint planning event* - plánovací událost:

- vybrat jaká práce se udělá
- připravit úkoly sprintu, které určí čas na úkoly pro celý tým
- definovat a diskutovat kolik práce je třeba udělat během sprintu
- 4-hodinový limit pro 2-týdenní sprint, poměrně pro jinou délku
  - v první půli se celý tým (vývojový tým, scrum master a product owner) dohodnou, které produktové položky (backlog) budou v daném sprintu uvažovat
  - v druhé půli vývojový team urči práci (úlohy) potřebné na dodání položek backlogu, co rezultuje do *sprintového backlogu*

Denní scrum

Každý den během sprintu team organizuje *denní scrum* (nebo *stand-up*) se specifickými zásadami:

- Všichni členové softwarového týmu přijdou připraveni.
- Denní scrum začne přesně načas i když někteří členové chybí.
- Denní scrum má proběhnout každý den na stejném místě a ve stejný čas.
- Délka denního scrumu je omezena na 15 minut.
- Každý je vítán, i když obvykle mluví jen teamové role scrumu.

SCRUM in under 10 minutes

- https://www.youtube.com/v/Q5k7a9YEoUI%26hl=en%26fs=1%26

Příklady dalších agilních metodik

Extrémní programování (XP)

XP je pravděpodobně nejznámější agilní metodikou. Propaguje časté dodávky software v krátkých vývojových cyklech. Kromě toho navrhuje párové programování, jednotkové testování celého kódu (nejdříve se vytvoří test, až pak samotný kód), programovat jen to, co je v danou chvíli nezbytné, jednoduchý a jasný kód.

Mezi další praktiky patří společné vlastnictví kódu a neustálý refaktoring. Vývojáři by měli počítat se změnami požadavků v průběhu času. Důležitá je častá komunikace se zákazníkem i mezi programátory.

Příklady dalších agilních metodik

Vývoj řízený vlastnostmi (FDD – Feature Driven Development)

FDD začíná vytvořením doménového modelu popisujícího celý systém. Ten se převede do seznamu vlastností (elementární funkcionality, které přináší hodnotu uživateli). Vývoj má celkem pět fází (první tři sekvenční, další dvě iterativní). Iterace trvá většinou dva týdny. Během každé iterace se implementují konkrétní užitné vlastnosti systému. Zákazník průběžně dostává mezivýsledky a nové verze produktu. Na rozdíl od XP nebo SCRUM je jednotlivým programátorům práce přidělena – nevybírají si ji sami.

Příklady dalších agilních metodik

Vývoj řízený testy (TDD – Test Driven Development)

TDD navrhuje psaní testů před samotným kódem a následně naprogramovat samotný kód. Implementuje se přesně takové množství kódu, jaké dokáže projít testem. TDD navrhuje psaní testů před samotným kódem a následně naprogramovat samotný kód. Implementuje se přesně takové množství kódu, jaké dokáže projít testem.

Příklady dalších agilních metodik

Crystal metodiky

Nejde jen o jednu metodiku. Hlavní myšlenkou je to, že je lepší metodiku přizpůsobit danému projektu, žádná metodika nebude vyhovovat každému projektu. Vytvoření individuální a účelové metodiky pro konkrétní projekt je první fází vývoje. Pro vytvořenou metodiku je rozhodující například velikost projektu a vývojového týmu.

Příklady dalších agilních metodik

Lean development

Lean development je spíš než metodikou souhrnem pravidel, jejichž používání by mělo zefektivnit a zrychlit vývojový proces. Tato pravidla zní: eliminovat zbytečné (to, co nepřináší zákazníkovi žádnou hodnotu), zdůraznit proces učení, rozhodovat se tak rychle a pozdě, jak je možné, posílit odpovědnost týmu, zabudovat integritu a vidět systém jako celek.