Kapitola 3 – Agilný vývoj softvéru

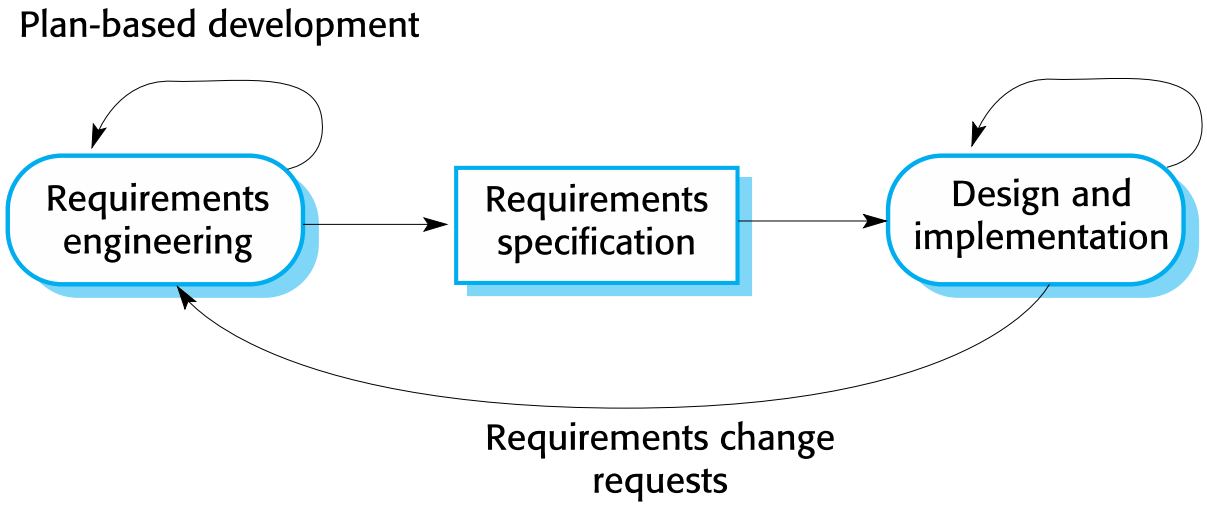
Rýchly vývoj softvéru

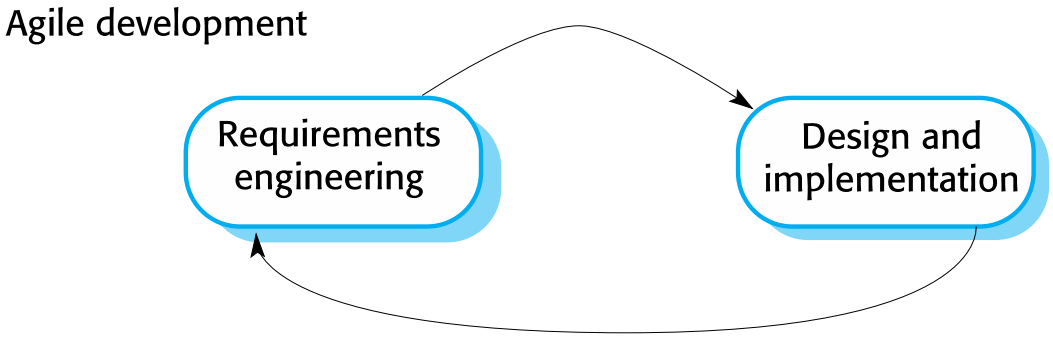
- Rýchly vývoj a dodávka je v súčasnosti často najdôležitejšou požiadavkou na softvérové systémy
  - Podniky fungujú pri rýchlo sa meniacich požiadavkách a je prakticky nemožné vytvoriť súbor stabilných softvérových požiadaviek
  - Softvér sa musí rýchlo vyvíjať, aby odrážal meniace sa obchodné potreby.
- Plánom riadený vývoj je nevyhnutný pre niektoré typy systémov, ale nespĺňa tieto obchodné potreby.
- Agilné vývojové metódy sa objavili koncom 90-tych rokov, ktorých cieľom bolo radikálne skrátiť dodaciu dobu fungujúcich softvérových systémov.

Agilný vývoj

- Špecifikácia programu, návrh a implementácia sú vzájomne prepojené
- Systém je vyvinutý ako séria verzií alebo prírastkov so zainteresovanými stranami zapojenými do špecifikácie prírastkov a ich vyhodnotenia
- Časté dodávanie nových verzií na testovanie
- Rozsiahla podpora nástrojov (napr. automatizované testovacie nástroje) používaných na podporu vývoja.
- Minimálna dokumentácia – zameranie na fungujúci kód

Plánom riadený a agilný vývoj





Plánom riadený a agilný vývoj

✧ Plánom riadený vývoj

- Plánom riadený prístup k softvérovému inžinierstvu je založený na samostatných fázach vývoja, pričom výstupy, ktoré sa majú vytvoriť v každej z týchto fáz, sú vopred naplánované.
- Nie nevyhnutne vodopádový model – je možný postupný vývoj podľa plánu
- K opakovaniu dochádza v rámci jednotlivých aktivít vývoja.

✧ Agilný vývoj

- Špecifikácia, návrh, implementácia a testovanie sú vzájomne prepojené a o výstupoch z procesu vývoja sa rozhoduje prostredníctvom procesu vyjednávania počas procesu vývoja softvéru.

Agilné metódy

- Nespokojnosť s režijnými nákladmi spojenými s metódami návrhu softvéru v 80. a 90. rokoch 20. storočia viedla k vytvoreniu agilných metód. Tieto metódy:
  - Sa zameriavajú viac na kód ako na návrh
  - Sú založené na iteratívnom prístupe k vývoju softvéru
  - Sú určené na rýchle dodanie funkčného softvéru a jeho rýchle prispôsobenie, aby vyhovoval meniacim sa požiadavkám .
- Cieľom agilných metód je znížiť réžiu v softvérovom procese (napr. obmedzením dokumentácie) a vedieť rýchlo reagovať na meniace sa požiadavky bez nadmerného prerábania.

Agilný manifest

- *Viac jednotlivci a interakcie ako procesy a nástroje*
- *Viac funkčný softvér ako komplexná dokumentácia*
- *Viac spolupráca so zákazníkom ako vyjednávanie zmluvy*
- *Viac reakcie na zmenu ako ísť podľa plánu*

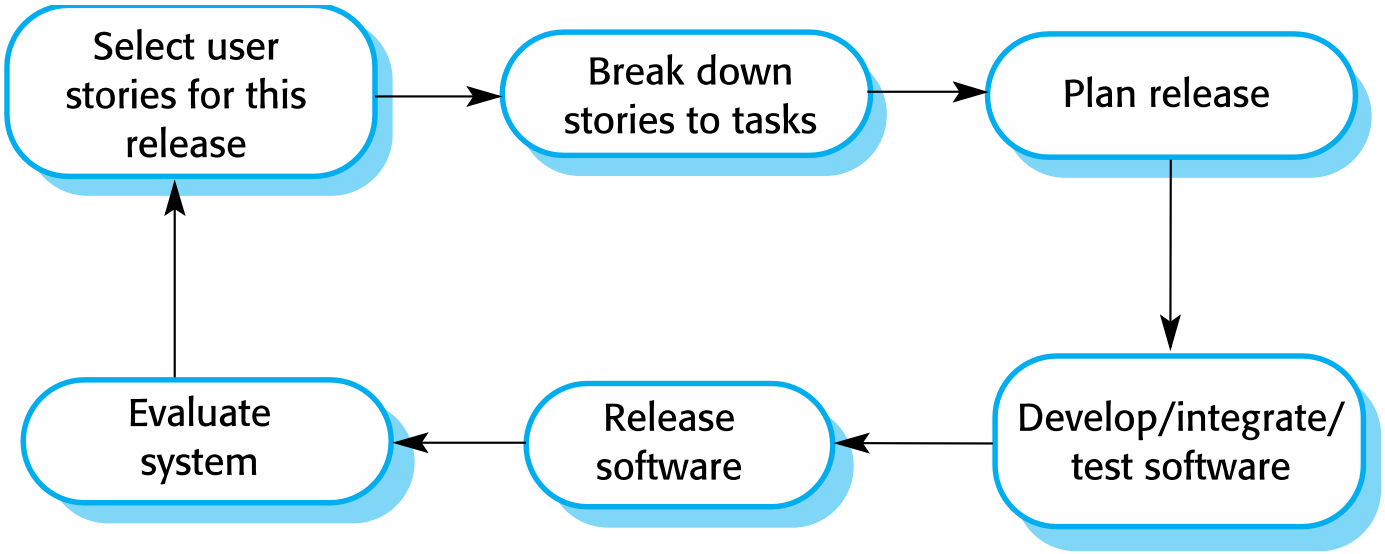
Princípy agilných metód

Princíp	Popis
Zapojenie zákazníka	Zákazníci by mali byť úzko zapojení do celého procesu vývoja. Ich úlohou je poskytovať a uprednostňovať nové systémové požiadavky a vyhodnocovať prírastky systému.
Prírastkové dodanie	Softvér sa vyvíja postupne, pričom zákazník špecifikuje požiadavky, ktoré majú byť zahrnuté v každom prírastku.
Ľudia nie procesy	Zručnosti vývojového tímu by sa mali uznávať a využívať. Členovia tímu by mali mať možnosť rozvíjať svoje vlastné spôsoby práce bez normatívnych procesov.
Prijať zmeny	Očakávajte, že sa systémové požiadavky zmenia, a preto navrhnite systém tak, aby vyhovoval týmto zmenám.
Zachovajte jednoduchosti	Zamerajte sa na jednoduchosť vo vyvíjanom softvéri aj v procese vývoja. Vždy, keď je to možné, aktívne pracujte na odstránení zložitosti systému.

Extrémne programovanie

- Veľmi vplyvná agilná metóda vyvinutá koncom 90. rokov minulého storočia, ktorá zaviedla celý rad agilných vývojových techník.
- Extrémne programovanie (XP) používa „extrémny“ prístup k iteratívnemu vývoju.
  - Nové verzie môžu byť zostavené niekoľkokrát za deň;
  - Prírastky sa dodávajú zákazníkom každé 2 týždne;
  - Všetky testy musia byť spustené pre každý prírastok a verziu, dodávka je akceptovaná iba vtedy, ak testy prebehnú úspešne.

Proces vydania verzie v XP



Praktiky XP 1/2

Princíp alebo prax	Popis
Prírastkové plánovanie	Požiadavky sú zaznamenané na kartách príbehov a príbehy, ktoré majú byť zahrnuté do vydania, sú vybrané na základe času, ktorý je k dispozícii a ich relatívnou prioritou. Vývojári rozdelia tieto príbehy do vývojových úloh.
Malé vydania	Najprv sa vyvinie minimálna užitočná sada funkcií, ktorá poskytuje obchodnú hodnotu. Vydania systému sú časté a postupne pridávajú funkcie k prvému vydaniu.
Jednoduchý návrh	Vykoná sa akurát návrhu na splnenie súčasných požiadaviek a nič viac.
Najskôr testy	Automatizovaný rámec testovania jednotiek sa používa na písanie testov pre novú časť funkčnosti pred implementáciou samotnej funkcie.
Refaktorovanie	Od všetkých vývojárov sa očakáva, že budú kód neustále zjednodušovať hneď, ako sa nájdu možné vylepšenia kódu. Vďaka tomu je kód jednoduchý a udržiavateľný.

Praktiky XP 2/2

Programovanie v pároch	Vývojári pracujú vo dvojiciach, navzájom si kontrolujú prácu a poskytujú si podporu, aby vždy odviedli dobrú prácu.
Kolektívne vlastníctvo	Dvojice vývojárov pracujú na všetkých oblastiach systému, takže sa nevyvíjajú žiadne ostrovčeky odbornosti a všetci vývojári preberajú zodpovednosť za celý kód. Každý môže čokoľvek zmeniť.
Neustála integrácia	Akonáhle je práca na úlohe dokončená, je integrovaná do celého systému. Po každej takejto integrácii musia prejsť všetky testy jednotiek v systéme.
Udržateľné tempo	Veľké množstvo nadčasov sa nepovažuje za prijateľné, pretože čistým efektom je často zníženie kvality kódu a strednodobej produktivity
Zákazník v tíme	Zástupca koncového používateľa systému (zákazník) by mal byť k dispozícii na plný úväzok pre tím XP. V extrémnom procese programovania je zákazník členom vývojového tímu a je zodpovedný za predloženie systémových požiadaviek tímu na implementáciu .

XP a agilné princípy

- Postupný vývoj je podporovaný malými a častými systémovými vydaniami.
- Zapojenie zákazníka znamená zapojenie zákazníka do tímu na plný úväzok.
- Ľudia nie procesy prostredníctvom párového programovania, kolektívneho vlastníctva a procesu, ktorý sa vyhýba dlhým pracovným časom.
- Zmena podporovaná prostredníctvom častých verzií systému.
- Zachovanie jednoduchosti prostredníctvom neustáleho refaktorovania kódu.

Vplyvné praktiky XP

- XP má technické zameranie a nie je ľahké ho integrovať do manažérskej praxe vo väčšine organizácií.
- V dôsledku toho, zatiaľ čo agilný vývoj využíva postupy z XP, metóda XP, ako bola pôvodne definovaná, nie je široko používaná.
- Kľúčové praktiky
  - Príbehy používateľov pre špecifikáciu
  - Refaktorovanie
  - Najskôr testy
  - Zapojenie zákazníka
  - Párové programovanie

Používateľské príbehy pre požiadavky

- V XP je zákazník alebo používateľ súčasťou tímu XP a je zodpovedný za rozhodovanie o požiadavkách.
- používateľov sú vyjadrené ako používateľské príbehy alebo scenáre.
- Tie sú napísané na kartách a vývojový tím ich rozloží na implementačné úlohy. Tieto úlohy sú základom harmonogramu a odhadov nákladov.
- Zákazník si vyberá príbehy na zahrnutie do ďalšieho vydania na základe svojich priorít a odhadov plánu.

Príbeh „predpisovania liekov“.

Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either ‘current medication’, ‘new medication’ or ‘formulary’.

If you select ‘current medication’, you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

If you choose, ‘new medication’, the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose ‘formulary’, you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click ‘OK’ or ‘Change’. If you click ‘OK’, your prescription will be recorded on the audit database. If you click ‘Change’, you reenter the ‘Prescribing medication’ process.

Príklady kariet úloh na predpisovanie liekov

Task 1: Change dose of prescribed drug

Task 2: Formulary selection

Task 3: Dose checking

Dose checking is a safety precaution to check that the doctor has not prescribed a dangerously small or large dose.
Using the formulary id for the generic drug name, lookup the formulary and retrieve the recommended maximum and minimum dose.
Check the prescribed dose against the minimum and maximum. If outside the range, issue an error message saying that the dose is too high or too low.
If within the range, enable the 'Confirm' button.

Refaktorizácia

- Bežnou múdrosťou v softvérovom inžinierstve je pripraviť sa na zmenu. Stojí za to venovať čas a námahu predvídaniu zmien, pretože to znižuje náklady neskôr počas životného cyklu.
- XP však tvrdí, že to nestojí za to, pretože zmeny nemožno spoľahlivo predvídať.
- Skôr navrhuje neustále zlepšovanie kódu (refaktorizáciu), aby sa zmeny uľahčili, keď sa majú implementovať.

Refaktorizácia

- Programátorský tím hľadá možné vylepšenia softvéru a robí tieto vylepšenia aj tam, kde ich bezprostredne nepotrebujete.
- To zlepšuje zrozumiteľnosť softvéru a znižuje tak potrebu dokumentácie.
- Zmeny sa robia jednoduchšie, pretože kód je dobre štruktúrovaný a jasný.
- Niektoré zmeny si však vyžadujú refaktoring architektúry a to je oveľa drahšie.

Príklady refaktorizácie

- Reorganizácia hierarchie tried na odstránenie duplicitného kódu.
- Upratovanie a premenovanie atribútov a metód, aby boli ľahšie pochopiteľné.
- Nahradenie vloženého kódu volaniami metód, ktoré boli zahrnuté v knižnici programov.

Najskôr testy

- Testovanie je pre XP základ a XP vyvinulo prístup, pri ktorom sa program testuje po každej zmene.
- Testovacie funkcie XP:
  - Test - prvý vývoj.
  - Vývoj prírastkového testu zo scenárov.
  - Zapojenie používateľov do vývoja a overovania testov.
  - Automatizované testovacie zväzky sa používajú na spustenie všetkých testov komponentov vždy, keď sa vytvorí nové vydanie.

Testom riadený vývoj

- Písanie testov pred kódom objasňuje požiadavky, ktoré sa majú implementovať.
- Testy sú napísané ako programy a nie ako údaje, takže sa môžu vykonávať automaticky. Test zahŕňa kontrolu, či bol vykonaný správne .
  - Zvyčajne sa spolieha na testovacie rámce.
- Regresia: Všetky predchádzajúce a nové testy sa spustia automaticky, keď sa pridá nová funkcia, čím sa skontroluje, či nová funkcia nezaviedla chyby.

Zapojenie zákazníka

- Úlohou zákazníka v procese testovania je pomôcť pri vývoji akceptačných testov pre príbehy, ktoré sa majú implementovať v ďalšom vydaní systému.
- Zákazník, ktorý je súčasťou tímu, píše testy počas vývoja. Celý nový kód je preto validovaný, aby sa zabezpečilo, že je to to, čo zákazník potrebuje.
- Ľudia, ktorí si osvojujú rolu zákazníka, však majú k dispozícii obmedzený čas, a preto nemôžu pracovať na plný úväzok s vývojovým tímom. Môžu mať pocit, že poskytnutie požiadaviek bolo dostatočným prínosom, a preto sa môžu zdráhať zapojiť sa do procesu testovania.

Problémy s vývojom kde sú skôr testy ako kód

- Programátori uprednostňujú programovanie pred testovaním a niekedy pri písaní testov robia skratky. Môžu napríklad písať neúplné testy, ktoré nekontrolujú všetky možné výnimky, ktoré sa môžu vyskytnúť.
- Niektoré testy môže byť veľmi ťažké písať postupne. Napríklad v zložitom používateľskom rozhraní je často ťažké napísať testy jednotiek pre kód, ktorý implementuje „logiku zobrazenia“ a pracovný postup medzi obrazovkami.
- Je ťažké posúdiť úplnosť súboru testov. Aj keď môžete mať veľa systémových testov, vaša testovacia sada nemusí poskytovať úplné pokrytie.

Párové programovanie

- Pri párovom programovaní programátori sedia spolu pri jednom počítači, aby vyvinuli softvér.
- Páry sa vytvárajú dynamicky tak, aby všetci členovia tímu navzájom spolupracovali.
- Zdieľanie vedomostí, ku ktorým dochádza počas párového programovania, je veľmi dôležité, pretože znižuje celkové riziká pre projekt - keď členovia tímu odídu.
- Párové programovanie nie je nevyhnutne neefektívne a existujú dôkazy, že pár pracujúci spoločne je efektívnejší ako 2 programátori pracujúci oddelene.

Párové programovanie - výhody

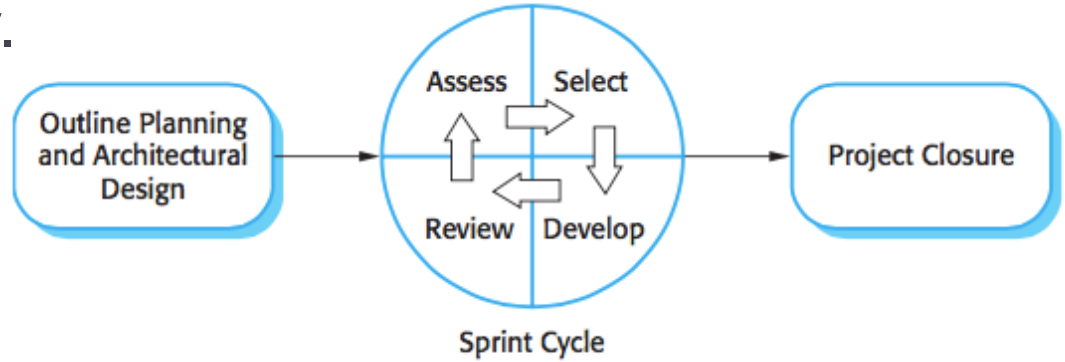
- Párové programovanie zahŕňa programátorov pracujúcich v pároch, ktorí spoločne vyvíjajú kód.
- Pomáha to rozvíjať spoločné vlastníctvo kódu a šíriť znalosti v rámci tímu.
- Slúži ako neformálny proces kontroly, pretože každý riadok kódu si prezerá viac ako 1 osoba.
- Podporuje refactoring, pretože celý tím môže mať prospech zo zlepšenia systémového kódu.

Agilné riadenie projektov

- Hlavnou zodpovednosťou manažérov softvérových projektov je riadiť projekt tak, aby bol softvér dodaný včas av rámci plánovaného rozpočtu na projekt.
- Štandardný prístup k riadeniu projektov je riadený plánom . Manažéri zostavia plán projektu, v ktorom je uvedené, čo by sa malo dodať, kedy by sa to malo dodať a kto bude pracovať na vypracovaní výstupov projektu.
- Agilné riadenie projektov si vyžaduje odlišný prístup, ktorý je prispôsobený postupnému vývoju a praktikám používaným v agilných metódach.

Scrum

- Scrum je agilná metóda, ktorá sa zameriava skôr na riadenie iteratívneho vývoja než na špecifické agilné postupy.



  - Počiatočná fáza je fáza plánovania, v ktorej stanovíte všeobecné ciele projektu a navrhnete softvérovú architektúru.
  - Séria cyklov - šprintov, kde každý cyklus vyvíja prírastok
  - Fáza ukončenia projektu uzatvára projekt, dokončuje požadovanú dokumentáciu, ako sú rámce pomoci systému a používateľské príručky, a hodnotí poznatky získané z projektu.

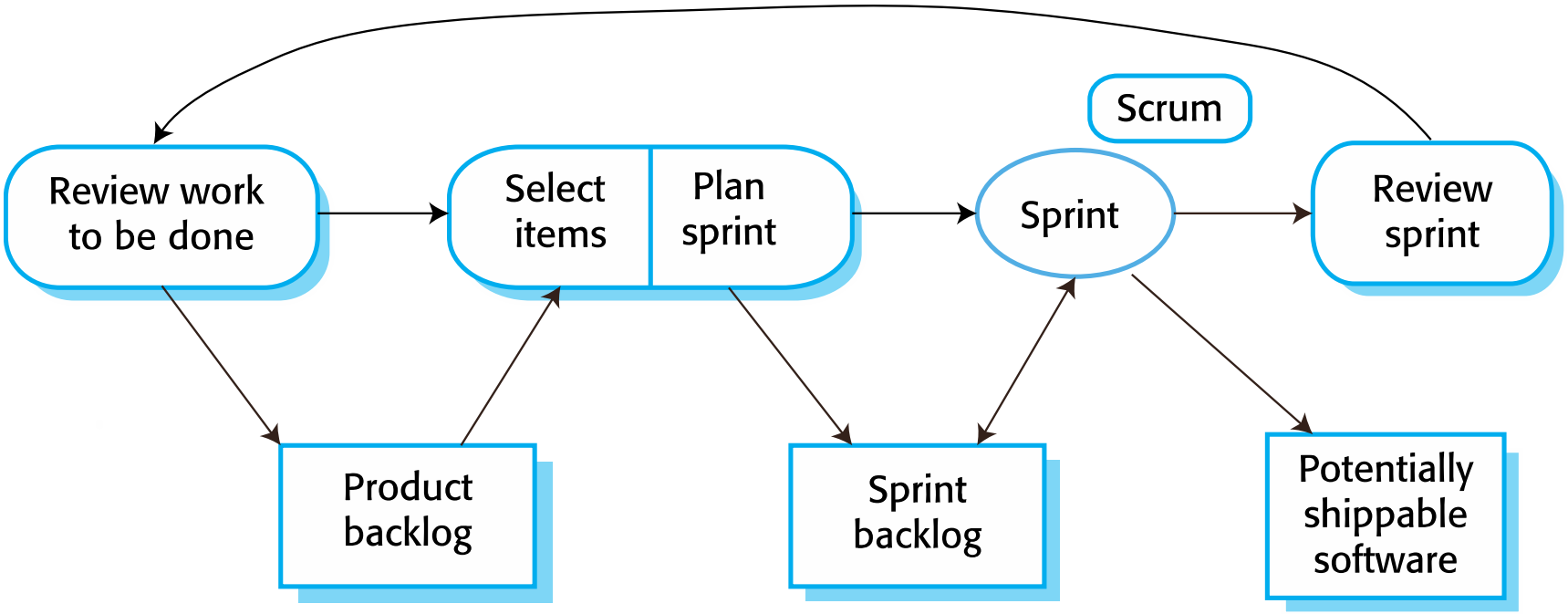
Terminológia scrumu 1/2

Scrum termín	Definícia
Vývojový tím	Samoorganizujúca sa skupina vývojárov softvéru, ktorá by nemala mať viac ako 7 ľudí. Sú zodpovední za vývoj softvéru a ďalších dôležitých projektových dokumentov.
Prírastok potenciálne odosielateľného produktu	Softvérový prírastok, ktorý sa dodáva zo sprintu. Myšlienkou je, že by mal byť „potenciálne odoslaný“, čo znamená, že je v hotovom stave a na jeho začlenenie do konečného produktu nie je potrebná žiadna ďalšia práca, ako napríklad testovanie. V praxi to nie je vždy možné dosiahnuť .
Produktový backlog	Toto je zoznam úloh, ktoré musí Scrum tím zvládnuť. Môžu to byť definície funkcií pre softvér, softvérové požiadavky, príbehy používateľov alebo popisy doplnkových úloh, ktoré sú potrebné, ako napríklad definícia architektúry alebo užívateľská dokumentácia .
Vlastník produktu	Jednotlivec (alebo prípadne malá skupina), ktorého úlohou je identifikovať vlastnosti alebo požiadavky produktu, uprednostniť ich pri vývoji a priebežne kontrolovať nevybavené produkty, aby sa zabezpečilo, že projekt naďalej spĺňa kritické obchodné potreby. Vlastník produktu môže byť zákazník, ale môže to byť aj produktový manažér v softvérovej spoločnosti alebo iný zástupca zainteresovaných strán.

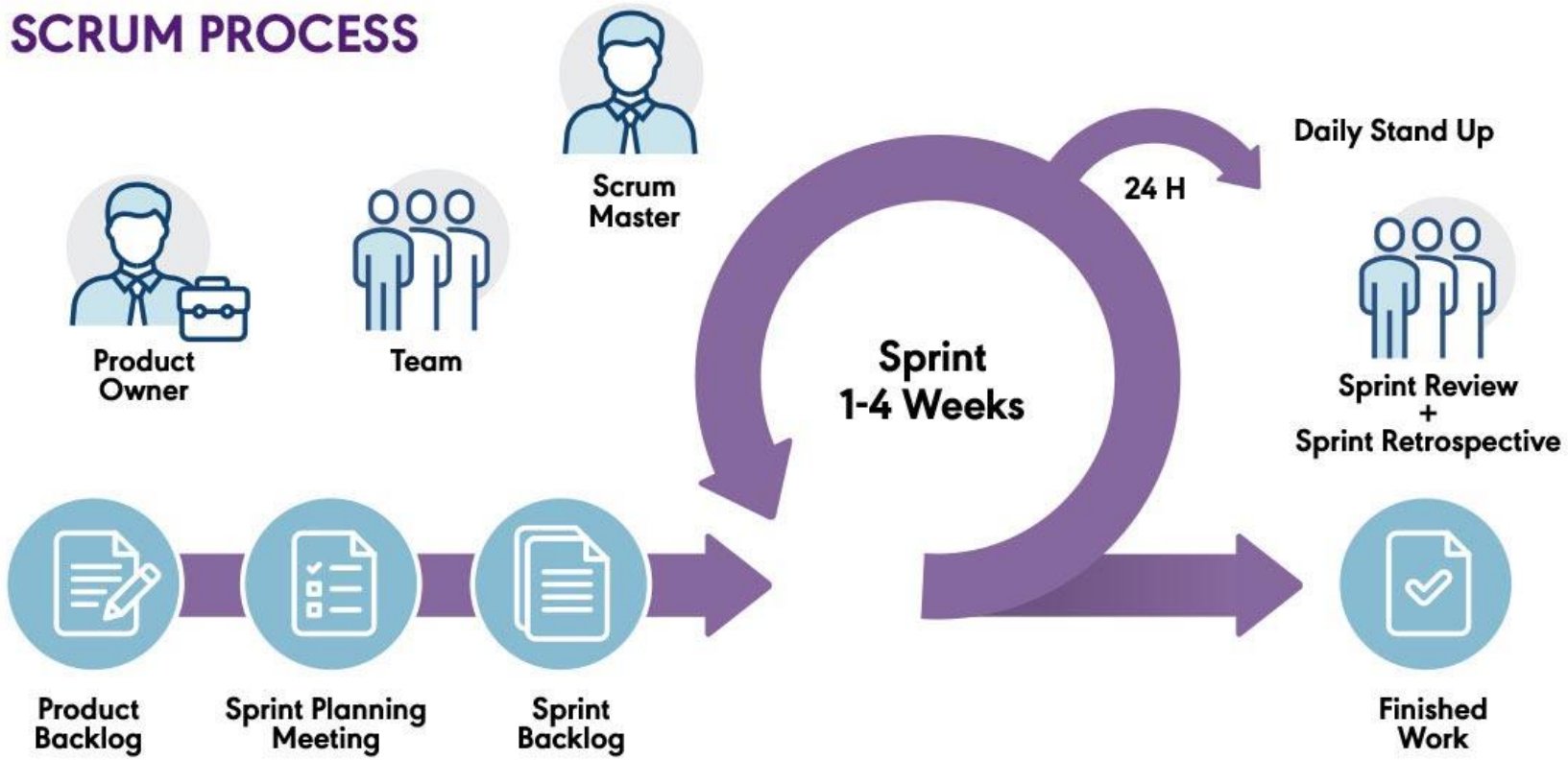
Terminológia scrumu 2/2

Scrum termín	Definícia
Scrum	Denné stretnutie Scrum tímu, ktoré hodnotí pokrok a určuje priority práce, ktorá sa má v daný deň vykonať. V ideálnom prípade by to malo byť krátke osobné stretnutie, na ktorom sa zúčastní celý tím .
ScrumMaster	ScrumMaster je zodpovedný za zabezpečenie dodržiavania procesu Scrumu a vedie tím pri efektívnom používaní Scrumu . Je zodpovedný za spojenie so zvyškom spoločnosti a za zabezpečenie toho, aby tím Scrum nebol odklonený vonkajším zásahom. Vývojári Scrumu sú pevne presvedčení, že ScrumMaster by sa nemal považovať za projektového manažéra. Iní však nemusia vždy ľahko vidieť rozdiel .
Šprint	Vývojová iterácia. Šprinty zvyčajne trvajú 2-4 týždne.
Rýchlosť	Odhad, koľko úsilia v oblasti nevybavených produktov môže tím pokryť v jednom sprinte. Pochopenie rýchlosti tímu im pomáha odhadnúť, čo je možné pokryť v šprinte, a poskytuje základ pre meranie zlepšovania výkonu .

Cyklus scrum šprintu



Metóda Scrum



Cyklus scrumu

- Šprinty majú pevnú dĺžku, zvyčajne 2-4 týždne.
- Východiskovým bodom pre plánovanie je produktový backlog, čo je zoznam prác, ktoré treba na projekte vykonať.
- Fáza výberu zahŕňa celý projektový tím, ktorý spolupracuje so zákazníkom na výbere nových funkcií a funkcií z produktového backlogu, ktoré sa majú vyvinúť počas šprintu.

Šprintový cyklus

- Po odsúhlasení sa tím zorganizuje na vývoj softvéru.
- Počas tejto fázy je tím izolovaný od zákazníka a organizácie, pričom všetka komunikácia prebieha cez takzvanej 'Scrum master'.
- Úlohou Scrum mastera je chrániť vývojový tím pred vonkajšími rušivými vplyvmi.
- Na konci šprintu je vykonaná práca skontrolovaná a prezentovaná zainteresovaným stranám. Potom začne ďalší šprint.

Tímová práca v Scrume

- ' Scrum master ' je facilitátor, ktorý organizuje denné stretnutia, sleduje nevybavenú prácu, zaznamenáva rozhodnutia, meria pokrok oproti nevybaveným a komunikuje so zákazníkmi a manažmentom mimo tímu.
- Celý tím absolvuje krátke denné stretnutia (Scrums), kde si všetci členovia tímu vymieňajú informácie, popisujú svoj pokrok od posledného stretnutia, problémy, ktoré sa vyskytli a čo sa plánuje na nasledujúci deň.
  - To znamená, že každý v tíme vie, čo sa deje, a ak sa vyskytnú problémy, môže preplánovať krátkodobú prácu, aby sa s nimi vyrovnal.

Výhody scrumu

- Produkt je rozdelený na súbor zvládnuteľných a zrozumiteľných častí.
- Nestabilné požiadavky nebrzdia pokrok.
- Celý tím má o všetkom prehľad a následne sa zlepšuje tímová komunikácia.
- Zákazníci vidia včasné dodanie prírastkov a získajú spätnú väzbu o tom, ako produkt funguje.
- Vytvorí sa dôvera medzi zákazníkmi a vývojármi a vytvorí sa pozitívna kultúra, v ktorej každý očakáva úspech projektu.

Distribuovaný Scrum