30.Životní cyklus softwaru

Rozděluje proces vývoje softwaru na za sebou jdoucí období, přičemž pro každé období stanovuje cíl. Existuje několik metodologických přístupů, které se navzájem liší v jednotlivých etapách a jejich dělení. Podstata těchto přístupů je vyjádřena v modelech životního cyklu.

Etapy životního cyklu softwaru

Analýza a specifikace požadavků

Je úvodní etapou ve vývoji softwaru. Má za úkol transformovat neformální popis požadavků od uživatele do strukturovaného popisu. Jejím cílem je tedy stanovení služeb, které zákazník od systému požaduje a vymezuje podmínky a omezení jeho vývoje a provozu. Proces tvorby požadavků můžeme rozložit do tří kategorií:

- získávání požadavků
- analýza požadavků
- definice požadavků

Pozornost v této etapě je důležité věnovat samotným požadavkům a ne jejich realizaci.s Samotné požadavky mohou být rozřazeny do několika skupin/typů (např. Funkcionální, na provoz, na rozhraní,...). Specifikaci požadavků je možné vizualizovat pomocí různých modelů (Data Flow Diagram, Use Case Diagram,...).
Důležitou součástí této etapy je studie vhodnosti, která odpoví na otázku zda má smysl se do projektu vůbec poštět. V této etapě je možné také identifikovat rizika a provést jejich analýzu. Nezbytným výstupem je plán akceptačního testování.

Architektonický a podrobný návrh

Architektonický návrh navazuje na analýzu požadavků a slouží k ujasnění koncepce celého systému a jeho

dekompozici. Definují se vztahy mezi jednotlivými částmi systému. Je specifikována funkcionalita a ohraničení podsystémů. V této části taky plánujeme testování celého systému a plán jeho nasazení včetně zaškolení uživatelů.

Podrobný návrh detailně specifikuje softwarové současti. Popisuje použité algoritmy, logické a fyzické struktury dat, rozhraní a způsob ošetřování chybových a neočekávaných stavů. Plánuje se implementace součástí a s tím také souvisí vypracování požadavků na lidské zdroje, dobu trvání a náklady. Zahrnuje také plán testování jednotlivých součástí.

Implementace

Transformace návrhu jednotlivých modulů a jejich vazeb do programové realizace. Zahrnuje také vytvoření programové dokumentace a otestování implementovaných součástí.

Integrace a testování

Spojení jednotlivých součástí do podsystémů. Testování těchto podsystémů. Po otestování a opravení chyb v podsystémech dochází k integraci podsystémů do celého systému, poté se testuje celý systém. Při testování a opravě chyb se vracíme k etapě implementace.

Provoz a údržba

Finální etapa. Jedná se o samotné nasazení softwaru a s ním spojené řešení problémů s nasazením. Řeší se zde také problémy s používáním. Etapa také zahrnuje opravy, rozšiřování a přizpůsobování softwaru podle požadavků okolí.

Akceptační testování

Spočívá v otestování systému uživatelem.Na základě akceptačního testování se pak zákazník rozhodne buď systém převzít, nebo při objevení závažnějších nedostatků je převzetí odloženo, dokud dodavatel chyby neodstraní. Po převzetí softwaru zákazníkem začíná instalace softwarového systému u zákazníka a školení uživatelů.

Modely životního cyklu

Definuje etapy vývoje softwaru a jejich časovou následnost. Pro každou etapu definuje nutné činnosti, její vstupy a výstupy. Rozlišujeme dva základní přístupy k procesu vývoje:

- lineární modely
- iterativní modely

Vodopádový model životního cyklu softwaru

Základním modelem životního cyklu softwaru je vodopádový model. Patří mezi lineární modely. Etapy jsou seřazena za sebou a následující etapa začne až po ukončení předcházející. Je zde možnost návratu k předchozí etapě. Tento model je přirozený a patřík k nejstarším modelům. Uživatel se zde účastní pouze definování požadavků a zavádění.

Výhody tohoto modelu spočívají v jednoduchém řízení. Model také při stálých požadavcích má nejlepší sktrukturu výsledného produktu.

Nevýhodou systému je skutečnost, že zákazník není schopen předem přesně stanovit všechny požadavky. Při změně požadavků tak je nutné projít celý proces znovu, což ovlivňuje výsledný produkt. S tímto také souvísí to, že zákazník vidí produkt až ve finální fázi a není tedy schopen jeho vývoj nějak ovlivňovat.

Iterativní modely

Systém se vyvíjí v iteracích. V každé iteraci se tvoří reálný výsledek. Předpokladem je to, že se zákazník účastní vývoje.

Výhodou těchto modelů je to, že po každé iteraci je k dispozici nějaký výsledek, který je možno předvést zákazníkovi. Dochází tak k rychlejšímu odhalení chyb ve specifikaci.

Nevýhodou je vyšší náročnost řízení. A výsledný produkt ma potenciálně horší výslednou strukturu. Tento nedostatek lze odstranit, nebo zmírnit refaktorizací.

Prototypování

Není to ucelený model. Bývá součástí jiných modelů a či metodik.
Prototyp je částečná implementace produktu, která prezentuje vnější rozhraní systému. Prototyp se již dále nepoužívá.

Inkrementální model

Kombinace lineárního a iterativního typu. Na základě specifikace celého systému se stanoví ucelené části systému. Může být realizováno například jako série vodopádů pro jednotlivé části systému, které jsou po dokončení prezentovány zákazníkovi.

Výhodou je možnost navrhnutí lepší struktury, než v případě iterativního modelu, aniž by pozdější upřesnění požadavků uživatelem mělo negativní dopady jako u vodopádového modelu.

Nevýhodou je, že vývoj po částech může vést ke ztrátě vnímání logiky a technických požadavků celého systému.

Spirálový model

Kombinuje prototypování a klade důraz na analýzu rizik. Jednotlivké etapy se ve vývoji softwaru opakují, vždy na vyšším stupni zvládnuté problematiky. Vyžaduje stálou spolupráci se zákazníky. Uživatel dostává k odzkoušení prototyp, který se po použití zahodí.

Výhodou tohoto modelu je jeho využití pro složitější projekty. Díky analýze rizik jsme schopni dříve odhalit chyby a nevyhovující postupy.

Nevýhodou je vysoká závislost na analýze rizik, která musí být prováděna na vysoké odborné úrovní. Vyžaduje precizní kontroly výstupů. Software je k dispozici až po poslením cyklu. Obsahuje mezníky, které se stanoví jako kontrolní body.

RAD

Rychlý iterativní vývoj prototypů. Funkční verze jsou k dispozici dříve než u jiných modelů. Je zde potřeba intenzivního zapojení zákazníka do vývojového procesu. Metodika má schopnost rychlé změny podle potřeb zákazníka. Výsledek nemusí být tak kvalitní. Upřednostňuje business pořeby před technologickými a inženýrskými.

RUP - Rational Unified Process

Na rozdíl od jiných modelů RUP není pouze koncepcí, ale je použitelný pro řízení reálných softwarových projektů. Spíše než konkrétní metodika je chápan jako rozšířitelný framework, který by měl být uzpůsoben organizaci nebo projektu. Založen na vývoji projektu iteračním způsobem(po každé iteraci je k dispozici spustitelný kód). Klade důraz na vizualizaci systému(UML) a průběžnou kontorlu kvality produktu. Cyklus je rozdělen na 4 fáze:

- zahájení
- projektování
- realizace
- předání

Tento model také obsahuje mezníky podobně jako spirálový model.

Agilní metodologie

Agilní metodologie byly vyvinuty jako protiklad k mohutným modelům. Tyto modely postupem času zmohutněly a pro potřeby menších projektů byly naprosto nevhodné. Kvalitu výsledného produkt přitom nezvyšovaly a cena takového projektu se pouze zvyšovala. Odpovědí na tyto problémy jsou právě agilní metodologie, které kladou důraz na člověka jako určující faktor pro kvalitu výsledného produktu.