Spiral model a jeho použitie pri modelovaní softvéru*

Tomáš Andel
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xandelt1@stuba.sk

30. september 2021

Abstrakt

Spiral model je metóda na vývoj softvéru. Vývoj pomocou tohto modelu sa vyznačuje tým, že kladie dôraz na redukciu riskov. Článok predstaví vývoj softvéru, niektoré tradičné modely, ich problémy a ako ich Spiral model rieši. Analyzuje postup vývoja pomocou Spiral modelu, porovná jeho výhody a nevýhody a predstaví ako vznikal a ako sa vyvíjal. Preskúma aké spôsoby modelovania sa využívajú pri vývoji touto metódou. Pozrie sa na rôzne možnosti jeho využitia a kde konkrétne sa používa. Predstaví novinky v tejto oblasti ako napríklad aké sú iné modely, ktoré sú založené na Spiral modeli alebo aké kombinácie Spiral modelu a iných modelov sa dnes využívajú na optimalizáciu vývoja softvéru.

1 Úvod

Vývoj softvéru je zložitý proces. Tvorba nápadu, určenie požiadaviek, dizajn, implementácia, testovanie, údržba - tieto všetky a ďalšie procesy nastávajú pri vývoji softvéru, bez jasného plánu môže byť vývoj zdĺhavý a drahý. Už od možnosti vývoja prvého softvéru ľudia hľadajú modely, podľa ktorých budú pracovať, a ktoré im umožnia efektívne vyvíjať komplexné softvéry.

Existuje množstvo vývojových modelov, niektoré sú vhodnejšie na menšie projekty, pričom sú nepoužiteľné vo veľkých, kde by vývoj postupoval neefektívnym spôsobom. Iné sú zasa vhodnejšie na veľké projekty, ale pri malých projektoch by bolo ich použitie zbytočne komplikované. Tento článok ukáže vývoj skôr veľkých projektov a prečo je na to výbornou voľbou práve Spiral model.

V časti 2 sa pozrieme na životný cyklus vývoja softvéru a aké fázy obnáša. V časti 2.1 bude predstavených niekoľko tradičných vývojových modelov a budú ukázané ich nevýhody a ako ich Spiral model rieši. V ďalšej časti 3 prejdeme na samotný Spiral model. Ukážeme si ako funguje, ako prebieha riešenie objavených rizík v časti 3.1, ako vznikal v časti 3.2 a kde sa využíva v časti 3.3. Na záver 2 budú zhrnuté získané poznatky.

*Semestrálny projekt v predmete Metódy inžinierskej práce, ak. rok 2021/22, vedenie: Ing. Vladislav Mlynarovič, PhD.

2 Vývoj softvéru

Softvéroví inžinieri sa vždy snažia vyprodukovať kvalitný produkt, ktorý zodpovedá požiadavkám klienta, neprekročí pridelený rozpočet a je vyprodukovaný načas. Bohužial, často tieto ciele nie sú dosiahnuté. Ale dobrý manažment projektu vo vhodnom prostredí, ktoré sa drží zaužívaných metód a modelov môže zaručiť konzistené dosahovanie týchto cielov. [3]

Životný cyklus vývoja softvéru je proces tvorby softvéru, ktorého cielom je finálny produkt čo najvyššej kvality a čo najnižšej ceny [1]. Väčšinou obnáša tieto fázy:[5]

1. **Iniciácia/Počiatočné plánovanie** - Diskusia o realizovateľnosti projektu, počiatočná estimácia nákladov a časovej náročnosti.
2. **Analýza požiadaviek a špecifikácia** - Identifikácia problémov, ktoré má vyvíjaný softvér riešiť a ich špecifikácia. Riešenie jeho operačných schopností, výkonostných charakteristík a infraštruktúry potrebnej na jeho údržbu.
3. **Funkčná špecifikácia alebo prototypovanie** - Identifikácia objektov, ich vlastnosti a vzťahy. Identifikácia obmedzení, ktoré obmedzujú správanie systému ať d.
4. **Rozdelenie systémov (Build vs. Buy vs. Reuse)** - Rozdelenie systému na menšie časti a zistiť, ktoré je vhodné vyrobiť, ktoré stačí odkúpiť a nakonfigurovať k požiadavkám systému, a ktoré je možné znova použiť z predošlých projektov.
5. **Architektúra** - Definuje prepojenia a vytvára rozhrania medzi jednotlivými podsystémami, komponentami a modulmi aby bol možný ich jednotlivý detailný dizajn.
6. **Detailný dizajn komponentov** - Definuje funkciu jednotlivých komponentov, ich interné správanie, ako transformujú vstupy na požadované výstupy.
7. **Implementácia komponentov** - Kodifikuje špecifikácie navrhnuté počas architektúry a detailného dizajnu komponentov do funkčného zdrojového kódu.
8. **Testovanie a kontrola integrity** - Kontrola systému a podsystémov na základe ich požiadaviek. Kontroluje celkovú integritu systému.
9. **Tvorba dokumentácie** - Vytváranie systematických dokumentov a užívatelských príručiek.
10. **Spustenie softvéru pre klienta/užívateľa** - Poskytnutie softvéru užívatelovi a prípadne inštrukcie na inštaláciu a konfiguráciu.
11. **Školenie a používanie softvéru** - Školenie užívatelov systému ako správne a efektívne softvér využívať.
12. **Údržba softvéru** - Udržovanie operatívnosti softvéru opravovaním objavených chýb, poskytovaním funkčných alebo výkonnostných vylepšení, údržbou podpornej infraštruktúry.

2.1 Metódy vývoja softvéru

Metódy alebo modely na vývoj softvéru popisujú spôsob navigácie medzi vyššie uvedenými procesmi vývoja.[6]

Existuje veľké množstvo modelov a každý z nich je vhodný v inej situácií. Pozrime sa na niekoľko tradičných modelov [3]:

- **Waterfall Model** - Jeho grafická reprezentácia vyzerá ako kaskáda vodopádov. Je to jeden z najstarších modelov a je základom mnohých iných modelov. Pri používaní tohto modelu je každý proces najprv dokončený a až potom sa prechádza na ďalší. Jeho nevýhodou je nízka úroveň flexibility, všetky požiadavky je potrebné vedieť už na začiatku. Nie je vhodný na veľké projekty. Analýza rizík je často zložitá.

- **Sashimi Model** - Waterfall model, pri ktorom je dovolená paralelná práca na viacerých fázach naraz. Je časovo výhodnejší.

- **V-Shaped Model** - Je považovaný za rozšírenie Waterfall modelu. Namiesto lineárneho pohybu nadol sa vytvára tvar V s vrcholom, v ktorom je implementačný proces. Model vytvára vzťahy medzi vývojovými procesmi pred implementáciou a ich príslušnými fázami testovania. Výhodou tohto modelu jednoduché použitie rovnako ako pri Waterfall modeli a narozdiel od Waterfall modelu je tu vyššia šanca úspechu kvôli plánovaniu testovania už na začiatku vývoja. Nevýhodou je nízka úroveň flexibility a nemožnosť tvorby skorých prototypov pretože všetká produkcia kódu je až vo fázi implementácie.

Ako je možné vidieť, nevýhodami týchto modelov sú hlavne nízka flexibilita a zlá analýza rizík. Pri vývoji veľkých projektov častov vznikajú rôzne riziká, ktoré sa pri tých menších objavujú menej. Preto veľké firmy hľadajú iné modely, pomocou ktorých tieto riziká vedia odhaliť a uskutočniť určité kroky na ich spracovanie.

V ďalšej časti sa pozrieme na Spiral model a ako dokáže tieto problémy riešiť.

3 Spiral model

Spiral model, pôvodne navrhnutý Barrym W. Boehmom, je evolučný softvérový procesný model, ktorý spája iteračnú vlastnosť prototypovania s riadenými a systematickými aspektmi lineárneho sekvenčného modelu. [2]

Jeho diagram vyzerá ako špirála s mnohými slučkami (viď. obrázok). Každá slučka špirály sa nazýva fáza vývojového procesu softvéru. Presný počet fáz závisí od zložitosti projektu. Projektový manažér analyzuje riziká a iné faktory, podľa ktorých dynamicky určuje počet týchto fáz.

Čím je väčší počet fáz, tým sú väčšie náklady projektu - polomer špirály predstavuje celkové náklady projektu. Uhol medzi pozorovaným miestom a začiatkom fázy predstavuje dosiahnutý postup v danej fáze. [4]

Každá fáza je rozdelená do štyroch kvadrantov, ako je znázornené na obrázku. Kvadranty sú [4]:

1. **Určovanie cieľov, alternatív, obmedzení** - V tomto kvadrante prebieha zbieranie požiadaviek od klientov, analýza a identifikácia cieľov a

potom diskusia ohľadom alternatívnych riešení, ktoré by mohli byť použité v ďalších častiach súčasnej fázy.

2. **Zhodnotenie riešení, identifikácia a riešenie rizík** - V druhom kvadrante sú všetky možné riešenia zhodnotené a vyberie sa najlepšie riešenie. Ďalej prebieha analýza rizík spojených s vybraným riešením, tie sú následne riešené pomocou najvhodnejšej stratégie. Na konci tohto kvadrantu je zhotovený prototyp, ktorý je v súlade s vybranou najlepšou stratégiou.

3. **Vývoj a testovanie produktu ďalšej verzie** - V treťom kvadrante prebieha všetká implementácia a testovanie. Určené špecifikácie sú vytvárané a testované. Na konci tohto kvadrantu je pripravená ďalšia verzia produktu.

4. **Plánovanie ďalšej fázy** - Klient zhodnotí vytvorený produkt a prebieha plánovanie na ďalšiu fázu.

Na doplnenie.

3.1 Riešenie rizík pomocou Spiral modelu

Na doplnenie.

3.2 Vznik Spiral modelu

Pred prvou definíciou Barrym Boehmom v roku 1986 sa Spiral model vyvíjal niekoľko rokov. Prvé implementácie vznikali na základe rôznych úprav Waterfall modelu aplikovaného na veľké vládne projekty.

Na doplnenie.

3.3 Využitie Spiral modelu

Na doplnenie.

3.4 Win-Win Spiral model

Na doplnenie.

3.5 Problemy Spiral modelu a ich riešenia

Na doplnenie.

4 Záver

Na doplnenie.

Literatúra

[1] Harness. Understanding the phases of the software development life cycle. `https://harness.io/blog/software-development-life-cycle/`, 2021.

[2] S. Jaiswal. Spiral model. `https://www.javatpoint.com/software-engineering-spiral-model`, 2019.

[3] G. Kumar and P. K. Bhatia. Comparative analysis of software engineering models from traditional to modern methodologies. In *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pages 189–196, 2014.

[4] S. K. Pal. Software engineering — spiral model. `https://www.geeksforgeeks.org/software-engineering-spiral-model/`, 2021.

[5] W. Scacchi. Process models in software engineering. In *J.J. Marciniak (ed.), Encyclopedia of Software Engineering, 2nd Edition, John Wiley and Sons, Inc, New York, December 2001.*, Institute for Software Research, University of California, Irvine, Oct. 2001.

[6] B. Shiklo. Software development models: Sliced, diced and organized in charts. `https://www.scnsoft.com/blog/software-development-models`, 2019.