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 popíše rôzne metódy na vývoj softvéru a predvedie čím sa Spiral model vyznačuje. Analyzuje postup vývoja pomocou Spiral modelu, porovná jeho výhody a nevýhody s inými vybranými modelmi 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.

V časti [] je základné predstavenie vývoja softvéru a prvotných metód (modelov). Popísanie Spiral modelu sa nachádza v časti [].

...

Motivujte čitateľa a vysvetlite, o čom píšete. Úvod sa väčšinou nedelí na časti.

Uveďte explicitne štruktúru článku. Tu je nejaký príklad. Základný problém, ktorý bol naznačený v úvode, je podrobnejšie vysvetlený v časti **??**. Dôležité súvislosti sú uvedené v častiach **??** a **??**. Záverečné poznámky prináša časť 4.

*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. [5]

Životný cyklus vývoja softvéru je proces tvorby softvéru, ktorého cielom je finálny produkt čo najvyššej kvality a čo najnižšej ceny [4]. Väčšinou obnáša tieto fázy: [6]

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. [7]

Existuje veľké množstvo modelov a každý z nich je vhodný v inej situácií. Pozrime sa na niekoľko tradičných modelov [5]:

- **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. Spiral model tieto problémy rieši.

3 Spiral model

Základným problémom je teda... Najprv sa pozrieme na nejaké vysvetlenie (časť **??**), a potom na ešte nejaké (časť **??**).[1]

Môže sa zdať, že problém vlastne nejestvuje [1], ale bolo dokázané, že to tak nie je [2, 3]. Napriek tomu, aj dnes na webe narazíme na všelijaké pochybné názory [**?**]. Dôležité veci možno *zdôrazniť kurzívou*.

Veľmi dôležitá poznámka. Niekedy je potrebné nadpisom označiť odsek. Text pokračuje hneď za nadpisom.

4 Záver

Literatúra

[1] J. O. Coplien. *Multi-Paradigm Design for C++*. Addison-Wesley, 1999.

[2] K. Czarnecki, S. Helsen, and U. Eisenecker. Staged configuration through specialization and multi-level configuration of feature models. *Software Process: Improvement and Practice*, 10:143–169, Apr./June 2005.

[1] Niekedy môžete potrebovať aj poznámku pod čiarou.

[3] K. Czarnecki and C. H. P. Kim. Cardinality-based feature modeling and constraints: A progress report. In *International Workshop on Software Factories, OOPSLA 2005*, San Diego, USA, Oct. 2005.

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

[5] 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.

[6] 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.

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