AKO AGILNÝ VÝVOJ NAHRADIL STARÉHO DINOSAURA

Perfektný plán neexistuje, ale so snahou sa k tomu môžeme priblížiť.

Eduard Fritscher

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
eduard.fritscher[zavináč]gmail[.]com

Abstrakt. *V softvérových projektoch, tak ako aj v projektoch z iných oblastí, hrá dôležitú rolu plánovanie. Správne odhady prispievajú k tomu, aby projekt skončil úspešne. V mojej eseji som sa zameral na porovnanie tradičných metód odhadovania, respektíve procesov pri plánovaní na základe vodopádového modelu oproti novým modelom a metód, ktoré sa používajú pri agilnom vývoji. Analyzoval som stav v manažmente plánovania a sústredil som sa najmä na odhadovanie vo vodopádovom modeli a v agilných metódach. Porovnal som metódy a techniky v odhadovaní ako aj v plánovaní. Snažil som sa poukázať na to, ako sa zmenilo odhadovanie pri projektoch a jednotlivých komponentoch s príchodom agilných metodík. Ďalej sa v eseji snažím zdôrazňovať potrebu toho, aby boli do manažmentu plánovania zainteresované aj vývojové tímy a zdôrazňujem aj benefity tohto spôsobu odhadovania.*

Kľúčové slová: *plánovanie, manažment, agilné metódy, odhad*

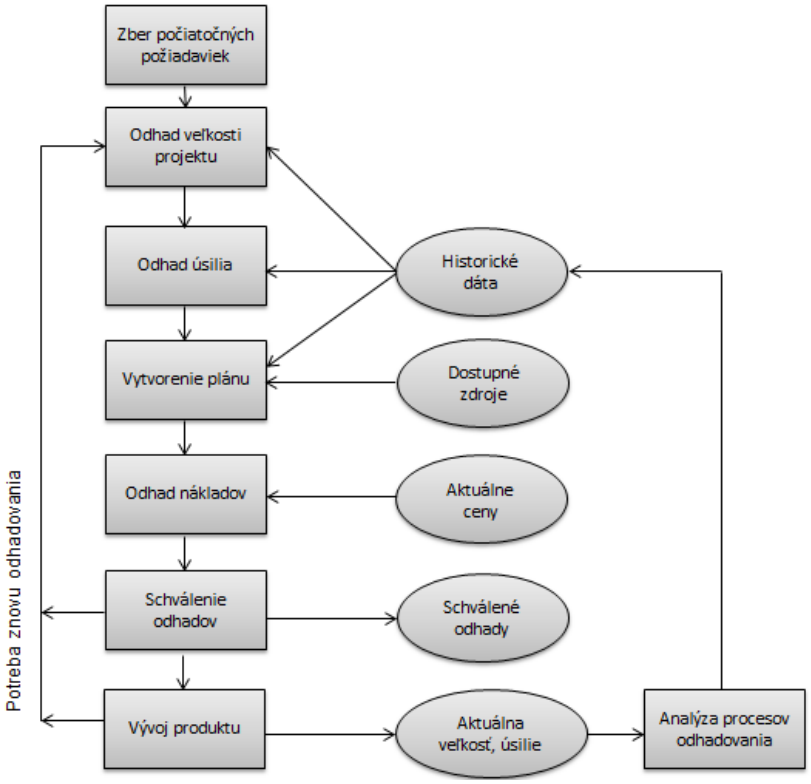
Úvod

Plánovanie bolo vždy v softvérovom vývoji jedným z najdôležitejších a najťažších krokov. Veľké softvérové firmy dlho používali vodopádový model a teraz sa už tvária že sa posunuli, ale pravda je taká, že niektoré firmy ešte stále ostali pri starých zvykoch. Časy v softvérovom inžinierstve sa však menia. Tak ako sa mení svet, tak sa musel zmeniť aj proces plánovania a príchod agilných metodík umožnil túto zmenu. Podľa môjho názoru

práve agilné metodiky umožnili, aby sa plánovanie stalo jednoduchšou a efektívnejšou záležitosťou.

Procesy pri plánovaní projektov

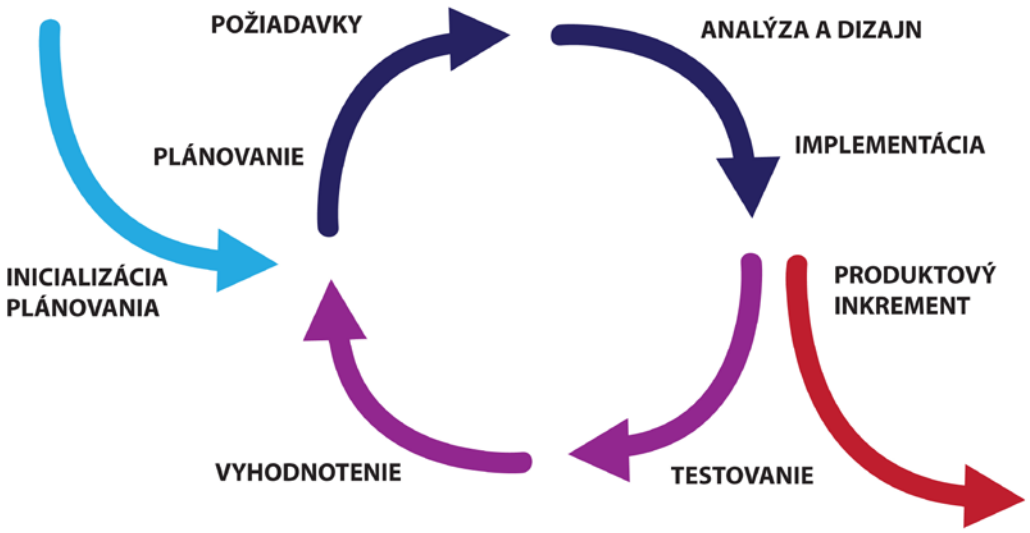
Procesy pri plánovaní projektov sú takmer tak dôležité ako samotný vývoj. Predstavte si, že máte špičkových vývojárov, ktorí pracujú na projekte, ale keďže neexistuje projektový plán a koordinácia medzi vývojármi, pravdepodobne je projekt odsúdený na zlyhanie. Plán pre softvérový produkt je vlastne mapa, ktorá ukazuje cestu do cieľa, čo v našom prípade je úspešné ukončenie projektu. Proces pri plánovaní projektov pri vývoji softvéru je vlastne metóda používaná v softvérovom priemysle. Pre správne naplánovanie projektu je asi najdôležitejšie to, aby boli odhady správne. Podľa Mehwishra existujú štyri hlavné kroky pri plánovaní projektu. Tieto hlavné kroky sú výpočet veľkosti softvéru, odhad pracovných dní (man day) podľa veľkosti softvéru, výpočet nákladov a rozpočtu, správne naplánovanie úloh a alokácia zdrojov.[1] S týmito hlavnými krokmi plne súhlasím, keďže bez nich by sme nemohli hovoriť o reálnom projekte.



Obr. 1 Základné procesy plánovania softvérového projektu vo vodopádovom modeli [1].

Na obrázku 1 máte možnosť vidieť základné procesy plánovanie pri vodopádovom modeli. Dlho sa používali a neboli s nimi väčšie problémy, ale ja vidím obrovské problémy. Vidím v tom byrokraciu, ktorá ja potrebná pri malej zmene. Chýba tam tá flexibilita čo opisuje náš svet v dvadsiatom-prvom storočí.

Tieto procesy a tento postup sa používal dlho, avšak do softvérového inžinierstva prišla nová éra a požiadavky pre vývojové tímy sa zmenili. Vodopádový model vo vývoji softvéru sa stal dinosaurom v tejto oblasti. Odpoveďou na túto problematiku bol agilný vývoj, ktorý sa veľmi rýchlo stal populárnym. Agilný vývoj bol pôvodne vymyslený pre malé tímy, ale časom sa ukázalo, že je dobre aplikovateľný a škálovateľný aj pre veľké tímy. Môj osobný názor je taký, že agilný vývoj mal rovnako veľký dopad pre oblasť vývoja softvéru ako kedysi prvé, plne automatizované, výrobné pásy pre automobilový priemysel. Proces plánovania pri agilnom vývoji sa veľmi líši od vodopádového modelu, ako to vyplýva aj z obrázku číslo 2.



Obr. 2 Základné procesy softvérového projektu pri agilnom vývoji.

Mne osobne sa najviac páči na agilnom vývoji práve to, čo je v momente jasné pre každého, kto sa pozrie na obrázok číslo 2. Celý softvérový projekt je vďaka tejto metóde oveľa dynamickejší a viac odolný k zmenám. Práve preto, lebo po každej iterácií sa môžu zmeniť požiadavky zákazníka a vývojové tímy majú na to možnosť rýchlo reagovať. Pri agilnom vývoji dokážeme ľahšie odhaliť zle zachytené požiadavky, keďže po každej iterácií sa odovzdáva produktový inkrement. Ten môže zákazník okamžite pozrieť a hneď sa k nemu vyjadriť. Tieto veci riešia problémy, ktorými softvérový priemysel už dlho trpí. Problémy a zlé zachytené požiadavky, neskoré odhalenie rizík alebo málo spätnej väzby od zákazníka. Ja si osobne myslím že práve v tomto spočíva krása agilného vývoja. Tie problémy som pocítil na vlastnej koži pri práci vo firme, keď nie raz sa stalo, že manažment zle zachytil požiadavky a celý čas sme vyvíjali niečo, čo reálne ani nebolo potrebné. Zvyčajne sa nato došlo až pri testovaní. Keby sme boli využívali agilný vývoj, tak pri prvom produktovom inkremente by sa zistilo, že špecifikácia je zlá a ušetrilo by nám to veľa cenných dní.

Aké spôsoby existujú pre odhadovanie?

Odhad časovej náročnosti softvérového projektu je jedna z najzložitejších aktivít, pretože ho ovplyvňuje množstvo rôznych faktorov. Pripraviť odhad tak, aby čo najvernejšie zodpovedal budúcnosti je teda netriviálna záležitosť. Našťastie existujú postupy, ako pripraviť čo najpresnejší odhad.

Metódy odhadovania pri vodopádovom vývoji [1]:

- Analogická metóda spočíva v tom, že projekt sa snažíme odhadnúť tak, že ho porovnávame s dokončeným projektom podobného typu. Pritom nám pomáhajú historické dáta zo skôr dokončených projektov. Avšak to funguje iba vtedy, keď sú k dispozícii predchádzajúce dáta. Je potrebná systematicky udržiavaná databáza z už dokončených projektov.
- Metóda z hora nadol vyžaduje menej funkčné a nefunkčné požiadavky a zaoberá sa celkovou charakteristikou systému. Tento odhad je zo začiatku pomerne abstraktný a konkretizuje sa každým krokom. Môžeme však podceňovať náklady ťažších technických komponentov z nižších úrovní. Prístup však berie do úvahy náklady na integráciu, konfiguráciu a dokumentáciu.
- Metóda z dola nahor spočíva v tom, že sa spraví odhad pre každý samostatný komponent a spájajú sa všetky komponenty tak, aby sa získal celkový odhad projektu. Tento prístup môže byť veľmi presný v prípade, ak systém bol navrhnutý detailne. Avšak metóda môže podhodnocovať náklady na činnosti na tej úrovni systému, akou je integrácia a dokumentácia.

Z vymenovaných metód, podľa môjho názoru, mala metóda z dol nahor vždy najväčší potenciál, pretože je aplikovateľná na hocijaký softvérový projekt, ale aj tak má svoje nedostatky. Tieto metódy sa používali pri vodopádovom vývoji softvéru, ale agilný vývoj priniesol pre odhadovanie projektu niečo úplne revolučné. Na začiatku sa spraví jeden odhad s tým, že sa navrhnú základné funkcie produktu a tieto funkcie sa rozdelia na šprinty. Podľa nich sa spraví určitý hrubý odhad. Krása agilného vývoja spočíva v tom, že sa jedná len o hrubý odhad a neskôr bude vývojový tím určovať odhady pre jednotlivé komponenty. Práve preto sú agilné metódy také výnimočné a dobré, lebo konečne rozhodujú o odhadoch jednotlivých komponentov tí ľudia, ktorí ich aj skutočne implementujú. Takto menej často dochádza k tomu, že sa plán nedodrží kvôli zlým odhadom.

Ako vieme správne odhadnúť náročnosť úloh?

Ďalšou dôležitou zložkou pre úspešný projekt sú techniky pre odhadovanie času a nákladov na projekt. Schopnosť presne odhadnúť čas alebo náklady na projekt a dopracovať sa k úspešnému záveru, je podľa mňa jednou z najťažších úloh pre softvérových inžinierov. Túto problematiku identifikoval aj Wayne Turek v práci [2], ktorá sa zaoberá siedmimi hriechmi projektového manažmentu. Podľa Tureka by mal hlavne tím určovať termíny pre odovzdanie komponentov počas vývoja, lebo je rovnako zlé, keď sa nedodrží termín, ako keď sa prečerpajú peňažné zdroje. Pri väčšine projektov sú termíny

určené vrcholovými manažérmi. Podľa môjho názoru má Turek pravdu v tom, že vývojový tím by sa mal zúčastniť určovania odhadov a termínov. Tento tím je vždy lepšie schopný odhadnúť reálny termín, lebo má skúsenosti s danými technológiami a nakoniec je to práve vývojový tím, na ktorom leží úspešnosť projektu.

Existuje veľa techník, pomocou ktorých sa dajú realisticky vytvárať odhady. Tieto techniky sa dajú kategorizovať do dvoch veľkých skupín [4]. V prvej skupine sú techniky, ktoré majú parametrický prístup a druhú skupinu tvoria techniky, ktoré majú heuristický prístup. Pri heuristických prístupoch sa nepoužívajú odhady založené na modeloch. Dokonca existuje veľa techník, ktoré spadajú do oboch skupín. Tieto odhady boli vypočítané len z obyčajných vzorcov, práve preto sa aj ťažko spĺňali. Všetky tieto techniky sú z čias, keď ešte vodopádový model bol jedinou možnosťou vo vývoji softvéru. Tieto časy sa však skončili a agilný vývoj priniesol do odhadovania úloh niečo úplne nové.

V agilnom vývoji sa používajú dve hlavné techniky odhadovania úloh [4], prvá je príbeh používateľov (angl. user story) a druhou sú pokrové karty. Tieto techniky umožnia vývojárom, aby sami rozhodovali o náročnosti a vyhradenom čase pre jednotlivé úlohy.

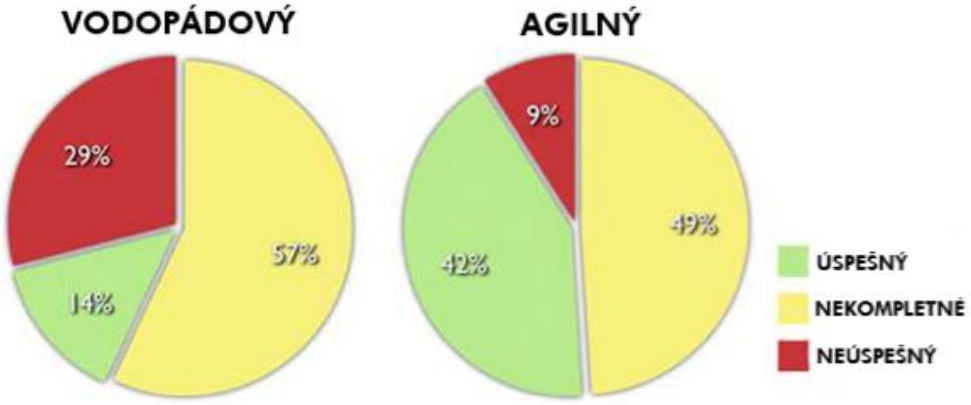
Techniky odhadovania pri agilnom vývoji [5]:

- Pri príbehu používateľov (angl. user story) sa navrhnú príbehy, ktoré určujú, ako používateľ interaguje so softvérom. Následne sa ohodnotí biznis hodnota a náročnosť tohto komponentu, ktorú samozrejme určujú vývojári. Podľa týchto údajov sa následne určí priorita a časový odhad.
- Pokrové karty si môžeme predstaviť tak, že pri odhadovaní náročnosti jednotlivých komponentov si každý z vývojového tímu vyberie tú kartu, ktorá podľa neho reprezentuje náročnosť daného komponentu. Potom sa všetky karty ukážu naraz, prípadne sa predebatujú veľké odchýlky v názoroch.

Tieto techniky zaručujú to, že odhad ktorý dostane vývojový tím, nie je nesplniteľný, pretože si ho navrhol tím sám. V tomto spočíva krása agilného vývoja.

Záver

Na záver môžeme povedať, že každý trend má svoje obdobia. Vodopádový model v softvérovom inžinierstve dominoval dlho, ale zmeny sú súčasťou života. Podľa môjho názoru zmena, ktorá nastala v oblasti vývoja softvéru, bola zmenou k lepšiemu. Agilný vývoj preukázal, že je dobre aplikovateľný pre malé i veľké firemné prostredia. Úspešnosť metódy len potvrdzuje kruhový diagram (pozri. Obr. 3) , ktorý svedčí o efektívnosti metódy.



Obr. 3 Úspešnosť jednotlivých vývojových metód.

V mojej eseji som sa snažil poukázať na to, ako agilné metódy zmenili trend nielen v softvérovom vývoji ale i v manažmente plánovania. Moje odporúčanie znie: nebojte sa používať nové prístupy v plánovaní, pretože všetko potrebuje raz za čas zmenu.

Použitá literatúra

1. Mehwish Nasir. 2006. A Survey of Software Estimation Techniques and Project Planning Practices. In Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD-SAWN '06). IEEE Computer Society, Washington, DC, USA, 305-310.
2. Turk, W. (2008). Seven Deadly Sins of Project Management.
3. Cohn, M. (n.d.). Estimating and Planning Agile Projects.
4. Smits, H. (2006). 5 Levels of Agile Planning: From Enterprise Product Vision to Team Stand-up, 1–11.
5. Pmp, L. B. (2005). Plans are Useless , but Planning is Indispensable The Basic Timeline, 1–8.

Annotation

How agile development replaced the old dinosaur

In software projects and also like in projects from other fields, planning playes a huge role. Correct estimates contribute to the succes of projects. In my esseay I have focused on the comparision of the traditional methods of estimating processes in planning which are usually used in the waterfall modell and the methods which are used in agile development. I have analyzed the state in planning management with the focus of estimation techniques in the waterfall modell and agile methodologies. I have compared methods and techniques in estimation and planning. I have tried to point out how the situation have changed after agile methods came to the scene. Also in this essay

I am trying to highlight the benefits and the need that development team should be involved in the estimating and the planning process.