# VPLYV (NE)PLÁNOVANIA NA AGILNÝ VÝVOJ SOFTVÉRU

*Význam plánovania pre projekt je najlepšie vidieť na spokojnosti zákazníka s produktom.*

*Michal Fojtík*

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

***Abstrakt.*** *Cieľom plánovania softvérového projektu je odhadnúť potrebné množstvo práce a vytvoriť samotný časový plán, ktorým sa bude projekt riadiť. Najnovšie trendy v oblasti softvérového inžinierstva, najmä agilne metódy vývoja softvéru, však pokladajú za dôležitejšiu schopnosť prispôsobovať sa zmenám, a nie striktné dodržiavanie plánu. Prispôsobivosť však neznamená úplnú absenciu plánovania, ale len iný prístup k tvorbe plánov. Na rozdiel od klasických spôsobov vývoja sa vytvárajú len krátkodobé plány, ktoré sú vytvárané celým tímom spolu so zákazníkom. Po splnení plánu je vždy vytvorený nový plán pre ďalšiu etapu vývoja. Z dlhodobého hľadiska sú identifikované len najzákladnejšie ciele projektu, ktoré však nemajú charakter plánu, ale len požiadaviek, ktoré ma hotový softvér spĺňať. Takýto systém plánovania sa ukazuje ako výhodný najmä pri menších projektoch, pri väčších projektoch nastavajú problémy. Táto esej si kladie za cieľ poukázať na výhody agilného plánovania a na možné riziká takéhoto prístupu k plánovaniu softvérového projektu.*

***Kľúčové slová:*** *agilný vývoj, plánovanie, plán*

## Úvod

Je veľmi ťažko predstaviteľné, že by v dnešnej dobe akýkoľvek významnejší projekt, nie len softvérový, mohol byť uskutočnený bez dôkladného plánovania. Preto je plánovanie softvérových systémov základnou a v podstate nevyhnutnou súčasťou manažmentu

softvérových projektov. Ďalšími procesmi súvisiacimi a nadväzujúcimi na plánovanie sú kontrola dodržiavania plánov, monitorovanie vývoja a v prípade nutnosti zmena plánov.

Hlavným účelom plánovania je definovať ciele projektu a požadovanú funkcionalitu. Následne je potrebné odhadnúť rozsah projektu, tým pádom aj jeho zložitosť, časovú náročnosť a potrebné množstvo ľudí. Po odhadnutí sa z výsledkov vytvára samotný plán, resp. harmonogram projektu do budúcnosti, kde sa špecifikované úlohy prideľujú tímom prípadne jednotlivcom.

Najväčším problémom plánovania a odhadovania je samozrejme jeho neurčitosť a závislosť od meniacich sa podmienok v čase implementácie projektu. Nesprávny predpoklad vo fáze plánovania môže v prípade neskorého odhalenia výrazne predĺžiť čas potrebný na dokončenie projektu, tým pádom projekt predražiť a ohroziť jeho ziskovosť. Tento problém sa softvéroví manažéri snažia vyriešiť novými, agilnými prístupmi k plánovaniu projektov, najmä čiastočnou elimináciou dlhodobých harmonogramov. Dá sa však vyvinúť kvalitný produkt bez dlhodobého plánovania, len s minimálnymi odhadmi a predpokladmi?

## Agilné metódy

Agilné metódy vývoja softvéru sú jedným z moderných prístupom k plánovaniu, manažovaniu a samotnému vývoju a testovaniu softvérových projektov. Najznámejšími takýmito metódami sú Extreme Programming, SCRUM a Lean Programming. Ich počiatky siahajú do začiatku 90. rokov 20. storočia, kedy sa tento spôsob vývoja začal vyčleňovať z klasických metód tvorby softvéru. Začiatkom nového tisícročia nastalo jeho prudké rozšírenie spolu so zvýšeným záujmom zo strany softvérového inžinierstva. Hlavným cieľom zástancov agilného vývoja je urýchliť vývoj softvéru, zjednodušiť proces vytvárania softvérového produktu a prispôsobiť ho čo najviac požiadavkám zákazníka. Agilný vývoj je najčastejšie aplikovaný na menšie a jednoduchšie projekty s menším počtom tímov a krátkou predpokladanou dobou vývoja.

Základnými myšlienkami agilného vývoja sú podľa jeho manifestu [3]:

1. Jednotlivci a zmeny pred procesmi a nástrojmi
2. Funkčný softvér pred vyčerpávajúcou dokumentáciou
3. Spoluprácu so zákazníkom pred zmluvnými rokovaniami
4. Prispôsobovanie sa zmenám pred striktným dodržiavaním plánu

Ako najdôležitejší dôvod rozvoja agilných metód sa uvádza prudký nárast počtu kvalitných vývojových nástrojov, tým pádom uľahčenie a zrýchlenie tvorby aplikácí. Súčasťou týchto trendov je aj prudký nárast počtu používateľov internetových služieb a celkovo informačných technológí, s nimi však prichádza aj zvyšovanie konkurencie v softvérovom priemysle. Toto núti softvérových vývojárov a firmy k zvýšeniu produktivity a k zvyšovaniu významu plánovania, flexibility a úžitkovej hodnoty softvéru [1].

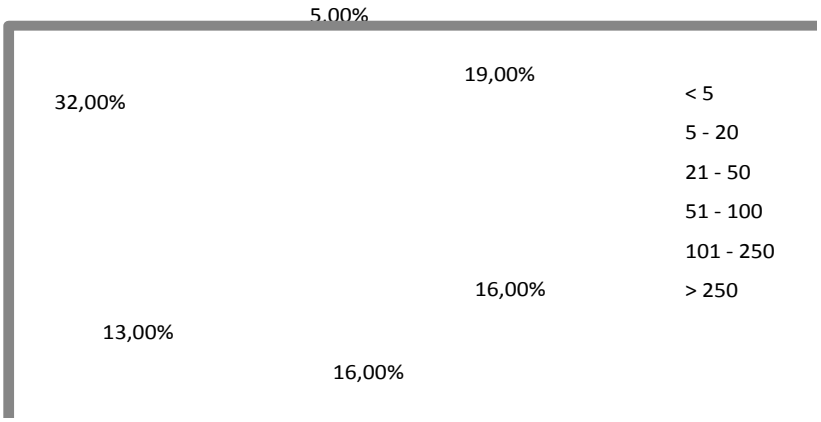
Najdôležitejším cieľom a mottom zástancom agilného vývoja je najmä spokojnosť zákazníka s výsledným produktom, resp. úspech produktu na trhu. Nie že by nebolo dôležité vytvoriť produkt s kvalitným kódom, technickou dokumentáciou a presným dodržiavaním vytýčeného procesu vývoja. Ak však bránia v realizácí primárneho cieľa,

musia sa iné priority odložiť nabok. Práve preto sa upúšťa od nemenných plánov a vytvárania dlhodobých a rozsiahlych plánov, ktorých časti v priebehu vývoja môžu stratiť na aktuálnosti. Zároveň sa od začiatku tvorby produktu počíta so zmenami, vďaka ktorým môže byť softvér kvalitnejší ako produkty konkurencie. Kvalita v kontexte agilného softvéru znamená čo najväčšiu spokojnosť zákazníka s produktom a teda meradlom dokončovania projektu je funkčnosť aplikácie. Preto sa dôraz kladie na úzku spoluprácu vývojárov so zadávateľmi, a časté odovzdávanie priebežného stavu produktu zákazníkovi. Ako hlavný spôsob komunikácie sa prezentuje osobný kontakt členov vývoja, čo si však vyžaduje kvalitné komunikačné schopnosti od všetkých členov. Taktiež sa od všetkých očakáva vysoká úroveň vedomostí, neustály prehľad o celkom projekte a motivácia dosiahnuť vytýčené ciele.

Prístup zástancov agilného vývoja softvéru zdôrazňuje viac individuálnu stránku vývoja každého softvéru, každý softvérový produkt je jedinečný a nedá sa dopredu presne definovať. Predchádzajúce prístupy k vývoju kládli dôraz na riadenie sa plánom vytvoreným v počiatočnej fáze projektu, čo spôsobovalo vysokú cenu v prípade zmien plánu počas vývoja. Na rozdiel od nich sa nové metódy sústreďujú skôr na samotný proces plánovania a zmeny plánu pokladajú za normálnu súčasť vývoja aplikácie. Preto nevytvárajú na začiatku jeden centrálny plán a analýzu pre celý proces vývoja, ale stanovujú sa najvýznamnejšie ciele vývoja. Tieto ciele sú stanovované spolu so zákazníkom, resp. odborníkmi a obsahujú len špecifikáciu čo má produkt robiť, nie ako má fungovať. Následne sa začína priamo vývoj produktu, ktorý prebieha neustále až po dokončenie projektu, paralelne s ostatnými procesmi ako analýza, testovanie a priebežná úprava plánov [1].

## Aplikácia agilných metód v praxi

Najvýznamnejším prieskumom zameraným na oblasť agilného softvéru je „State of Agile Development“ uskutočňovaný každý rok od roku 2006. Posledné výsledky sú z mesiacov jún - júl roku 2008, kedy sa prieskumu zúčastnilo 3061 respondentov z 80 krajín. Cieľom prieskumu bolo zmapovať situáciu v organizáciách, ktoré využívajú pri vývoji svojich projektov agilné metódy, nie zmerať celkový úspech agilných metód na trhu. Preto boli oslovení ľudia, ktorí reálne pracujú na projektoch aplikujúcich agilné metódy. Tieto prieskumy ukazujú, že agilný vývoj sa ďalej rozširuje na väčšie projekty. Podľa nich bolo v roku 2006 pracovalo 27% respondentov v organizáciách s viac ako 250 ľuďmi zainteresovanými v agilných projektoch, v roku 2007 to bolo 30% a v roku 2008 to už bolo 32% [6][7][8].




**Obr. 1**: Graf rozdelenia projektov podľa veľkosti organizácie [8].

Čo sa týka prínosov agility po jej zavedení do zabehnutých tímov, prieskum odhaľuje výrazné zlepšenia prehľadnosti a zvýšenie schopnosti prispôsobiť sa zmenám. Menšie zlepšenia pozorovali respondenti najmä čo sa týka zvýšenia produktivity, kvality produktu a morálky v tíme, zníženia potenciálneho rizika a urýchlenia dodania produktu zákazníkovi resp. na trh. Nedostatky alebo žiadne zlepšenie videli najmä v manažovaní rozložených tímov a nákladoch na vývoj [8].

**Tab. 1:** Tabuľka časti odpovedí na otázku „Akú hodnotu Vám priniesla implementácia agilného vývoja?“ [8].

| | Výrazné zlepšenie | Menšie zlepšenie | Žiadne zlepšenie |
|---|---|---|---|
| Prehľadnosť projektu | 41,50% | 41,80% | 15,00% |
| Schopnosť prispôsobiť sa zmenám | 50,50% | 42,10% | 6,60% |
| Zvýšenie produktivity | 23,60% | 50,50% | 15,60% |
| Zrýchlenie dodania produktu | 23,60% | 41,30% | 21,60% |
| Zvýšenie kvality produktu | 24,00% | 44,30% | 19,90% |
| Zníženie rizika | 16,60% | 48,00% | 23,60% |
| Zjednodušenie vývoja | 19,50% | 48,10% | 19,80% |
| Manažovanie rozložených tímov | 6,90% | 22,10% | 41,10% |
| Zníženie ceny | 7,60% | 30,40% | 39,50% |
| Zvýšenie morálky v tíme | 29,80% | 44,10% | 1,00% |

## Plánovanie využitím agilných metód

Najčastejších problémom pri plánovaní procesu vývoja softvéru je že nikto úplne presne nevie špecifikovať, čo má cieľový projekt spĺňať. Ešte väčším problémom býva neschopnosť ľudí oboznámiť ostatných s tým čo už o problémovej oblasti vedia. Moderné softvérové systémy sú tak zložité, že pri ich implementácí vzniká množstvo iných

problémov, ktoré sa ťažko dajú dopredu odhadnúť. Ak sú riešiteľmi projektu ľudia (čo platí pre každý projekt) musíme medzi ne zaradiť aj ľudské chyby, ktoré sú pri tvorbe projektu nevyhnuté a ťažko sa dopredu odhadujú. Tieto nútia ľudí meniť pôvodný plán a niekedy časť alebo celú architektúru. Ďalším problémom, ktorý sa môže vynoriť je zmena situácie na trhu, tým pádom zmena plánu zákazníka. Kvôli tomu v podstate nie je možné vytvoriť ideálnu architektúru systému a tým pádom aj stopercentne naplánovať projekt a držať sa jedného plánu. Zabrániť nám v tom môže aj ďalší faktor, ktorým je vzájomné prepájanie systémov, takže vývoj partnerského systému v čase vplýva na náš systém. Preto je myšlienka bezchybného, nemenného plánu, vytvoreného len z požiadaviek zákazníka nepredstaviteľná – takýto plán ešte nikdy nevznikol a nikdy nevznikne. Môžeme sa maximálne snažiť k nemu priblížiť [4].

Alebo môžeme ísť cestou agilného vývoja, ktorý s týmito obmedzeniami počíta. Ako už bolo spomenuté, najdôležitejšou časťou plánovania v agilnom vývoji nie je plán, ale samotný proces plánovania. Preto sú doňho zahrnutý všetci členovia agilného tímu. Predpokladom je, že najlepší odhad má človek, ktorý má skúsenosti, v minulosti už riešil podobné úlohy a dokáže si predstaviť kroky potrebné k úspešnému riešeniu úlohy. Takže plán nevytvára zopár manažérov pre celý tím, ale každý z členov má povinnosť vyjadriť sa, koľko času jemu resp. jeho tímu každá z daných úloh približne potrvá. Takýto prístup je výhodný aj z motivačného hľadiska, najmä v prípade neskúsených manažérov, ktorý si myslia že ak dajú vývojárom do plánu menej času, bude produkt hotový skôr. Toto však pôsobí skôr opačne a namiesto motivácie sú frustrovaní a prácu dokončia ešte neskôr. Naopak v prípade, že prácu v pohode stíhajú, zvykne ich morálka byť vyššia a s väčšou chuťou sa púšťajú do ďalšej práce.

Nemenej dôležitým princípom je neustále revidovanie plánu, tým pádom vlastne proces plánovania prebieha počas celého vývoja softvéru. Dopredu sa počíta so zmenami a chybami, v prípade ich výskytu sa plán modifikuje a pokračuje sa ďalej. Takže ak sa počas vývoja vyskytne potreba zmeny, či už z dôvodu zlej analýzy alebo z dôvodu nesprávnej realizácie, nevznikne neriešiteľný problém a značné škody pre zákazníka aj vývojára. Takáto spätná modifikácia plánu pomáha aj do budúcnosti pri plánovaní podobných projektov. Po ukončení projektu sa v pláne nachádzajú časové údaje vrátane opráv chýb a potrebných zmien, čo môže pri budúcich projektoch pomôcť odhadnúť dĺžku podobných úloh. Samozrejme všetko má svoje hranice, preto sa pri agilnom vývoji kladú vyššie nároky a zodpovednosť na programátorov. Najmä aby kládli dôraz na dôležitejšie funkcie a nenechávali si najzložitejšie veci nakoniec, aby bolo možné čo najskôr odhaliť možné komplikácie. Rovnako dôležitá je aj ich schopnosť komunikovať, oboznamovať ostatných s priebežným stavom a problémami, ktoré nastali alebo ich predpokladajú do budúcnosti [4].

Ďalším prínosom agilných metód do oblasti plánovania softvérových produktov je prístup k tvorbe úloh, ktorého účelom je odhadnúť množstvo potrebného času a prostriedkov čo najpresnejšie. Preto sa na počiatku tvorby produktu spolu zo zákazníkom definujú len hlavné ciele a funkcionalita produktu. Následne sa k požiadavkám vytvoria prislúchajúce úlohy. Lenže zložitým úlohám, ktoré si môžu vyžadovať niekoľko týždňov práce aby spĺňali istú funkcionalitu, je náročné odhadnúť dĺžku ich riešenia . Agilné metódy však idú ešte ďalej, rozdeľujú tieto úlohy na ďalšie menšie úlohy tak, aby bola čo najviac špecifická a primerane krátka. Ideálna dĺžka by

nemala prekračovať niekoľko desiatok hodín. Ku každej čiastkovej úlohe je okrem časového intervalu priradený aj človek zodpovedný za jej dokončenie. Takto sa dá dosiahnuť presnejší plán, ktorý je síce zložitejší, ale na druhej strane prispieva k vyššej produktivite a k rozloženiu zodpovednosti. Zároveň v rámci dodatočných úprav plánu sa plán dopĺňa o počet reálne odpracovaných hodín a upravuje sa počet hodín do dokončenia čiastkovej úlohy. Tým pádom sa dá odhadnúť dĺžka riešenia možných budúcich úloh aj iných režijných činností potrebných k ich dokončeniu.

## Neplánovanie využitím agilných metód

Napriek vyzdvihnutým pozitívam ani agilné metódy nie sú univerzálnym riešením na všetky problémy. Naopak môžu v prípade nesprávneho pochopenia a aplikovania princípov skomplikovať situáciu. Najčastejším problémom agilného vývoja sú ľudia, keďže jeho hlavným oporným bodom je ich kvalifikácia a motivácia. Mnohokrát sa stáva, že členovia tímu, najmä programátori nevidia zmysel v plánovaní všeobecne a už vôbec nemajú chuť ísť do podrobností. Podobne je to aj so zaznamenávaním priebehu, každého problému a každej zmeny, nemotivovaní a neinformovaní členovia v plánoch tímu nevidia zmysel. Radšej problém vyriešia čo najskôr, bez toho aby ho zaznamenali. Tým pádom strácajú detailné plány časť svojho zmyslu, najmä čo sa týka ich užitočnosti pri plánovaní budúcich úloh. Príčinou bývajú najčastejšie nedostatočné manažérske schopnosti vývojárov. Vhodným riešením sa zdá byť zvyšovanie kvality a vzdelania ľudí v tíme a poskytovanie lepších podporných prostriedkov na uľahčenie manažérskych úloh [2].

Argumentmi odporcov agilného plánovania sú nedostatočná dokumentácia a poriadok v projekte, absencia podrobnejšieho návrhu. Rovnako nepríjemné môže byť aj špecifikovanie abstraktných požiadaviek, ako sú rýchlosť, bezpečnosť alebo užívateľská prívetivosť. Tieto sa ťažko transformujú na krátke úlohy s časovými intervalmi, nemenej ťažko sa overuje ich plnenie a úroveň dokončenia. Tieto efekty môžu viesť resp. zapríčiniť iné problémy ako sú zložité určovanie koncovej ceny, náročné dohody s klientmi. Niektorí zákazníci môžu mať problém aj s tým, že musia určiť niekoho zo svojich ľudí, aby sa stretával s vývojármi na pravidelných stretnutiach. Ďalšími nepriaznivými dôsledkami môžu byť neočakávané zväčšenie rozsahu projektu alebo problémy v prípade nutnosti nahradiť niektorého člena tímu. Dajú sa však minimalizovať ak sa na ne pri plánovaní dopredu myslí a agilný tím sa im snaží aktívne predchádzať. Vôbec nevyplývajú z podstaty agilného vývoja, ale sú skôr všeobecné a môžu sa vyskytovať rovnako v prípade použitia iných metód plánovania. Reálnejším problémom je skôr náročný prechod na takýto spôsob vývoja pre ľudí, ktorí sú zvyknutí na tradičné plánovanie. Najmä ak sa týka ľudí s vynikajúcimi technickými zručnosťami, ktorým prechod istý čas potrvá a počas neho zoslabí ich výkonnosť.

Preto je agilný prístup najčastejšie využívaný v menších tímoch s pár desiatkami ľudí. Najčastejšie kvôli jednoduchšej komunikácí a preto, že menšie tímy väčšinou riešia menšie projekty, takže je pre vývojárov ľahšie mať celkový prehľad nutný k úspešnej aplikácí agilných metód. Pri niektorých veľkých projektoch vzniká špecifický problém s rôznymi štandardami, obmedzeniami a reguláciami kvality, ktoré si vyžadujú disciplinovaný vývoj s dôkladnou dokumentáciou. Takýmito citlivými projektmi sú napríklad vládne, armádne,

vesmírne a podobné projekty. V ich prípade potreba dôslednej kontroly procesu vývoja a testovania zabraňuje aplikácí agilného prístupu.

Existujú snahy použiť agilné metódy na rozsiahlych a náročných projektoch, často aj úspešné. Niektoré zo základných pravidiel agility sa však pri nich ukazujú ako obmedzujúce. Plánovať dopredu len jednu iteráciu, resp. šprint v každom tíme v projektoch s množstvom tímov určite nie je ideálne. Najmä ak počet iterácí realizovaných v každom tíme dosiahne niekoľko desiatok, hrozí strata prehľadu o projekte. Nehovoriac o tom že jednotlivý členovia tímov nemôžu mať prehľad o činnosti všetkých ostatných tímov. Tým pádom sa vývojárom sťažuje úloha plánovania ďalších šprintov, prípadne sa stáva až neuskutočniteľnou. Klasické metódy v plánovaní riešia tieto problémy tak, že na začiatku procesu odhadnú a naplánujú celý projekt. Toto je však nevyhovujúce, keďže vôbec nemusí byť isté že navrhnutým plánom sa docieli produkt, ktorý si zákazník objednal. Jedným z agilných prístupov je vytvorenie plánov na rôznych úrovniach, od zachovania tých viac agilnejších, ako denné príspevky alebo jednotlivé šprinty cez väčšie produktové plány zahŕňajúce viacero tímov až po hlavnú víziu produktu [5].

## Záver

V tejto eseji som sa pokúsil opísať možnosti aplikovania agilného vývoja v oblasti plánovania projektov. V prvej časti práce som opísal agilné metódy a ich hlavné rysy čo sa týka celého procesu vývoja softvérového projektu. V hlavnej časti som sa pokúsil zhrnúť základné princípy agilného plánovania, riziká ich nedodržiavania a výhody ich dodržiavania na úspešné dokončenie projektu. Napriek tomu, že v manifeste agilného softvéru je napísane, že dodržiavanie plánu je druhoradé, neznamená to absenciu plánovania, ako sa mnoho ľudí mylne domnieva. Dôležité je zmeniť pohľad ľudí na proces vytvárania plánov ako na prostriedok k dosiahnutiu cieľa, nie ako nemennú vec zadanú na začiatku vývoja.

## Použitá literatúra

1. Conboy K., Fitzgerald B.: Toward a Conceptual Framework of Agile Methods: A Study of Agility in Different Disciplines. s. 1-4.
2. Hochmüller E., Mittermeir R. T.: Agile Process Myths. ACM Press.
3. Kolektív autorov: Agile Manifesto. Dostupné na internete: http://www.agilemanifesto.org/, [cit: 2009-Október]
4. Murauskaite A., Adomauskas V.: Bottlenecks in Agile Software Development Identified Using Theory of Constraints (TOC) Principles. Dostupné na internete: http://gupea.ub.gu.se/dspace/bitstream/2077/10457/1/gupea_2077_10457_1.pdf, [cit: 2009-Október]
5. Smits, H.: The Impact of Scaling on Planning Activities in an Agile Software Development Context. In: *Proceedings of the 40th Hawaii International Conference on System Sciences – 2007*, ACM Press, s. 2-5.

6. VersionOne: Survey: "The State of Agile Development". Dostupné na internete: http://www.versionone.net/pdf/StateofAgileDevelopmentSurvey.pdf, [cit: 2009-Október]
7. VersionOne: Survey: "The State of Agile Development". Dostupné na internete: http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf, [cit: 2009-Október]
8. VersionOne: 3rd Annual Survey: 2008 "The State of Agile Development". Dostupné na internete: http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf, [cit: 2009-Október]

## Annotation

*Impact of (not) planning on agile software development*

*The purpose of software project planning is to measure the amount of work and create a project implementation schedule. Latest trends in software engineering, in particular agile methods of software development, consider the ability to adapt to changes to be more important than strict plan adherence. However, adaptability does not mean lack of planning. It is only different approach to planning. Unlike classical methods of development, only short plans are being made by the whole team along with customer representative. After finishing the plan, new one for next phase of development is made. In the long-term, only basic purposes of project are indentified, however. They do not have the characteristic of a plan, but only of requirements software should fulfil once finished. This system of planning is useful mostly in smaller projects, since in bigger projects, problems might arise. The purpose of this essay is to point out advantages and possible risks of this approach to software project planning.*