Manažment softvérového projektu a organizácia softvérových tímov

KRISTIÁN SZOBI

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava
szobi@chello.sk

Abstrakt. Pri vývoji softvéru hrá dôležitú úlohu zloženie a spôsob organizácie tímov. Táto esej sa zaoberá pohľadom na organizáciu softvérových tímov zo sociálneho pohľadu. V eseji približujem základné typy sociálnych štruktúr, popisujem ich prístup k softvérového procesu i projektu a výhody i nevýhody jednotlivých typov. Na záver spomínam typ používaný v praxi, napríklad firmou Microsoft, v ktorej som pracoval.

Úvod

Organizácia softvérových tímov tvorí jednu z hlavných častí pri manažovaní softérového projektu. Štruktúra organizácie závisí od viacerých faktorov. Spomeniem typ vyvýjaného softvéru, jeho zložitosť a v neposlednom rade aj zloženie samotných tímov.

Pri vývoji softvéru je veľmi dôležitý aj sociálny aspekt [6]. Rozoznávajú sa tri základné sociálne štruktúry. Pre každý z nich zhrniem a porovnám vlastnosti z pohľadu konceptualizácie úloh, metód a nástrojov pri vývoji softvéru. Uvediem aj stručný popis hybridných modelov, ktoré sa používajú napríklad v spoločnosti Microsoft alebo v iných spoločnostiach predávajúcich softvér.

Tri typy štruktúr vo vývojových tímoch

Typ	Definícia
Sekvenčný	Softvérový vývoj je produkčná snaha založená na lineárnej množine diskrétnych úloh. Ľudia pracujú v špeciálnych funkciách s formalizovanou interakciou na báze funkcií. Ľudia sú hodnotení za ich špecializované schopnosti
Skupinový	Na softvérový vývoj sa pozerá ako na kombináciu vývoja a produkcie, kde množina diskrétnych úloh je opakovaná až pokiaľ produkt nie je hotový. Vývojári sú organizovaní do vzájomne závislých skupín a sú hodnotení za ich špeciálne schopnosti a za schopnosť pracovať

Manažment v softvérovom inžinierstve, december 2005, s. 1-6.

	s druhými.
Sieťový	Na softvérový vývoj sa pozerá ako na proces stáleho vývoja so špecifickým zameraním na výsledok/produkt. Úlohy nie sú sekvenčné a sú zviazané s jednotlivcami (alebo malými skupinami), ktorých účasť je založená na interakcii. Členovia skupiny sú hodnotení za to, čo dokážu vyprodukovať. Tento prístup implikuje zložitú sieť väzieb medzi ľuďmi a centralizovaným manažmentom.

Tab. 1 Popis základných sociálnych typov

Existujú tri základné typy štruktúr vo vývojových softvérových tímoch: sekvenčný, skupinový a sieťový (ich bližší popis je v tabuľke č.1). Tieto štruktúry prezentujú idealizované pohľady [6]. Použijem ich, aby sme lepšie pochopili problémy hybridných modelov softvérového vývoja – modelov, kde sú skombinované vlastnosti viacerých typov.

Sekvenčný typ

Sociálna štruktúra zapuzdrená v sekvenčnom type je priama a predpokladá známe a predšpecifikované požiadavky na vyžadované úlohy. Sociálnu štruktúru môžeme vidieť ako nezávislú od väčšieho sociálneho a organizačného kontextu. Aj keď je táto štruktúra hierarchická, potreba diskusie je malá. Vrámci projektu má každá osoba úlohu, ktorá je diskrétna a špecializovaná [6]. Naviac, predšpecifikované požiadavky na úlohy ďalej implikujú predpísaný pohľad na produkčný proces. To znamená, že sekvenčný typ je podporovaný tvrdením, že dobrý proces vedie aj k dobrému produktu.

Základom sociálnych interakcií v sekvenčnom type je koncept riadenia. Z toho vyplýva, že interakcie medzi ľuďmi sú sledované len z pohľadu funkcie práce, ktorú vykonávajú. Tento pohľad umožňuje výmenu ktoréhokoľvek človeka iným s tou istou úrovňou schopností.

Dôraz v práci zahrňuje potrebné informácie na pracovný produkt a/alebo dokumenty na splnenie úlohy. Môžeme teda povedať, že potrebná komunikácia vrámci tímu i komunikácia medzi tímami môže byť definovaná, formalizovaná a pravdepodobne aj automatizovaná [6]. Sekvenčný typ by teda profitoval z automatizovaných činností. Orientácia riadenia, formalizované interakcie medzi členmi tímu a zameranie na automatizované činnosti znamenajú, že nie je takmer vôbec potrebné, aby vznikli silné putá medzi členmi tímu [6]. Príkladom sekvenčného typu je napríklad tradičný vodopádový model.

Na základe úrovne špecializácie a oddelenia úloh, práca pri sekvenčnom type sa javí ako rutina. To môže spôsobiť, že členovia tímu len veľmi ťažko vidia hodnotu ich individuálnej práce vrámci celku. Oddelenie úloh môže tiež viesť k ohraničenej interakcii s inými členmi tímu a zmenšiť pravdepodobnosť, že sa vytvorí súdržnosť softvérového tímu. Typické prístupy na zjemnenie týchto prípadov sú „cross-training“ a väčšie zasahovanie manažmentu napríklad uskutočňovaním formálnych stretnutí [5].

V sekvenčnom type spôsobujú orientácia na proces a špecializácia úloh, že individuálny člen môže ľahko ignorovať prácu týkajúcu sa viacerých úloh (ako napríklad dokumentácia). Je to spôsobené tým, že slabý výkon sa ukáže v inej časti procesu a nie v tej, v ktorej bol zavedený. Napríklad zlé požiadavky neovplyvnia priamo analytika a ani zlý kód priamo neovplyvní programátora.

Existujú viaceré techniky na zmiernenie tohto nedostatku - napríklad revízie a kontrolné zoznamy (check lists). Použitie týchto techník má často dva dôsledky:

- Priestor pre zásah manažmentu
- Prerozdelenie členov tímu aby boli súčasťou aj inej úlohy

Pretože je sekvenčný typ založený na rutine, cieľom je často automatizácia. Tento trend môžeme vidieť vo vývoji integrovaných vývojových prostredí a CASE systémov [6].

Skupinový typ

Sociálna štruktúra v skupinovom type odráža interaktívnu snahu. V tomto type sú procesy vývoja softvéru založené na množine preddefinovaných úloh avšak do úvahy sa berú aj individuálne schopnosti a slabé stránky členov tímu. Na úlohy, ktoré sú často sekvenčné, sa môžeme pozerať iteratívne a explicitná pozornosť je zameraná na zlepšovanie procesu členmi skupiny [6].

Skupinový typ v jeho iteratívnej povahe taktiež explicitne rozlišuje, že softvérový vývoj a produkcia sú často úzko spojené [6]. Preto je skupinový typ normatívny. Naviac, skupinový typ navrhuje voľnú hranicu medzi tímom a sociálnym kontextom a tvrdí, že prelínanie hraníc je dôležitou časťou vývoja. Hlavná myšlienka však stále odzrkadľuje prístup, v ktorom najdôležitejší je proces rovnako ako v sekvenčnom type.

Skupinový typ softvérového vývoja vytvára explicitnú požiadavku na sociálnu interakciu medzi členmi skupiny [6]. Sociálne štruktúry sú orientované na riešenie nevyhnutných konfliktov, ktoré vznikajú pri spolupráci. To znamená, že tu existuje potenciál pre čiastočné automatizovanie produkčných úloh. Taktiež sú potrebné aj dodatočné nástroje a metódy, ktoré priamo podporujú a/alebo umožňujú spoluprácu medzi členmi tímu. Príkladom tohto typu môže byť špirálový prístup.

Skupinový typ vníma dôležitosť interakcie medzi členmi tímu a snaží sa medzi nimi dosiahnuť zhodu. Skombinovaním s cyklickou a integrovanou povahou produkčných úloh sa navrhuje, aby členovia tímu boli vyškolení a podporovaní v procese učenia sa skupiny a manažmente konfliktov [5]. To znamená, že v skupinovom type je úlohou zlepšiť schopnosti člena tímu v tímovej spolupráci.

Z pohľadu skupiny je iteratívna povaha projektu veľmi dôležitá [6]. Ako však tímy vedia kedy prestať iterovať? Môžem s istotou povedať, že najznámejšou formou kontroly iterácie sú peniaze alebo nejaké iné ohraničenie zdrojov. Avšak iná možnosť sa javí napríklad zavedením medzi-iteračných úloh ako sledovanie zmien a verzií, kontrola „vydania“ a plánovanie „vydania“ [5].

Skupinový typ by ocenil nástroje podporujúce spoluprácu v tíme. Skupinový typ taktiež využíva CASE nástroje ale pohľad na ich použitie je iný ako v sekvenčnom type – sústreďuje sa na podporu spolupráce [7].

Sieťový typ

Sociálna štruktúra v sieťovom type predpokladá kolekciu vzťahov medzi členmi siete. Tieto vzťahy odrážajú frekvenciu a hodnotu zo vzájomných interakcií. Silnejšie väzby vznikajú pri spoločných záujmoch, značnom zdielaní informácií a interakcií na vyšších stupňoch. Slabšie väzby znázorňujú menej časté interakcie a rôzne formy toku informácií. V sieťovom type sa v procese vývoja zvyčajne objavujú a premietajú sieťové väzby, ktoré sa vyvinuli medzi účastníkmi [6]. Na rozdiel od prvých dvoch typov, prioritu má produkt a samotný proces je až druhoradý. V sieťovom type individuálni členovia a sociálne vzťahy medzi nimi definujú snahu softvérového vývoja. Navyše, táto sieť je plne zahrnutá v širšom sociálnom kontexte, ktorý nemusí byť ohraničený organizačnou alebo geografickou hranicou [6].

Vznikajúci proces je ohraničený úlohami, ale tieto úlohy sú funkciou sociálnych vzťahov, ktoré tvoria sociálnu sieť. Takže, aj keď nie je tento proces centrálny, existuje nejaká forma kontroly verzií, testovania a dokumentácie. Sieťový typ vychádza z tvrdenia, že dôležitým faktorom pri tvorbe dobrého produktu je mať dobrých ľudí [6]. Z tohto prístupu, kde sa dôraz kladie na samotných ľudí, vychádza aj tvrdenie, že je veľmi náročné, možno až nemožné, vymieňať kľúčových členov (člena) siete, ku ktorým vedie veľa spojení vrámci siete. Títo ľudia tvoria spojenia, ktoré vytvárajú samotnú sieť.

Na podporu siete je kritické aby nástroje na vývoj softvéru poskytovali medziprepojenia [5]. Môžeme jednoducho povedať, že nástroj na vývoj softvéru je cenený na základe toho ako pomáha individuálnemu členovi a/alebo ako dobre umožňuje zdielanie v sieti. Druhým dôsledkom je to, že sieťový typ, ktorý je zároveň vznikajúci aj popisný, znázorňuje vývojový pohľad na softvér. Centralizácia sociálnych štruktúr a interakcia individuálnych členov v sieťovom modeli často spôsobujú, že snaha skupiny je často sporná [5]. Z tejto perspektívy sú interakcie medzi členmi zamerané na vlastnosti produktu, funkcie a akcie. Exituje zopár procedurálnych detailov na riešenie sporov, avšak očakávania medzi členmi sú vyjadrené ako „ukáž a povedz“. To znamená, že riešenie konfliktov je často založené na poskytnutí zdrojových kódov k podnetu, ktorý sa má vyriešiť.

Vývoju softvéru pomocou modelu „tím hlavného programátora“ je jeden z príkladov. Silné väzby sú medzi členmi a hlavným programátorom. Naopak, slabé väzby sú medzi samotnými členmi tímu. Nárast snahy vývoja otvoreného softvéru odráža druhú formu sieťového typu. Sieť má v tomto prípade viacero hlavných ťažísk v závislosti od dôležitosti a viacnásobné väzby medzi mnohými členmi tímu.

V sieťovom type je často náročné oceniť prínos bez zatajenia ďalších spojení. Kvôli vzájomnej závislosti na práci skombinovanou s individualizovanou podstatou spôsobu vykonania práce, vyhodnocovanie prínosu je založený na výsledkoch. Tento spôsob je použitý aj vo firme Microsoft. Prístup založený na výsledkoch si vyžaduje silný projektový manažment, ktorý je často centralizovaný do jedného alebo viac uzlov siete. Takže sieťový typ rozptyľujúci prácu vlastne koncentruje manažment. Môžeme

teda povedať, že aj keď sa projekt riadi jedným človekom (alebo uzlom), tento sa môže meniť ale málokedy sa riadenie zdieľa [4].

Sieťový typ začína s množinou individuálnych ľudí navzájom spojených vyvíjajúcimi sa sociálnymi väzbami. V tomto prostredí môže nastať problém pri opakovateľnosti a stabilite – preto je dôležitá snaha o zaistenie spoločného procesu pre opakované činnosti ako prideľovanie úloh, hlásenie pokroku a pod. Jednou metódou zlepšenie sieťového typu môže byť verejná projektová sledovacia tabuľa [3]. Táto metóda by pomohla udržať a sledovať vzájomné vzťahy medzi členmi siete.

Nástroje pre použitie v sieťovom type musia na základe jeho povahy podporovať znovupoužiteľnosť a interakciu. To znamená z pohľadu siete najmä jednoduché zdielanie súborov.

Hybridné modely

V praxi je viac pravdepodobnejší scenár, že tím si adoptuje hybridnú sociálnu štruktúru, ktorá obsahuje prvky viacerých základných typov. Napríklad, Brooks [2] a Baker [1] píšu o snahe firmy IBM pri vývoji systému System 360. Brooks podčiarkuje dôležitosť štruktúry projektového vývoja, zatiaľ čo Baker vysvetľuje fungovanie tímu hlavného programátora. Sekvenčný typ podporovaný Brooksom poskytuje štylizovaný pohľad na vývoj kým Bakerov popis poskytuje náhľad ako hlavný programátor môže vytvoriť hybridnú sociálnu štruktúru.

Jeden úsudok, ktorý z tohto vyplýva je: aby fungoval vývoj softvéru, hybridná sieťová štruktúra vzťahov je potrebná medzi vývojármi [4]. Na základe existujúcich dôkazov si myslím, že je to pravda. Napríklad, Weinberg zaznamenal ako produktivita vývojárov klesla potom, čo boli odstránené automaty na nápoje a sladkosti z oddychovej miestnosti programátorov, pretože následne strávili menej času vzájomnou neformálnou komunikáciou. Odstránením automatov zbavili vývojárov ich závislosti na neformálnej sieti, ktorá vznikala v okolí týchto automatov.

Zaujímavé je porovnanie s vývojom baleného softvéru. Prístupy pri vývoji baleného softvéru sa líšia od tradičného softvérového vývoja. Sám som pracoval vo firme Microsoft. Interakcia medzi vývojármi je väčšia a menej sa spolieha na formálny proces. Naopak, väčší dôraz sa kladie na konštrukciu produktu. Tento dôraz na produkt umožňuje interné vzájomné súperenie medzi jednotlivými vývojármi. V základe, prístup k vývoju vo firme Microsoft je hybridom medzi skupinovým a sieťovým typom.

Na základe tohto prehľadu sa dajú odvodiť ďalšie dva fakty:

- Pre každú snahu v softvérovom vývoji slúži malé množstvo kritických ľudí ako uzly vo vyvíjajúcej sa sieti programátorov
- Akonáhle sa stanú vývojové siete formálne štruktúrované, problémy konfliktov a závislostí sa stanú viac a viac dôležité

Záver

Sociálny pohľad na softvérový vývoj nám pomáha identifikovať tri základné typy sociálnej organizácie. Ich rozdiely nás vedú k otázke, ktorý prístup je ten najlepší? To je nepochybne empirická i filozofická otázka a ide za hranice tejto eseje. V eseji popisujem vlastnosti jednotlivých typov, možné komplikácie pri ich využití a ich odstránenie. Tak či onak, spomínam hybridný model, ktorý existuje a v ktorom sa miešajú prvky týchto základných typov. Cieľom bolo podať čitateľovi informácie zo sociálneho pohľadu pri výbere typu organizácie softvérových tímov.

Použitá literatúra

1. Baker, F. Chief Programmer Team Management of Production Programming. *IBM Systems Journal,* 11(1), 1972, 56-73.
2. Brooks, F., The Mythical Man-Month, *Datamation,* 1974. 44-52.
3. Raymond, E., *The Cathedral and the Bazaar*, Available online at http//www.tuxedo.org/~esr/writings/, 1999.
4. Sawyer, S. (2000) "Packaged Software: Implications of the Differences from Custom Approaches to Software Development, *European Journal of Information Systems, 9* (1), 2000, 47-58.
5. Sawyer, S., Farber, J. and Spillers, R.. Supporting the social processes of software development teams. *Information Technology & People, 10*(1), 1997, 46-62.
6. Sawyer Software Developing Teams: T*hree Archtetypes and their Differences, 2000.*
7. Vessey, I., and Sravanapudi, P. CASE Tools as Collaborative Support Technologies. *Communications of the ACM, 38*(1), 1995, 83-95.

Annotation

Management of software project and the software team organization

An important role in the process of software development is the way of organizing software teams. This paper focuses on the software team organization from social perspective. I talk about basic types of social structures and describe their attitudes to both software process and software project, and the cons and pros of the particular type. At the end, I am mentioning a model used in real-world scenario by for example Microsoft corp. in which I used to work.