cesko.digital

Příručka řízení státních IT projektů

Robin Carnahan, Randy Hart & Waldo Jaquith
Přeložil Tomáš Znamenáček

Pokud tento text čtete v tištěné podobě a nemáte klikací odkazy, můžete si otevřít elektronickou verzi textu na bit.ly/prirucka-cd.

Obsah

Úvod

Pouze 13 % velkých státních softwarových projektů končí úspěchem.[1] Prostředí veřejné správy je pro vývoj softwaru obzvlášť složité, protože veřejná správa postrádá základní znalosti moderního vývoje softwaru a spoléhá se na zastaralé způsoby nákupu softwaru.

Státy jsou přitom na moderním softwaru a hardwaru stále víc závislé, nedokázaly by bez nich poskytovat základní veřejné služby a úspěch jakékoliv velké politické změny je podmíněn úspěchem její softwarové infrastruktury. Různé státní úřady čelí podobným výzvám – musí v rámci svých rozpočtových a personálních možností modernizovat zastaralé technologické systémy, které jsou drahé, neflexibilní a neefektivní.[2] Státní činitelé a úřady při tom ovšem často spoléhají na tytéž zastaralé procesy, které je do současné problémové situace dostaly.

Veřejnost si zaslouží vládu disponující toutéž špičkovou technologií, kterou lze získat v běžném komerčním prostředí. Závisí na tom důvěra veřejnosti ve stát.

Tato příručka je určena exekutivcům, rozpočtovým specialistům, zákonodárcům a dalším „netechnickým" lidem, kteří rozhodují o rozpočtech státních IT projektů a dohlíží na ně. Dozvíte se v ní, jaké otázky si musíte klást, jak identifikovat žádoucí výsledky, a naučíte se základy moderního vývoje softwaru, aby vaše projekty pokud možno končily úspěchem.

Kromě toho vám příručka poskytne nástroje pro řešení souvisejících problémů, například:

- Souběžný provoz, údržba a modernizace starších IT systémů
- Závislost na dodavateli plynoucí ze starších smluvních závazků
- Bariéry v komunikaci a sdílení informací uvnitř organizací, strach z rizika

- Dlouhé rozpočtovací cykly neodpovídající moderním metodám vývoje softwaru
- Bezpečnostní rizika
- Hledání vhodných dodavatelů

Text se věnuje především zakázkovému vývoji softwarových systémů, ale je důležité si uvědomit, že prvky šité na míru zákazníkovi často obsahuje i běžný komerční krabicový software (COTS, *commercial off-the-shelf*) nebo software poskytovaný jako služba (SaaS, *software as a service*). Jakmile si necháte na běžném komerčním softwaru něco upravit na míru, většina této příručky se vztahuje i na něj – viz též sekci *Pozor na past krabicového softwaru na míru* níže.

Veřejná správa musí být dobrým správcem veřejných peněz, což znamená, že musí pro své zaměstnance vyžadovat dobře použitelné, přiměřeně drahé a dlouhodobě udržitelné digitální nástroje. A přesně takové vám pomůže získat následující text.

Základní principy návrhu moderního softwaru

Technologické projekty mají větší šanci uspět, pokud „netechnické“ profese dohlížející na jejich průběh a rozpočet rozumí šesti základním principům moderního vývoje softwaru, kterými jsou: **design zaměřený na uživatele**, **agilní vývoj softwaru**, **DevOps**, **modulární architektura**, **modulární zakázky** a **vlastnictví produktu**. Jde o obecné principy, jejichž pochopení nevyžaduje technické vzdělání. Ale jakmile je jednou ovládnete, dostanete se skrz vatu a technický žargon na jádro úspěšného řízení softwarových projektů.

Design zaměřený na uživatele

Veškerý vývoj softwaru by se měl točit kolem potřeb skutečných koncových uživatelů, konkrétních lidí, kteří budou systém používat. Těmito koncovými uživateli mohou být například žadatelé o dávky, pracovníci call center, sociální pracovníci nebo kdokoliv jiný z veřejné správy a bezpočtu dalších společenských skupin.

Návrh softwaru pro uživatele a s uživateli snižuje rizika projektu, protože zvyšuje šance, že výsledný software bude řešit skutečné problémy – nikoliv to, co za problémy považuje omezená skupina účastníků projektu. K identifikaci oněch skutečných problémů existuje řada různých postupů, například rozhovory s uživateli nebo testování použitelnosti.

V projektu zaměřeném na uživatele slouží veškerá práce k naplnění potřeb koncových uživatelů. Práce se hledá a prioritizuje v těsné a pravidelné spolupráci s koncovými uživateli a přihlíží k technickým

omezením, kterým ovšem bezvýhradně nepodléhá. (Jinými slovy, cílem práce je získat nějakou novou hodnotu pro uživatele. Během tohoto procesu je nutné respektovat možnosti zvolených technologií, například programovacích jazyků nebo serverového softwaru, ale zdánlivá technická omezení by nikdy neměla vést k tomu, že se nějaká práce vynechá.) Technický tým i koncoví uživatelé během práce pravidelně kontrolují výstupy a systém se nepovažuje za hotový, dokud koncoví uživatelé neodsouhlasí, že splňuje všechny jejich potřeby.

Design zaměřený na uživatele stručně řečeno říká, že je potřeba řešit problémy konkrétních relevantních uživatelů, nikoliv problémy šéfů jejich šéfů.

Agilní vývoj softwaru

Ve veřejné správě odjakživa převládá praxe podrobného plánování velkých softwarových projektů nadlouho dopředu. Nejen vývojáři softwaru ovšem už před léty došli k tomu, že tyto plány nikdy neodpovídají realitě a vždy vyžadují řadu drahých úprav. Takže je načase, aby i zadavatelé přestali vyžadovat podrobné dlouhodobé plánování a začali softwarové projekty rozpočtovat jinak.

Metodě plánování celého projektu předem se říká *vodopád*. Kdybyste tímto stylem plánovali dejme tomu měsíční rodinný výlet po Evropě, pečlivě byste předem naplánovali každý den od podrobného itineráře cesty přes zamluvení hotelu až po předplacení všech jídel nebo zakoupení lístků na různé atrakce. Což by vůbec nefungovalo, protože byste nevyhnutelně narazili na změny, nečekané možnosti a podobně. Racionálně uvažující člověk nebude chtít rozhodnout všechno předem na začátku cesty, protože ještě neví, co ho cestou čeká. Většina lidí by si naplánovala hrubou trasu a klíčové zastávky – a podrobnosti by doplnili až po cestě.

Podobný, takzvaný *agilní* přístup lze zvolit ve vývoji a modernizaci softwarových systémů. Agilně vedené projekty nespoléhají na léta drahého plánování a sbírání požadavků, po kterém by teprve došlo na samotný vývoj softwaru. Agilní projekty se plánují pouze v hrubých obrysech a kladou důraz na co nejrychlejší start reálných prací, které pak postupně směřují k dobře popsanému celkovému cíli projektu. Na jeho naplnění obvykle pracuje malý, samostatný a motivovaný tým (obvykle 5–9 lidí včetně vývojářů, produktových manažerů, UX expertů, copywriterů nebo bezpečnostních expertů), který se soustředí na potřeby koncových uživatelů a v opakovaných dvoutýdenních cyklech dodává konkrétní, fungující software.

První den každého cyklu si tým naplánuje práci na následujících 14 dní. (Cyklus může být kratší, například týdenní, i delší, například až měsíční. Dva týdny jsou nejčastější.) Každý úkol, na kterém bude pracovat, má podobu takzvané „user story" neboli uživatelského příběhu, což je konkrétní uživatelská potřeba identifikovaná během práce s uživateli.[3] Seznamu všech user stories se říká *backlog*.

Na vybraných user stories tým 14 dní pracuje a na závěr cyklu zkontroluje výsledky, otestuje je na uživatelích a nabere si další user stories z backlogu na příští cyklus, načež se vše opakuje. Cyklům se obvykle říká *sprinty*.

Software dodaný v úvodních sprintech obvykle nestojí za mnoho a často je dokonce později úplně nahrazen jiným, ale důležité je, že pomalu a systematicky formuje technický přístup celého projektu a pomáhá jej smysluplně integrovat mezi stávající systémy úřadu.

Každý sprint bez výjimky končí dodáním fungujícího softwaru – plně otestovaného, plně dokumentovaného, připraveného k použití. Dochází tedy k průběžnému dodávání hodnoty až do okamžiku, kdy je systém připraven k širšímu nasazení. Tým pak pracuje dál, dokud se nevyčerpají všechny cíle nebo peníze, podle toho, co přijde první.[4]

Dodavatel je placen za čas svých zaměstnanců, nikoliv za softwarový systém. Veškeré výstupy – software, dokumentace, výzkum, grafické návrhy, *zkrátka všechno* – patří státu a jsou odevzdány na kon-

ci každého sprintu. Technologie se mění, politika se mění, předpisy se mění, zákony se mění, priority se mění. Žádný projekt naplánovaný pečlivě do detailu předem se s těmito změnami nebude schopen vyrovnat a je velká šance, že vůbec nedojde do konce – anebo jen za cenu výrazných vícenákladů, průtahů a drahých požadavků na změny.

Díky spojení agilního vývoje s designem zaměřeným na uživatele může vývojářský tým neustále iterovat směrem ke skutečným potřebám uživatelů, a to způsobem, na který by při plánování předem nikdy nepřišel.

V roce 2019 vydalo americké ministerstvo obrany studii Software Acquisition and Practices (SWAP), která obsahuje i stručný návod, jak odhalit rádoby agile. Najdete v něm praktický souhrn pravidel agilního vývoje a sadu otázek, kterými si i netechnický lídr týmu může ověřit, že skutečně dělá agile a nikoliv parodii na něj.

Vlastnictví produktu

Pokud chce veřejná správa získat zpět kontrolu nad svými projekty, musí se soustředit na výsledky, nikoliv na výstupy. To znamená, že musí upustit od některých tradičních manažerských praktik a orientovat se především na produkt.

Slovo „produkt“ může znít v kontextu veřejné správy cize, ale jde o důležitou součást terminologie technologického světa. „Produkt“ je zkrátka cokoliv, co vzniká: webová stránka, mobilní aplikace, intranetová aplikace, atd. A ačkoliv to slovo dává větší smysl v kontextu běžného byznysu, prakticky celé produktové uvažování se dá bezvadně uplatnit i ve veřejné správě.

Product owner neboli vlastník produktu je klíčovou osobou každého softwarového projektu a *musí* jít o státního zaměstnance. Vlastník produktu spolupracuje se všemi zainteresovanými stranami na směřování produktu, s důrazem na to, aby z něj mohli co nejrychleji těžit koncoví uživatelé. Vlastník produktu je součástí agilního procesu vývoje, ve

kterém průběžně definuje a prioritizuje práci produktového týmu, pomocí jasných výkonnostních metrik měří pokroky projektu a komunikuje se všemi zúčastněnými stranami.

Vlastník produktu nepotřebuje špičkové znalosti technologií. Měl by ale znát uživatele systému, jeho doménu (například systém zdravotního pojištění nebo fungování registru vozidel) a různé mantinely, ve kterých se systém musí pohybovat.

Silný vlastník produktu se stará o to, aby měl projekt jasnou vizi a strategii, aby měly projektové týmy prostor pro vlastní rozvoj a aby stavěly nebo nakupovaly to, co bude skutečně postupně přinášet hodnotu koncovým uživatelům. Nemilosrdně prioritizuje tak, aby produkt splňoval potřeby uživatelů a největší pozornost dostávaly potřeby, které jsou v dané chvíli nejdůležitější. Vlastník produktu potřebuje od svého úřadu dostatečný mandát, aby mohl zastupovat účastníky projektu a dělat operativní rozhodnutí bez komplikovaných rozhodovacích hierarchií.

Tato představa se liší od běžného projektového řízení státních projektů. Agilní vlastník produktu nemá Ganttův diagram ani podrobnou pětiletku. Naopak má ale jasnou představu, čím pomůže koncovým uživatelům a jak toho dosáhne. Jeho nejdůležitějším úkolem je pečlivě sledovat celý vývojářský tým a hlídat, aby se pohyboval na ideální hranici mezi potřebami státu a potřebami koncových uživatelů systému.

Vlastník produktu se může učit za pochodu, ale je lepší ho proškolit předem. Agilní vývoj nebo některé jeho konkrétní varianty (například Scrum) vyučuje řada kurzů, některé přímo určené pro vlastníky produktů. Škála je široká od videí na YouTube až po osobně vedené vícedenní kurzy s certifikací. Čím důležitější projekt, tím formálnější a důslednější by mělo být vzdělání vlastníka produktu.

DevOps

Historickým vývojem došlo k tomu, že o provoz hotového softwaru se často stará jiný tým než ten, který jej vytvořil. Dodavatel například stráví několik let vývojem softwaru a následně ho předá státnímu IT týmu (anebo jinému dodavateli), kterému občas zabere měsíce práce, než systém bezchybně rozběhne na svých serverech. Celý proces obvykle doprovází nemalá míra frustrace a vzájemného obviňování, výsledkem čehož může být i selhání celého projektu. Veřejná správa proto často trvá na tom, že dodavatel softwaru musí systém i provozovat, na vlastní infrastruktuře a neomezeně dlouho. Tím se výrazně omezuje soutěž mezi dodavateli (většina z nich se hostingem softwaru nezabývá) a dochází k závislosti na jednom dodavateli (*vendor lock-in*), ze které pak obvykle plynou vysoké provozní náklady. Spoléháním na tento zastaralý režim provozu získáte za větší peníze menší hodnotu, než jakou byste dostali, kdybyste zvolili moderní nástroje běžně používané v soukromém sektoru.

Řešením je DevOps, tedy spolupráce týmů zodpovědných za vývoj (***dev****elopment*) a provoz softwaru (***op****erations*). Cílem je automatizovat práci, která je potřeba k testování softwaru a jeho nasazení na produkční server, kde z něho mají užitek uživatelé. Vývojáři už pak nemohou prostě předat dokončenou práci provoznímu týmu s tím, že „jim to funguje". Součástí jejich práce je napsat automatické procesy pro správné nasazení softwaru na produkční server. Nesou zodpovědnost – jak praktickou, tak smluvní – za to, že jejich kód poběží na serveru správně.[5]

Je dost pravděpodobné, že tímto způsobem vznikla většina softwaru, který denně používáte na svém telefonu a počítači. DevOps automatizuje testování kvality, testování bezpečnosti, integraci práce od více vývojářů i nasazování softwaru na server. Pokud je součástí tohoto procesu i testování bezpečnosti, přidává se do zkratky občas i bezpečnost, *security*, a mluví se tedy o DevSecOps.

Modulární architektura

Velké a složité softwarové projekty mají tendenci se hroutit pod vahou vlastní administrativy. Žádný z jednotlivých vývojářů nerozumí celému systému, ale každý nový člen projektového týmu zvyšuje složitost komunikace uvnitř týmu. Roste potřeba dalších dodatečných rolí (například softwarových architektů), se kterými musí vývojáři konzultovat svou práci, a jednotliví vývojáři se musí navzájem pečlivě koordinovat, aby předešli konfliktům. S rostoucí velikostí týmu zkrátka roste režie na jeho řízení a klesá čas strávený samotnou prací. Pokud se chcete tomuto osudu vyhnout, je lepší velké projekty rozbít na několik menších, víceméně samostatných. V tomto modelu každá komponenta komunikuje s ostatními prostřednictvím jednoduchých modulárních standardů, takže libovolný kus skládačky lze kdykoliv vyměnit. Místo monolitu, na který bude za pár let každý nadávat, vznikne malý ekosystém, jehož jednotlivé části lze snáz aktualizovat a upravovat podle měnících se potřeb. Každou komponentu má na starosti samostatný agilní tým, který se mimo jiné stará o dokumentaci komunikačního rozhraní komponenty (API, *application programming interface*). Nároky na koordinaci mezi týmy pak nejsou tak velké – z větší části jim stačí sledovat dokumentaci API ostatních komponent, se kterými spolupracují.

Pokud jednotlivé komponenty používají dostatečně obecné, abstraktní API (tedy jakýsi společný standard), mluvíme o architektuře orientované na služby neboli *service-oriented architecture*, SOA. Jde vlastně o analogii konceptu standardizovaných součástek, který umožnil průmyslovou revoluci. Standardizovaná rozhraní jsou základní ideou cloud computingu, elektrických zásuvek, USB konektorů, Lega, vlaků a nespočtu dalších moderních produktů a procesů.

Modulární návrh IT systémů z menších kusů propojených otevřenými a dokumentovanými API je „stříbrná kulka“, která vede k flexibilním a udržitelným systémům, lepšímu naplnění uživatelských potřeb a snížení provozních nákladů.

Modulární zakázky

Agilní vývoj zaměřený na uživatele, vlastnictví produktu, DevOps a modulární architektura umožňují rozdělit velké, vysoce rizikové projekty na menší a zvladatelnější zakázky. Jednotlivé kontrakty by měly být natolik malé, aby se úřad nemusel bát odebrat zakázku špatně fungujícímu dodavateli a nahradit jej jiným. (Podrobně viz sekci *Nakupujte služby, ne software*.) Zbytek dodavatelů mezitím může pracovat dál, takže celková ztráta hybnosti bude minimální. Nový dodavatel by navíc neměl mít potíže přebrat práci po předchozím, protože ten měl podle pravidel agilního vývoje dodávat co 14 dní úplný, dokumentovaný a otestovaný software. Další výhodou modularizace je, že se výsledné kontrakty mohou vejít do limitu pro malé zakázky a jsou tím pádem zatížené menší administrativní režií.

Existují dodavatelé, kteří se na tento styl práce přímo specializují. Zhruba se dá říct, že agilní vývojářský tým 5–9 lidí stojí v americkém prostředí zhruba 1–2 miliony dolarů ročně, podle přesného umístění.

Tento přístup vyžaduje koordinaci a ochotu na straně investora. Lidé zodpovědní za nákup softwaru bývají často zvyklí na tradiční přístup outsourcování IT projektů: jedna zakázka popsaná dlouhým poptávkovým dokumentem, ještě delší nabídky ze strany dodavatelů a zastaralé certifikace vesměs založené na metodice vodopád. Obecně řečeno nemají agilní týmy zaměřené na uživatele ani páru o tom, co znamená CMMI nebo EVMS – tyto standardy už se dnes za vhodnou cestu k flexibilním a nákladově efektivním systémům nepovažují. Celkově tedy jde o zbytečnou překážku pro vstup dalších dodavatelů, kteří mohou ve spolupráci se státem teprve začínat a nechce se jim investovat zdroje do sepisování nabídkových dokumentů.

Moderní procesy vývoje softwaru vychází z potřeb uživatele, používají agilní metodiku vývoje a opírají se o vlastnictví produktu, DevOps, modulární architekturu a modulární zakázky. Pochopením těchto klíčových konceptů získáte dobrou šanci řídit softwarové projekty efektivně a zvládnout zbytek tohoto textu.

Rady pro rozpočtování a řízení technických projektů

Změňte svůj pohled na rizika

Za posledních pár desítek let veřejná správa přestala pro vývoj kritických systémů používat vlastní zaměstnance a začala spoléhat na externí dodavatele. Jde o zdánlivě bezpečnější řešení, které je často motivované také omezenými vlastními kapacitami a sliby levnějších „krabicových" řešení ze strany dodavatele.

Na projektech jako Heathcare.gov jsme se ovšem naučili, že outsourcovat jde pouze práci, nikoliv riziko selhání.[6] Neúspěšné projekty jdou na vrub zadavatele, nikoliv dodavatelů. Veřejná správa je občanům zodpovědná za své fungování a její orgány tedy musí mít pod kontrolou projekty, na které fungování státu spoléhá. Pokud úřad žádá o peníze na IT systém, ve skutečnosti nepotřebuje vyřešit technický problém – potřebuje vyřešit nějaký problém spojený s fungováním státu. Technologie zde slouží jako prostředek, nikoliv jako cíl.

To neznamená, že by veřejná správa musela všechny systémy vyvíjet vlastními silami. Musí mít ale jasná očekávání ohledně lidských výstupů a technických standardů spojených s bezpečností dat, jejich používáním, interoperabilitou, sledováním a vyhodnocováním.

Zatímco technické know-how je běžně dostupné a levné, schopnost vést státní úřad je vzácná a cenná. Veřejná správa musí přijmout odpovědnost za vlastní projekty a riziko jejich selhání; externí dodavatelé by měli fungovat pouze jako najatá pomocná síla, kterou lze v případě nespokojenosti snadno nahradit.

KONTROLNÍ SEZNAM

- Projekt má vyhrazeného vlastníka s dostatečně silným mandátem. Vlastník je zaměstnanec zadavatele, nikoliv externista nebo zaměstnanec nějaké státní IT společnosti.
- Všichni účastníci projektu si uvědomují, že stávající přístup metodou vodopád většinou nevede k úspěchu a přechod k agilnímu vývoji a modulárním zakázkám toto riziko výrazně snižuje.
- Účastníci projektu nepovažují externí dodavatele za „vlastníky" projektu nebo jeho výstupů, ale za nahraditelné pomocníky.

KLÍČOVÉ OTÁZKY

- Určili a proškolili jste státního zaměstnance (nikoliv externího dodavatele!), který bude fungovat jako vlastník projektu a bude udávat jeho směr a priority a dohlížet na práci vývojářského týmu?
- Existuje v úřadu hierarchie, v rámci které může kdokoliv získat shora podporu týkající se nové metodiky vývoje? Sahá tato hierarchie až do nejvyšších pater veřejné správy? Může někdo z účastníků projektu novou metodiku bojkotovat? Pokud ano, jakým způsobem můžete tyto problémy předat výš a vyřešit je, aniž by ohrozily úspěch projektu?
- Jakým způsobem přebírá úřad zodpovědnost za vedení projektu a vlastnictví jeho výstupů? Nepřenáší riziko na externího dodavatele?

Nakupujte služby, ne software

Když poptáváte software, nekupujete *věc*, ale *službu*: práci týmu vývojářů a designérů na základě priorit určených vlastníkem produktu. Tento myšlenkový posun vede k úplně jinému – mnohem jednodušššímu – poptávkovému řízení a formulování smluv.

Váš poptávkový dokument neboli RFP (*Request for Proposal*) by měl popisovat obecný cíl projektu a měl by obsahovat první verzi produktového backlogu, tedy seznamu práce, kterou je potřeba udělat. Tento výčet sepsaný vlastníkem produktu by měl mít podobu *user stories*, tedy potřeb koncových uživatelů, a mělo by být jasně řečeno, že slouží pouze k hrubé představě o typu poptávaných prací, nikoliv jako jejich vyčerpávající a definitivní seznam. Oběma stranám musí být zřejmé, že detailní předmět práce se bude měnit v závislosti na prioritách a postupující analýze potřeb; se změnami je třeba počítat předem a pro moderní metody vývoje softwaru by neměly představovat problém.

Poptávkový dokument by měl popisovat spíše cíle projektu než výčet prací nebo podrobných vlastností produktu. Tím se vyhnete tradičním doobjednávkám změn u dodavatele, protože rozsah prací není dán předem a tým pracuje zkrátka na tom, co mu řeknete. Pokud se váš dodavatel pyšní „agilním" přístupem, ale změny má tendenci řešit pomocí dodatečných objednávek, mělo by se vám v hlavě rozsvítit červené světlo.

Abyste měli jistotu, že dodavatel naplní požadované technické specifikace, měl by váš poptávkový dokument obsahovat plán posuzování kvality (QASP, *Quality Assessment Surveillance Plan*) vhodný pro agilní vývoj. Podle tohoto plánu se na konci každého sprintu posoudí, jestli jsou odevzdané výstupy otestované, zabezpečené, přístupné, dokumentované a správně nasazené.[7] Kvalitativní požadavky může naplnit pouze skutečný, nasazený a fungující software, nikoliv sebelepší popis toho, co bude software dělat někdy v budoucnosti.

Historicky existuje velký tlak na to, aby zakázky probíhaly za pevnou cenu, protože to zdánlivě snižuje jejich rizika. Pokud ale můžete průběžně vyhodnocovat kvalitu skutečně dodaného softwaru, pak vám větší flexibilitu dávají průběžné platby za čas a náklady (se stropem pro celkové výdaje). Tento režim placení vám navíc dává jednodušší cestu ven pro případ, že dojde k zásadní změně zadání projektu, nebo když nejste spokojeni s kvalitou dodaného softwaru. Pokud práce dodavatele z libovolného důvodu nenaplňuje vaše před-

stavy, prostě mu přestanete zadávat další úkoly, kontrakt tím efektivně vzato zanikne a můžete zkusit jiného dodavatele.

KONTROLNÍ SEZNAM

- Projekt má vyhrazeného vlastníka s dostatečně silným mandátem. Vlastník je zaměstnancem zadavatele, nikoliv externistou nebo zaměstnancem nějaké státní IT společnosti.
- Osoba zodpovědná za podpis zakázky se staví pozitivně k projektu samotnému i k novým metodám vývoje softwaru.
- Poptávkový dokument je formulován čistě jako poptávka vývojářských služeb, nikoliv jako poptávka konkrétního výstupu.
- Poptávkový dokument požaduje multifunkční tým, který kromě vývojářů obsahuje například designéry nebo experty na UX.
- Poptávkový dokument nemá více než 20 stránek.
- Poptávkový dokument obsahuje backlog alespoň s desítkou user stories.
- Platby budou probíhat průběžně podle času a nákladů, se stropem pro celkové výdaje.
- Pro výběr dodavatelů bude použit nejjednodušší možný nástroj, který vám umožní oslovit vhodné dodavatele.

KLÍČOVÉ OTÁZKY

- Má produktový vlastník mandát dělat rychlá autoritativní rozhodnutí za celého zadavatele?
- Je produktový vlastník připraven investovat do své role většinu svého pracovního času?
- Je vedení zadavatele připraveno řídit produkt podle zjištěných potřeb koncových uživatelů, na základě přímých konverzací s těmito uživateli? Anebo chce řídit produkt podle vlastních osobních preferencí?

- Mluví poptávkový dokument jasně o pravidelném odevzdávání fungujícího kódu, dokumentace a testů? Říká jasně, že veškeré pracovní výstupy budou majetkem státu?

Pozor na past krabicového softwaru na míru

Běžný krabicový komerční software (*Commercial off-the-shelf Software*, COTS) a software poskytovaný jako služba (*Software as a Service*, SaaS) nabízí výbornou příležitost, jak pořídit nový software nebo infrastrukturu rychle, bez nutnosti vyvíjet software na zelené louce. Pokud například potřebujete textový procesor, dává bezvadný smysl koupit standardní krabicový produkt.

Pokud ale poptáváte velký, specializovaný systém, bez kterého se neobejdete, dávejte si na nabídky běžného krabicového softwaru velký pozor. Dodavatelé velice často prezentují svůj komerční, „na míru upravitelný" software jako všelék, který zvládne všechny vaše jedinečné regulatorní a procesní požadavky. A nejspíš mají pravdu – ovšem většinou jen za cenu výrazných úprav.

Než tedy podobný software zakoupíte, poptejte se jiných státních úřadů, jaké s ním mají zkušenosti. Velmi pravděpodobně se dozvíte, že produkt, který byl prezentován jako hotové řešení, si nakonec na úpravách vyžádal mnohem víc času a peněz, než se původně předpokládalo.

Při plánování rozpočtu nediktujte konkrétní řešení, dejte úřadu prostor pro výběr, které části systému si může koupit a které postavit. Pokud rozpočet vyžaduje krabicový software, úřad může skončit uvázaný k vysoce upravované verzi krabicového produktu, která nepůjde bez velkých nákladů upgradovat. A podobně pokud rozpočet předepisuje použití SaaS řešení, může tím úřad natlačit do nevyhovujícího SaaS systému, u kterého utratí podstatné peníze za dodatečného

„softwarového integrátora“ a propojení se stávajícími systémy. V obou případech dochází k nežádoucímu *vendor lock-inu*, závislosti na jednom dodavateli.

Běžný komerční software skutečně může být dobrým základem i pro velké nové systémy. Ale zadavatel musí k jeho volbě přistupovat s otevřenýma očima a vědomím, že v něm velmi pravděpodobně nekupuje specializované řešení na klíč.

KONTROLNÍ SEZNAM

- Rozpočet projektu nepředepisuje použití krabicového softwaru, SaaS, ani softwaru na míru, ale dává zadavateli možnost vybrat libovolnou kombinaci těchto přístupů podle potřeby.
- Tvrzení dodavatele o tom, že jeho krabicový software nebo SaaS bude fungovat i bez nákladných úprav, je nezávisle ověřeno u jiných států nebo úřadů, kde tyto produkty používají a mají s jejich úpravami zkušenosti.

KLÍČOVÉ OTÁZKY

- Pokud si necháváte upravit krabicový software na míru, jak budou probíhat jeho budoucí aktualizace? Kolik další práce bude potřeba na integraci těchto úprav a kdo za to zaplatí?
- Co se stane, když váš poskytovatel SaaS jednoho dne zčistajasna skončí?
- Bude mít stát bezplatný a snadný přístup ke svým datům, datovým modelům a API?

Chtějte důkazy místo slibů

Historicky se pokrok v softwarových projektech nejčastěji měřil porovnáváním dokončené práce s časovým harmonogramem napsaným

na začátku projektu, k čemuž slouží nástroje jako Ganttovy diagramy nebo seznamy hotových úkolů. Tento přístup má jednu nevýhodu: nefunguje. A právě na tom, že nefunguje, je založený celý agilní vývoj. Moderní softwarové týmy vůbec netuší, co znamenají zkratky jako CMMI nebo EVM, a nebudou mít zájem o projekty, které něco podobného vyžadují.

Místo abyste pokrok poměřovali nějakým grafem, který za tím účelem vznikl, je lepší sledovat objem skutečně hotové práce. Přidejte se na konci každého sprintu k závěrečné poradě, kde se celému projektovému týmu a přizvaným koncovým uživatelům předvádí práce dokončená v rámci sprintu. Vyzkoušejte si hotový web. Nainstalujte aplikaci. Shánějte se po takzvaném diagramu spalování (*burn down chart*), který ukazuje množství zbývající práce a potřebného času.

Důležitou pojistkou proti iluzornímu pokroku je také plán posuzování kvality (*Quality Assurance Surveillance Plan*, QASP), který by měl být součástí smlouvy s dodavatelem. Podle tohoto plánu se na konci každého sprintu kontroluje, jestli veškerá odevzdaná práce odpovídá předepsaným standardům. Plán jasně říká, jakým způsobem stát zjistí, jestli je práce dostatečně kvalitní – a tím pádem předchází i nepříjemným překvapením pro dodavatele.

Plán posuzování kvality nevyžaduje vznik výstupů čistě pro kontrolu průběhu práce – nejlepší je prostě zkoušet, *jestli výsledek skutečně funguje*. Jde o radikálně odlišný přístup k hlídání pokroku technologických projektů. Má výhodu v tom, že je objektivnější, měřitelnější a funkčnější než subjektivní či právnické interpretace nějakých sáhodlouhých seznamů systémových požadavků.

KONTROLNÍ SEZNAM

- Vlastníkem produktu je vyhrazený státní zaměstnanec s dostatečně silným mandátem.
- Neexistují žádné plánovací nebo reportovací požadavky, které by byly v rozporu s agilním vývojem. Neexistují tedy například žádné

termíny, do kterých by měly být hotové konkrétní úkoly, a neexistuje podrobná specifikace požadovaných funkcí – ani v poptávkovém dokumentu, ani v akvizičním plánu, ani v legislativě.
- Na konci každého sprintu zkontroluje státem zaměstnaný vývojář, jestli dodaná práce splňuje měřítka definovaná plánem posuzování kvality.
- Všichni nadřízení vlastníka produktu jsou ochotni posuzovat pokroky projektu primárně v podobě demonstrací fungujícího softwaru, diagramu spalování a přehledu hotových a zbývajících user stories.
- V úřadu je předem určen člověk, který je připraven opakovaně vysvětlovat pokroky projektu všem nadřízeným, pro které může být způsob sledování pokroku agilních projektů na rozdíl od vodopádu nezvyk.

KLÍČOVÉ OTÁZKY

- Jste schopni nabídnout všem účastníkům projektu podporu při přechodu na takto radikálně odlišný způsob měření pokroku projektu? Od samotných úřadů až po vládu a parlament? Může se někdo z nich zablokovat a trvat na Ganttově diagramu, takže by agilní vývoj nepřipadal v úvahu?
- Kdo bude mít na starost prezentaci výsledků mimo úřad, tedy například vládě?

Investujte do vlastních lidí

Lidé bez zkušeností s vývojem softwaru nemohou kvalifikovaně vést softwarové projekty ani rozhodovat o jejich rozpočtech. Stát musí zajistit, aby úřady zaměstnávaly dostatek lidí s odpovídajícími zkušenostmi.

Je sice lákavé „vyřešit“ tento problém spoléháním na zaměstnance nějaké centrální státní IT agentury nebo na dodavatele, ale jedině sám zadavatel je schopen přesně posoudit vlastní potřeby, vysvětlit je dodavatelům a následně zkontrolovat jejich výstupy. Proto nutně potřebuje zvládnout související know-how.

Snažte se zjistit, jestli lidé zodpovědní za vedení a rozpočet projektu mají odpovídající technické zkušenosti. S výjimkou těch úplně nejmenších úřadů má prakticky každý někoho technicky orientovaného, kdo může vedení projektu doplnit – ačkoliv naopak prakticky nikdo k tomuto účelu nezaměstnává přímo softwarové vývojáře.

Pokud nemáte potřebné know-how sami, musíte zaměstnat někoho, kdo ho přinese, i kdyby to mělo být jen na určitou dobu nebo externě. Nejlepší je vývojář nebo designér se zkušenostmi ve vývoji moderního softwaru, ideálně pro veřejný sektor. Také můžete jednoho nebo víc zaměstnanců poslat na školení základů agilního vývoje softwaru, po celých Spojených státech je řada programátorských „kempů“ a existují i online varianty.

Náklady na zaměstnání vývojáře nebo zvýšení kvalifikace stávajících zaměstnanců jsou ve srovnání se státními výdaji za IT směšné. A jakmile váš zaměstnanec jednou zažije agilní projekt od začátku do konce, rozhodování o rozpočtech dalších softwarových projektů už pro něj bude jednodušší.

Analogicky musí úřad zaměstnat dostatek vývojářů k tomu, aby dokázal posoudit a zkontrolovat výstupy od dodavatelů. Zde vývojáři dohlíží na kvalitu výstupů a kontrolují, aby dodavatelé pracovali, na čem mají.

Software sice v principu není nikdy „hotový“ – vždy bude potřeba ho měnit podle nových technologií, předpisů, regulací, zákonů a uživatelských potřeb –, ale časem přijde chvíle, kdy vám na další práci bude stačit výrazně menší počet vývojářů. Zvlášť v tomto okamžiku je pak důležité mít několik vlastních lidí, kteří danému softwaru rozumí a budou schopni jej udržovat.

U větších projektů je potřeba vývojářský tým nasmlouvat na dobu neurčitou, pod dohledem státního vlastníka projektu. U vodopádu by se této fázi říkalo „provoz a údržba“, ale v agilním vývoji jde o standardní mix prací: analýza uživatelských požadavků, design, vývoj softwaru, atd.[8]

KONTROLNÍ SEZNAM

- O rozpočtu projektu rozhodují lidé se zkušenostmi v agilním zakázkovém vývoji větších softwarových systémů.
- Pokud nikoho takového nemáte, stát má pro tyto příležitosti smlouvu s vhodným dodavatelem, který může know-how přinést externě. Tento dodavatel nesmí být ve střetu zájmů, tedy nesmí pro stát dělat nic jiného a nesmí být ve spojení s dodavateli krabicového softwaru.
- Úřad má určeného státního zaměstnance, který projekt technicky povede. Tento zaměstnanec musí mít prokazatelné zkušenosti se zakázkovým agilním vývojem softwaru.

KLÍČOVÉ OTÁZKY

- Když dodavatel na konci každého sprintu odevzdává hotový kód, který konkrétní státní zaměstnanec má na starosti kontrolu jeho kvality?
- Pokud úřad říká, že potřebuje 10 milionů dolarů na dokončení konkrétního softwarového projektu, kdo z lidí zodpovědných za rozpočet má zkušenosti na to, aby tuto částku posoudil?
- Až bude software hotový, kdo ho bude udržovat? Má úřad vlastní zaměstnance, kteří toho jsou schopni? Zúčastní se procesu vývoje, aby systému rozuměli a dokázali ho podporovat?

Minimalizujte cenu změn

Váš stát bude existovat déle než libovolný software. Což znamená, že z vašeho krásného nového softwaru bude jednou pravděpodobně ošklivý starý a špatně použitelný software.

Sebelepší software bude časem potřeba nahradit, ať už částečně, nebo zcela. A pokud software kupujete jako jeden velký monolit, zaručeně časem přestane plnit potřeby úřadu.

Technologie se mění – stejně jako politika, regulace, zákony a priority. Libovolný projekt naplánovaný podrobně předem nebude schopný se těmto změnám přizpůsobit a riskuje neúspěch nebo výrazné překročení časového a finančního rozpočtu.

Místo abyste kupovali jeden velký kus proprietárního softwaru, trvejte na tom, aby váš dodavatel standardně používal open-source software a architekturu orientovanou na služby (*service-oriented architecture*, SOA). Od samotného začátku tím snížíte náklady na aktualizaci a změny systému.

KONTROLNÍ SEZNAM

- Systémy, ať už původně cloudové nebo právě do cloudu stěhované, používají architekturu orientovanou na služby (SOA), která není vázaná na konkrétní produkty nebo dodavatele.
- Data jsou v zájmu přenositelnosti uložená v otevřených, nepatentovaných formátech podporovaných více dodavateli.
- API používají otevřená schémata.
- Kdekoliv je to možné, místo komerčního softwaru se používá open-source software, který snižuje riziko závislosti na jednom dodavateli.
- Stát vlastní všechny výstupy od dodavatele.
- Pokud jsou některé komponenty systému řešené běžným krabicovým softwarem, jeho dodavatel poskytuje smluvně i technicky

možnost přejít ke konkurenci a nabízí přiměřeně nákladnou možnost exportovat všechna uložená data.

KLÍČOVÉ OTÁZKY

- Jaký je váš plán snižování časových a finančních nákladů na budoucí aktualizace systému kvůli změnám technologií, politiky nebo dodavatelů?
- Kolik budou stát změny systému kvůli změnám technologií nebo politiky?
- Jsou API otevřená a použitelná jinými dodavateli?
- Jsou používané datové formáty standardizované, otevřené a použitelné jinými dodavateli?
- Udržování softwaru ve formě vyžaduje pravidelnou a průběžnou práci – máte její plán?

Měřte pokrok na iterace, ne na milníky

Hodnota by neměla vzniknout až na konci projektu – koncoví uživatelé by ji měli začít postupně získávat nejpozději šest měsíců od uzavření smlouvy. Už na konci prvního sprintu musí zadavatel dostat ke kontrole první fungující kód a následně se tento proces bude opakovat s každým dalším sprintem. Koncoví uživatelé by měli kontrolovat výsledky každého sprintu bez ohledu na to, jestli došlo k nasazení nové produkční verze pro běžné použití.

Nepoměřujte pokroky projektu story pointy, počtem řádků kódu, člověkohodinami ani jinými podobnými metrikami. Jediným opravdu smysluplným měřítkem úspěchu je hodnota dodaná koncovým uživatelům. A tu nejlépe posoudíte tím, že si na konci každého sprintu

v rámci závěrečné porady promluvíte se *scrum masterem*[9] a státním vlastníkem produktu.

KONTROLNÍ SEZNAM

- Dodavatelský tým používá agilní metodiku vývoje.
- Dodavatel má povinnost na konci každého sprintu nasadit novou verzi funkčního softwaru na státem vlastněnou infrastrukturu.
- Projektový tým pravidelně mluví s koncovými uživateli a konzultuje s nimi své výsledky, jednak kvůli ověření funkčnosti hotové práce, jednak kvůli plánování nové.
- Poptávkový dokument neobsahuje podrobný harmonogram prací, zmínky o Ganttově diagramu ani formální závazky typu IV&V a podobně.
- Existuje předem určená osoba, která má na starost pravidelně docházet na hodnocení sprintů a informovat vedení projektu o jeho pokrocích.

KLÍČOVÉ OTÁZKY

- Dokáže zadavatel během šesti měsíců dodat koncovým uživatelům nějakou hodnotu? V čem přesně tato hodnota spočívá?
- Je zadavatel připravený na to, že dodavatel bude neustále průběžně mluvit s koncovými uživateli softwaru, potenciálně tedy zaměstnanci zadavatele?

Omezte celkové výdaje

Čím je projekt dražší, tím menší má šanci uspět. Obecně řečeno byste za celý projekt neměli utratit více než 10 milionů dolarů.[10] (Vzácnými výjimkami z tohoto pravidla jsou extrémně složité projekty jako na-

příklad pojištění nezaměstnanosti nebo systémy týkající se programu Medicaid.)

KONTROLNÍ SEZNAM

- Zadavatel chápe, že nedostává zlomek potřebných peněz – dostává úplně nový proces vývoje softwaru a k němu adekvátní systém financování.

KLÍČOVÉ OTÁZKY

- Pokud projekt „vyžaduje" 20 milionů dolarů, kolik hodnoty lze uživatelům dodat za polovinu nebo desetinu této částky? (Pokud „žádnou", projekt je ztracený.)
- Pokud vlastní náklady doplní dotace shora, zvlášť v nějakém výhodném poměru (například 9:1 v programech Medicaid), bude mít někdo problémy, když projekt nevyčerpá všechny dostupné peníze?
- Je mezi účastníky projektu někdo, jehož odměny se odvíjí od proinvestovaných prostředků a má tedy motivaci utratit sto milionů tam, kde by deset stačilo?

Omezte velikost zakázek

Dlouhodobá spolupráce s jedním dodavatelem nebo využití většího počtu týmů od jednoho dodavatele zní lákavě, ale nevyhnutelně vede k závislosti na tomto dodavateli. Rozdělením projektů na menší zakázky motivujete dodavatele k vývoji udržitelnějšího softwarového ekosystému namísto monolitu, a navíc mají menší zakázky znatelně větší šanci na úspěch.[11]

Vyžadujte, aby žádná jednotlivá zakázka nestála ročně víc než dva miliony dolarů a žádná smlouva netrvala déle než tři roky. Tím pádem nedostanete od jednoho dodavatele víc než dva vývojářské týmy –

pokud jich projekt vyžaduje víc, sežeňte je u jiného dodavatele a nechte je pracovat samostatně. Omezte také délku poptávkového dokumentu, neměli byste do něj investovat víc než 60 dní a neměl by zabrat víc než 20 stránek.

Kromě ochrany před závislostí na jednom dodavateli mají menší zakázky ještě jednu výhodu: mají menší pravděpodobnost, že je někdo napadne, protože při jejich pořizovací hodnotě se nevyplatí do nich investovat úsilí a právníky. Pokud k dodavatelům přistupujete otevřeně a s respektem a nevyžadujete po nich stovky stránek nabídkové dokumentace, je větší šance, že s vámi budou chtít spolupracovat i do budoucna.

S rostoucím počtem lidí na projektu roste objem času, který jim zabere vzájemná koordinace. Řešením je pracovat souběžně, bez větší koordinace, což se dá zařídit stavěním systému z volně provázaných komponent. Pokud na vašem projektu pracují týmy od více než jednoho dodavatele, máte také o něco lepší možnosti, když bude potřeba některého z dodavatelů vyměnit.

KONTROLNÍ SEZNAM

- Pokud bude projekt vyžadovat větší počet zakázek, došlo k analýze rozsahu první z nich a máte aspoň hrubou představu, co budou obsahovat ty ostatní.
- Pokud bude na projektu pracovat více než jeden vývojářský tým, systém využije architekturu orientovanou na služby (SOA).
- Pokud je to možné, projekt je rozdělen na podlimitní zakázky nebo zakázky malého rozsahu, které mají rychlejší a jednodušší administrativu.
- První identifikovaný projekt má relativně nízkou technickou složitost, nízké politické riziko a vysokou hodnotu pro koncové uživatele, takže na něm týmy mohou vyzkoušet nový styl práce, experimentovat a učit se v prostředí s relativně nízkým rizikem.

KLÍČOVÉ OTÁZKY

- Přečetla si osoba zodpovědná za zadávání zakázek tohoto průvodce?
- Chápe dotyčná osoba, že po ní nikdo nechce udělat veškerou práci na zakázce za 50 milionů dolarů? Chápe, že dvoumilionové zakázky je mnohem jednodušší zpracovat a agilně bude též jednodušší je uřídit?

Plaťte za ekosystém, ne za monolity

Dávejte si pozor, abyste jeden zastaralý systém nenahradili jiným zastaralým systémem. Trvejte na systému volně provázaných komponent, které se dají postavit postupně. Výsledný systém pak v budoucnu nebudete muset nahrazovat jednorázově, ale bude možné postupně obměňovat zastarávající komponenty podle potřeby.

KONTROLNÍ SEZNAM

- Každá zakázka musí přinést nějakou hodnotu koncovým uživatelům – nemůže jít jen o „údržbu serverů“ nebo „nastavení databáze“, ale třeba o „webovou aplikaci pro správu oprávnění“ nebo „zjednodušení náborového procesu“.
- Neexistuje jediný „enterprise architekt“, architektura se objeví postupně během agilního vývoje.
- Pokud je projekt dost velký na to, aby na něm pracovalo několik agilních týmů zároveň, nikdo neočekává, že všichni členové všech týmů budou chodit na společné porady.
- Poptávkový dokument předepisuje použití architektury orientované na služby.

KLÍČOVÉ OTÁZKY

- Existuje nějaké slabé místo, jehož vyřazením by došlo k výpadku celého systému? Pokud ano, systém je spíš monolit než síť služeb.
- Pokud potřebujete vyřadit jednoho dodavatele kvůli nízké kvalitě výstupů, mohou ostatní pracovat bez přerušení?

Rozšiřte okruh svých dodavatelů

Vaši stávající dodavatelé se nejspíš do moderního stylu vývoje softwaru popisovaného v tomto textu dvakrát nepohrnou – začali jste s nimi kdysi spolupracovat právě pro styl práce, kterého se teď chcete zbavit. Bude potřeba najít a přilákat nové firmy, které používají moderní metodiky vývoje.

Menší, kvalifikovaný agilní tým velmi pravděpodobně najdete i ve svém státě, pokud je to pro vás priorita.[12] Pokud ale chcete snížit náklady, je důležité zvážit i spolupráci se vzdálenými a distribuovanými dodavatelskými týmy.

Se státech jako Kalifornie, Washington, New York, Virginie nebo Maryland může agilní tým snadno stát dvakrát tolik co tentýž tým na středozápadě nebo severu Ameriky. To je rozdíl milionu dolarů ročně, bez dopadu na kvalitu. Vyplatí se tedy zvážit, jak moc stojíte o místní tým, případně tým ze svého státu.[13] Podporou distribuovaných týmů získáte kromě jiného přístup k mnohem větší zásobárně talentu, takže se vyplatí do distribuované spolupráce investovat[14] a návštěvy na místě vyžadovat jen v případě potřeby, například kvůli kontaktu vývojářů s koncovými uživateli.

A jak ony malé kvalifikované týmy najít? Řada měst a států má přímo seznamy vhodných agilních dodavatelů, viz například pravidelně aktualizovaný seznam kalifornského ministerstva informatiky. Zkuste

některé z těchto dodavatelů oslovit v budoucích poptávkách. Poptejte se také mezi svými kolegy z jiných státních úřadů, jestli by vám vhodného dodavatele nedoporučili. Anebo se zkuste vžít do role softwarového vývojáře, co hledá práci, a projděte si známé weby s nabídkami pracovních příležitostí a networkingu – velmi pravděpodobně na nějakého agilního dodavatele ze svého státu narazíte. Celý proces nemusí trvat déle než pár hodin.

Častou praxí je dávat přednost dodavatelům, kteří už poptávaný systém jednou postavili. Vůbec to není nutné, jen si tím omezujete výběr na několik velkých mezinárodních firem. Lepší je rozšířit záběr na společnosti, které úspěšně dodaly nějaký analogický systém – pokud někdo například vyrobil rezervační web pro půjčovnu aut, jistě by uměl udělat například web pro turistické povolenky, a hlavní vývojář databáze pro sledování komet by jistě uměl udělat databázi pro sledování polohy státních vozidel. Hledáním relevantních zkušeností po této ose jistě najdete řadu vhodných dodavatelů, kteří by vaši zakázku zvládli.

KONTROLNÍ SEZNAM

- Poptávkový dokument je stručný (maximálně 20 stránek) a srozumitelný i pro vývojáře, kteří pro stát běžně nepracují.
- Akviziční plán počítá s aktivním oslovením malých dodavatelů.
- Poptávkový dokument není na webu zamčený za registračním formulářem, ale je volně k dispozici, aby ho komunita dodavatelů mohla snadno sdílet.
- Poptávkový dokument od dodavatele vyžaduje, aby ve své nabídce uvedl klíčové odpovědné osoby (nejvýše tři), například hlavního vývojáře nebo hlavního designéra.
- Akviziční plán počítá s tím, že si s několika nejslibnějšími dodavateli osobně promluvíte, a to nikoliv s jejich obchodními zástupci, ale s klíčovými vývojáři.

- Zaměstnanci dodavatele nejsou nuceni pracovat na místě u zadavatele.
- Dodavatelské týmy a státní vlastník produktu mají svolení používat videohovory, chat, veřejný verzovací systém a jiné nástroje běžně používané agilními týmy, viz například seznam nástrojů pro vzdálené zaměstnance GSA.

KLÍČOVÉ OTÁZKY

- Plynou vám nějaké výhody – politické či jiné – z toho, že zakázku zadáte v rámci státu? Nebo tuto podmínku máte přímo jako součást zadání? Pokud ano, může vás to omezit při hledání vhodného dodavatele?
- Stojí vám úspora jednoho milionu dolarů na každý agilní tým za to, abyste překonali námitky proti vzdáleným týmům?
- Udělali jste si aspoň letmý průzkum trhu, abyste věděli, jaké dodavatele oslovit svým poptávkovým dokumentem – anebo pouze naslepo poptáváte a doufáte?

Podělte se o svůj software s ostatními

Hotový software – ať už jako celek, nebo některou svou komponentou – by pravděpodobně mohl být užitečný pro jiné státní úřady. V mnoha státech navíc platí legislativa, díky které je software vytvořený státem automaticky veřejným dílem.

Otevřeně vyvíjený software lépe motivuje zaměstnance dodavatele – nabízí jim dobrou referenci do portfolia pro budoucí zaměstnavatele, případně i jen možnost pochlubit se kamarádům. Motivovaný dodavatel pro vás odvede lepší práci. A při zadávání budoucích zakázek se

můžete odkázat na volně dostupný kód, takže zájemci uvidí, na čem přesně nebo proti jakému API by pracovali.

KONTROLNÍ SEZNAM

- Poptávkový dokument požaduje, aby byl zdrojový kód od prvního okamžiku vyvíjen veřejně na nějaké sociální platformě pro vývojáře, například GitHubu nebo GitLabu.
- Poptávkový dokument vyžaduje, aby byl zdrojový kód jasně označen jako veřejné dílo nebo uvolněn pod nějakou open-source licencí.
- Poptávkový dokument vyžaduje dodržování pravidel bezpečnosti, zejména tedy důsledné oddělení softwaru od dat a konfigurace (například přístupových hesel), což by mělo být ověřeno automatickými testy.
- Poptávkový dokument vyžaduje dostatečnou dokumentaci, aby projekt dokázal spustit i vývojář, který na něm nepracuje.

KLÍČOVÉ OTÁZKY

- Nebude vám někdo – například osoba zodpovědná za bezpečnost projektu – dělat problémy kvůli zveřejňování kódu pod open-source licencí?
- Víte o nějakých jiných státních úřadech nebo organizacích, které by z vašeho softwaru mohly profitovat? Můžete s nimi zadání a vývoj projektu konzultovat?

Berte software jako provozní výdaj

Zakázkový software, na rozdíl od mostů nebo jiných kapitálových infrastrukturních projektů, není nikdy „hotový“, takže je potřeba počítat s jeho průběžnými úpravami. Jen tak můžete naplnit dnešní potřeby uživatelů, nikoliv ty včerejší.

U malých systémů to znamená přidat k vašemu vývojářskému týmu řekněme do jednoho úvazku navíc. U velkých, klíčových systémů to může znamenat celý dodatečný tým, který se bude starat o údržbu a rozvoj.

Na údržbu softwaru se z hlediska rozpočtu obvykle hledí úplně jinak než na samotný vývoj, což je chyba. Udržovat software znamená jednoduše pokračovat v jeho úpravách podle potřeb uživatelů, které se mění podle zákonů, regulací, politik, doporučených praxí nebo nových technologií. Jsou k tomu potřeba tytéž dovednosti, metody a úkoly, jaké byly potřeba během vývoje. Takže pokud vám někdo navrhuje převést projekt do nějaké speciální fáze provozu a údržby (*operations and maintenance*, O&M), mělo by se vám v hlavě rozsvítit červené světlo.

Obecně řečeno vás agilní tým 5–9 vývojářů bude stát 1–2 miliony dolarů ročně v závislosti na tom, odkud přesně je. Financování se dá postupně navýšit během několika rozpočtových cyklů podle toho, jak se vám bude dařit úspěšně snižovat rizika, kontrolovat výdaje a dodávat iterativně hodnotu koncovým uživatelům.

Ve výsledku tím můžete získat předvídatelný zdroj financování softwarových projektů a nahradit jím nepředvídatelné kapitálové výdaje. A váš zřizovatel na oplátku získá předvídatelné roční náklady na všechny vaše softwarové projekty.

KONTROLNÍ SEZNAM

- Zadavatel si uvědomuje, že software je nutné zlepšovat průběžně po celou dobu jeho existence, protože „údržba“ se kvalitativně nijak neliší od vývoje.
- Zadavatel poptává agilní vývojářské služby.
- Zeptali jste se zadavatele, jestli by nechtěl finance na projekt dostávat průběžně v rámci provozního rozpočtu, namísto jedné velké sumy.
- Tento režim financování byl konzultován se všemi potřebnými nadřízenými organizacemi.
- Pokud je projekt vysoce rizikový, dostane v prvním roce pouze několik málo milionů dolarů a financování se bude navyšovat až podle hodnoty dodané uživatelům.

KLÍČOVÉ OTÁZKY

- Budou vyžádané prostředky utraceny během jednoho rozpočtovacího období?
- Pokud zadavatel žádá 50 milionů dolarů, kolik hodnoty lze koncovým uživatelům dodat za dva miliony, další dva miliony, a tak dále?
- Pokud je projekt financovaný federálními dolary, je federální agentura ochotná vzít v úvahu provozní model financování?

Ptejte se dodavatele na technické věci

Když úřad poptává rozpočet na zakázkový vývoj softwaru, často dochází k tomu, že spolu o technických věcech mluví dvě skupiny netechnických lidí. V takové situaci pochopitelně jen zřídka padnou

klíčové otázky jako například ty, které jsou uvedené v tomto textu. Všechny podobné technické otázky ale padnout musí – a musí na ně padnout i správné odpovědi.

Přidělení peněz na projekt, který je předem odsouzen k nezdaru, nikomu neprospěje. A pokud zadavatel přesně neví, co chce, velmi pravděpodobně to také nedostane.

KONTROLNÍ SEZNAM

- Úřad postupuje podle návodu Digital Services Playbook.
- Pokud má být výstupem projektu web, úřad dodavatele odkáže na oficiální webový design systém Spojených států.
- Úřad postupuje podle Deseti přikázání softwaru.
- Úřad si prostudoval tento text.
- Úřad prošel testem rádobyagilního vývoje.

KLÍČOVÉ OTÁZKY

- Co přesně chce úřad koupit? Proč? Kdo z toho bude mít užitek?
- Které části systému budou psané na míru? Které budou řešené běžným, nepřizpůsobeným krabicovým softwarem? Kolik budou stát aktualizace tohoto softwaru? Co se stane, když dodavatel nějaké komerční komponenty, například databázového systému, skončí s podnikáním?
- Kdo jsou koncoví uživatelé vašeho systému? Mluvili jste s nimi? Co chtějí?
- Jste připraveni reagovat na změny?
- Kolik bude stát přechod na nový systém?
- Co děláte pro to, abyste se v budoucnosti vyhnuli nákladným požadavkům na změny?

Dodatek: Na co se ptát

Při posuzování rozpočtu na nový zakázkový software není praktické procházet celý předchozí text a hledat v něm vhodné otázky. Zde vám nabízíme několik základních, otevřených otázek, kterými si můžete ověřit, jestli váš projekt směřuje k úspěchu.

JAKÉ JSOU CÍLE PROJEKTU? JAKÉ VÝSLEDKY JSOU PRIORITNÍ?

Špatná odpověď: Cokoliv technického.
Správné odpověď: Aspoň jedna konkrétní uživatelská potřeba.

JAKÉ UŽIVATELSKÉ POTŘEBY PROJEKT NAPLŇUJE?

Špatná odpověď: Cokoliv mimo jasné uživatelské potřeby získané přímo od uživatelů.
Správná odpověď: Úřad pomocí rozhovorů s koncovými uživateli analyzoval jejich konkrétní potřeby a dokáže několik z nich jasně popsat.

POKUD VYBRANÝ DODAVATEL NESPLNÍ PŘEDSTAVY, JAK OBTÍŽNÉ BUDE SMLOUVU UKONČIT? JAK DLOUHO BUDE TRVAT JEHO NAHRAZENÍ JINÝM DODAVATELEM? KOLIK TO ASI BUDE STÁT?

Špatná odpověď: „Smlouvu bychom vypovídali neradi. Najít nového dodavatele by trvalo měsíce nebo roky a vyžadovalo by to velké úsilí našich zaměstnanců, takže by to projekt zdrželo o řadu měsíců. Pokud se rozhodneme vyměnit dodavatele, znamenalo by to do velké míry začít úplně znova.“

Správná odpověď: „Jsme domluveni na účtování podle času a nákladů, takže stačí dotyčnému dodavateli přestat zadávat další úkoly, čímž efektivně vzato dojde k zániku smlouvy. Když znova zveřejníme poptávkový dokument, nového dodavatele bychom mohli mít do šesti týdnů. Zátěž našich zaměstnanců bude minimální, zpoždění projektu pouze oněch šest týdnů."

BUDE POPTÁVKOVÝ DOKUMENT OBSAHOVAT PODROBNOU SPECIFIKACI FUNKCÍ SYSTÉMU?

Špatná odpověď: „Strávili jsme poslední rok analýzou podnikatelských požadavků a máme do poptávkové dokumentace nachystané stovky požadavků, abychom měli jistotu, že dostaneme, co potřebujeme."
Správná odpověď: „Soustředíme se spíš na cíle, kterých chceme pomocí systému dosáhnout. Místo podrobné specifikace požadavků máme základní várku user stories v backlogu, podle kterého se může tým pustit do práce."

JAK DLOUHÝ BUDE POPTÁVKOVÝ DOKUMENT?

Špatná odpověď: „Dali jsme dohromady několik set stran požadavků plus dalších 50 stran standardních smluvních podmínek."
Správná odpověď: „Máme necelých 20 stránek a rádi bychom se rozsahově vešli do podlimitní zakázky, abychom snáz a rychleji našli vhodného dodavatele."

PŘEDPOKLÁDÁTE PEVNOU CENU ZA CELOU ZAKÁZKU, NEBO PRŮBĚŽNÉ PROPLÁCENÍ ČASU A NÁKLADŮ?

Špatná odpověď: „Pevnou cenu, jinak se náklady nedají uhlídat."
Správná odpověď: „Průběžné placení času a nákladů, protože nám dává největší flexibilitu při řešení uživatelských potřeb a reagování na

nečekané technické potíže. A kromě toho nám umožňuje snadno vyměnit nefunkční dodavatele, aniž by to ohrozilo úspěch projektu.“

JAKOU HODNOTU ZÍSKAJÍ UŽIVATELÉ BĚHEM PRVNÍHO PŮL ROKU?

Špatná odpověď: „Žádnou, za šest měsíců se nedá nic stihnout. Systém nasadíme, až bude hotový.“
Správná odpověď: Konkrétní příklady funkcionality.

KDO BUDE VLASTNÍKEM PRODUKTU?

Špatná odpověď: „Co je vlastník produktu?“
Správná odpověď: Jméno konkrétního člověka – anebo že pro tuto roli školí některého z interních zaměstnanců.

JAKOU POUŽÍVÁTE METODIKU VÝVOJE?

Špatná odpověď: „Vodopád“ anebo jakákoliv odpověď, ze které plyne nepochopení otázky.
Správná odpověď: „Agilní vývoj“, „extrémní programování“ nebo „Scrum“.

KDO Z TÝMU, KTERÝ ZAKÁZKU PŘIPRAVUJE, MÁ ZKUŠENOSTI S VÝVOJEM SOFTWARU?

Špatná odpověď: „Nikdo.“
Správná odpověď: Jméno konkrétního člověka.

JAK ČASTO SE BUDE SYSTÉM NASAZOVAT DO PRODUKCE?

Špatná odpověď: „Až bude hotový.“
Správná odpověď: „Na konci každého sprintu.“

BUDE PROJEKT OBSAHOVAT AUTOMATICKÉ TESTY? INTEGRAČNÍ TESTY? AUTOMATICKÉ NASAZENÍ? BEZPEČNOSTNÍ TESTY?

Špatná odpověď: „Pracujeme na tom."
Správná odpověď: „Ano, od začátku."

KOLIK BUDOU STÁT ZMĚNY PROJEKTU?

Špatná odpověď: Jakákoliv zmínka o objednávkách změn (*change request*, CR).
Správná odpověď: „Systém se bude tak či onak neustále měnit na základě nových požadavků od koncových uživatelů, nových technologií, nových zákonů. Proto nás platíte podle času a nákladů, a proto děláme agilní vývoj, aby vás změny nestály zbytečné peníze navíc."

JAK ZJISTÍTE, JESTLI PROJEKT POKRAČUJE DOBŘE A JESTLI DODAVATELÉ PRACUJÍ, JAK MAJÍ?

Špatná odpověď: „Najali jsme si externí dozor, experta na nezávislou verifikaci a validaci (IV&V), který nám posílá měsíční reporty."
Správná odpověď: „Dodavatelé nám musí pravidelně předvádět fungující software, který odráží naše priority, splňuje naše technické standardy a splňuje požadavky koncových uživatelů. Pokud nedochází k plnění standardů nebo dodavatel během šesti měsíců nezačne plnit požadavky uživatelů, rozloučíme se s ním."

KOMU SOFTWARE PATŘÍ?

Špatná odpověď: „Dodavateli."
Správná odpověď: „Státu", případně „jde o veřejné dílo".

Poznámky

1. Statistika mluví o projektech v hodnotě nejméně 6 milionů dolarů, v Evropě a Spojených státech, které skončily úspěšně v termínu a stanoveném rozpočtu. Standish Group: Haze.

2. Z 90 miliard dolarů, které byly ve federálním rozpočtu v roce 2019 vyhrazeny na IT výdaje, je 80 % vyhrazeno na údržbu staršího softwaru (Agencies Need to Develop Modernization Plans for Critical Legacy Systems, GAO, červen 2019). V odkazovaném textu se píše, že nevhodně udržovaný starší software vede k bezpečnostním problémům, neplnění úkolů, personálním potížím a růstu nákladů.

3. User story má obvykle formu „jako [role uživatele] potřebuji [funkce], abych [důvod]". Například: „Jako sociální pracovník potřebuji mít poznámky k danému případu uložené na telefonu, abych se k nim dostal i v oblastech bez mobilního signálu." Veškerá technická práce pak má za úkol pokrýt nějakou user story.

4. Podle výzkumu serveru Stack Overflow z roku 2018 provedeného mezi 57 075 vývojáři používá agilní metody 85 % profesionálních softwarových vývojářů. Podobně studie společnosti HP z roku 2015 zjistila, že „drtivá většina oslovených organizací dnes primárně používá agilní metody vývoje". Popisovaný proces tedy není nijak výjimečný.

5. Další informace o DevOps například v článku amerického ministerstva obrany Is Your Development Environment Holding You Back? A DIB Guid for the Acquisition Community.

5. Viz případové studie Case Study of CMS Management of the Federal Marketplace (americké ministerstvo zdravotnictví) a The Spectacular Fall and Fix of HealthCare.gov (Harvard Business School).

6. Jako příklad poptávkového dokumentu nabízíme tuto poptávku amerického daňového soudu, která v sekci Deliverables and Performance Standards obsahuje i plán posuzování kvality.

7. Další informace o rozdílném přístupu agilního vývoje k provozu a údržbě systémů najdete například v článku Software maintenance is an anti-pattern na blogu 18F.

8. Scrum je jedna z konkrétních verzí agilního vývoje. Přebírá jeho základní charakteristiky (iterativní vývoj a pragmatický přístup k procesům) a přidává k nim několik jasně definovaných definic, procesů a rolí, z nichž jednou je právě zmíněný scrum master – člověk, který má stručně řečeno za úkol odstraňovat procesní překážky bránící týmu v efektivní práci. (pozn. překl.)

9. Společnost Standish Group ve svém CHAOS Reportu z roku 2014 na vzorku 25 tisíc softwarových projektů zjistila, že software dražší než 10 milionů dolarů uspěje pouze v osmi procentech případů. S klesajícími náklady jde úspěšnost projektů rychle nahoru, kolem hranice jednoho milionu dolarů už se pohybuje kolem 70 %.

10. Společnost Standish Group ve svém CHAOS Reportu z roku 2014 na vzorku 25 tisíc softwarových projektů zjistila, že čím je softwarový projekt dražší, tím menší má šanci uspět. Omezením rozsahu jednotlivých zakázek rozdělíte projekt na větší počet menších komponent, které mají jednotlivě větší šanci uspět – a s nimi pak i projekt jako celek.

11. Aljašské ministerstvo zdravotnictví a sociálních věcí se s hledáním vhodných dodavatelů potýkalo v roce 2017. O tom, jak se jim nakonec podařilo přitáhnout malého místního agilního dodavatele, si můžete přečíst v článku How Alaska is using transparency to attract modern software vendors.

12. Úřad pracovních statistik nabízí přehled mezd softwarových vývojářů podle státu, podle kterého je mezi nejdražším státem (Washington) a nejlevnějším státem (Portoriko) rozdíl 150 %. I v rámci jednotlivých států se náklady mohou zásadně lišit, například mezi městem a venkovem. Takže trváním na práci dodavatele v místě zakázky můžete snadno zdvojnásobit své náklady.

13. Podrobný návod na distribuovanou spolupráci viz 18F's best practices for making distributed teams work.

Příručka řízení státních IT projektů
Robin Carnahan, Randy Hart & Waldo Jaquith

z anglického originálu pro Česko.Digital přeložil Tomáš Znamenáček

github.com/cesko-digital/derisking-handbook

verze 1.0.4, leden 2020