Open source

Open source jako koncept transparentní tvorby vývoje softwaru není ničím novým. Základem je myšlenka, že by měl být kód softwaru, který je někomu nabízen či prodáván, otevřený a pochopitelný pro další editaci. To je důležité jak pro vývojáře, který může získávat cenné podněty od komunity, nové funkce či možnosti pro svůj nástroje, tak také pro společnosti, které mají možnost si daný software upravit a přizpůsobovat dle vlastních potřeb, klesá riziko ukončení vývoje určité služby hlavním vývojářem atp.

Základní knihou, která popisuje výhody a specifika vývoje aplikací formou otevřeného kódu je dílo Erica S. Raymondse The Cathedral and the bazaar, které bylo publikováno již v roce 1999. Kniha pracuje se dvěma základními koncepty toho, jak lze vyvíjet software. První cestou jsou proprietární řešení, kdy za celým dílem stojí jasně definovaná organizace, která se snaží vytvořit co možná nejlepší produkt. Vydávány jsou verze jen stabilní, které mohou posloužit koncovým uživatelům a přinášejí něco skutečně nového. Periodicita je nižší, testování probíhá obvykle opět v nějaké jasně definované uzavřené komunitě.

Oproti tomu otevřený software zpravidla vychází z myšlenky, že programátor řeší svůj problém a s velkou pravděpodobností není sám. Nabízí své řešení k obecné diskusi a doufá, že se k němu připojí další – ať již přidáním funkcí, testováním nebo třeba zcela jiným pohledem na řešený problém. Primární vývojový směr je tak vývojář vývojáři. Verze mohou přicházet velice rychle za sebou, kód testuje více lidí, obsahuje proto menší množství chyb atp. Tento princi je podobný organicky vznikajícímu tržišti. To samozřejmě nevylučuje možnost označit určité verze jako stabilní, nebo za svoji práci požadovat odměnu. Otevřený kód je záležitostí transparentnosti a komunity, nikoli primárně ceny.

Z hlediska formální lze říci, že open source znamená dostupnost zdrojového kódu a splnění určitých licenčních podmínek tak, aby bylo možné s ním dále (i když třeba s určitými omezeními) manipulovat. Současně lze říci, že se celý fenomén stále více přesouvá do dalších oblastí lidského života – existují open source filmy, romány, hovoří se o otevřených datech nebo o zveřejňování vědeckých výsledků, které budou dostupné pro všechny.

Soudobá technická praxe

Otevřená řešení se stávají stále populárnější nejen pro vývojáře, ale především také pro konzumenty těchto aplikací. V následujícím se pokusíme ukázat některé různé licenční podmínky, které se do něj promítají, stejně jako základní trendy a možnosti vývoje nebo oblasti, kde se podobná řešení efektivně uplatňují.

Z hlediska licence nestačí, aby programátor dal k dispozici veškerý zdrojový kód, ale musí splňovat několik dalších podmínek. Nabyvatel licence musí být oprávněn počítačový program dále úplatně i bezúplatně distribuovat, rozšiřovat jej či rozmnožovat a současně musí mít k dispozici stejná práva ke zdrojovému kódu. Ten musí být napsán tak, aby bylo možné jej pochopit a dále upravovat. Zakázány jsou tedy techniky, které vedou k záměrné nesrozumitelnosti kódu či šíření částečně přeložených souborů, do kterých by bylo možné zasahovat jen s velkými obtížemi.

Dalším fundamentálním požadavkem je absence diskriminačních opatření. Není tedy možné říci, že pro určitou skupinu uživatelů či formy využití je daný software otevřený a pro jiné ne. Z tohoto hlediska lze dovodit, že vývojáři nepřísluší hodnotit chování koncových uživatelů na úrovni licenčních ujednání.

Další podmínky se pak týkají šíření softwaru – předně se musí licence vázat pouze na jednu aplikaci a není možné pomocí licencí vytvářet vazby mezi různými programy.[1] Odvozený software musí mít stejnou licenci jako ten zdrojový. Tímto je pak chráněna samotná možná existence open source řešení. Mezi nejznámější příklady konkrétních licencí patří GNU GPL, GNU LGPL, MIT license, Apache License 2.0, Eclipse Public License atp. Obecně lze říci, že jich existují desítky a liší se v různých detailech, které není vhodné brát na lehkou váhu.

Zajímavou otázkou v tomto kontextu může být, jak lze na podobném programovém vybavení vydělat. Možností je samozřejmě více a vždy záleží na druhu vyvíjeného softwaru. První variantou, která je zřejmě nejčastější, je prodej doplňkových služeb – může jít o technickou podporu, pronájem cloudu nebo další zpoplatněné funkce, které se nacházejí mimo samotný nástroj. Lze ale také model financování založit na dobrovolném dárcovství, reklamě nebo třeba prodeji obsahu, což je dnes asi cesta nejpopulárnější.

Je třeba zdůraznit, že na vývoji open source nástrojů lze vydělávat, ač častý stereotypní názor hlásá opak. Příkladem může být společnost Red Hat, která má roční obrat přes jednu miliardu dolarů, a jen za třetí čtvrtletí roku 2012 vykázala čistý zisk ve výši 38,2 milionů dolarů (asi 0,19 dolaru na akcii), což představuje 46% meziroční nárůst. Společnost vyvíjí serverový operační systém a další enterprice řešení.

Častým zdrojem financování je také institucionární podpora či prostý nadšenecký pokyn, které nemají od počátku žádný ekonomický potenciál. Můžeme se setkat s modelem konzultační tvorby, kdy pro jednu nebo více organizací je vyvíjena nějaká aplikace formou otevřeného kódu. Vychází se obvykle z myšlenky, že samostatný vývoj otevřeného řešení je rychlejší, levnější a lze jej snáze odladit, což je pro zadavatele důležité kritérium.

Častým modelem financování je také dvoukolejní vývoj, kdy programátor v jednu verzi aplikace nabízí jako bezplatnou a otevřenou a druhou s funkcemi navíc jako placenou. Příkladem může být opět společnost Red Hat, která testuje nové funkce a vlastnosti na distribuci Fedora a zapracovává je do placené firemní distribuce Red Hat Enterprise Linux, která mimo funkcí navíc disponuje uživatelskou podporou a dalšími benefity.

Open source není ale jen otázkou módní vlny či bezplatného používání, ale také modelu vývoje, který se obvykle značně liší od katedrálního modelu, na který jsou vývojáři zvyklí. Představa řízeného vývoje, s jasnou strukturou a organizací, kterou zná většina vývojářů v případě otevřeného modelu většinou úplně nefunguje. Vývojář se stává postupem času spíše mentorem, koordinátorem a lídrem celé skupiny vývojářů, který musí udržet celý proces tvorby v chodu. Je třeba, aby komunikoval s dalšími programátory, grafiky či testery, staral se o dobrou implementaci funkcí či marketing. Z programátora se tak stále více stává manažer, který řídí tým, kterému obvykle nic neplatí.

---

[1] Například nelze říci, že kancelářský balík bude open source jen tehdy, když bude uživatel používat také konkrétní webový prohlížeč nebo si koupí určitou doplňkovou službu.

Na druhou stranu je častý také hybridní model, který je vidět u větších projektů či firem jako je Firefox či Red Hat, které disponují určitým množstvím klasických vývojářů, ke kterým se připojují další vývojáři z komunity. Náročné části kódu na vývoj, matematický popis a týmovou spolupráci jsou obvykle v takovém případě řešeny interně a dílčí moduly a přidané funkce s testováním zajišťuje komunita.

Tento aspekt struktury firem, respektive modelu vývoje mezi katedrálou a tržištěm je velice důležitý a je vždy vhodné uvážit, co konkrétně pro daný projekt je či není potřeba. Vhodně zvolená architektura umožní podstatně lepší vývoj a nabídne kvalitnější výsledky.

Pro otevřený vývoj je vhodné používat dostatečně robustní webové prostředí. Dříve běžně užívaná webová stránka a e-mailová konference se jeví jako nepříliš efektivní a tak se zcela přirozeně objevují nástroje, které umožní lepší týmovou spolupráci, sledování změn a dohled nad celým projektem. Zřejmě nejznámějším a největším nástrojem tohoto typu je SourceForge, který nabízí nejen možnost ukládat aktuální verzi programu, ale podporuje pokročilé nástroje pro verzování, diskusní fórum, systém na sledování změn a řadu dalších funkcí.

Dalším podobným nástrojem je Google Code, který nabízí mimo jiné také robustní API či Launchpad. Je vždy třeba zvážit, jak robustní nástroje se bude pro vývoj daného produktu hodit a zda je vhodné využít API, které by vývoj mohlo značně ulehčit. Online programovací nástroje, které složí například pro vývoj aplikací v AJAX, jsou z tohoto hlediska také velice praktické, protože umožňují okamžitou online spolupráci, ladění a diskusi nad problémem.

Open source umožňuje dobré přizpůsobení aplikace potřebám zákazníka – není problém implementovat jeho vlastní moduly nebo mu na míru danou aplikaci upravit. Ostatně jde o jeden z častých modelů vývoje, kdy nové funkce a modul základní (core) aplikace vznikají na přání klientů a jsou pak dále nabízena uživatelům.

Právě tato možnost přizpůsobení a do určité míry ztráta závislosti na dodavateli aplikaci zvyšuje atraktivitu produktu a při dodržení norem umožňuje snadný transfer dat mezi jednotlivými zařízeními a programy. To je také důvod, proč nejrůznější úřady všude ve světě přecházejí hromadně na open source řešení. Nejde jen o úsporu, ale především o nezávislost, možnost přizpůsobení si systému dle vlastních potřeb a modularitu.

Lze bez nadsázky říci, že open source se stává módou, která může znamenat významnou konkurenční výhodu. Právě móda je jedním z nejdůležitějších parametrů, které lze v oblasti ICT sledovat – podléhá mu mnohem více než svět oblečení či hraček. Jde o skutečnost, kterou by měl mít každý vývojář v oblasti open source na paměti, když se pouští do vývoje libovolného produktu.

Open source programování má v obecné rovině často silně modulární charakter, což může být jak pozitivní, z hlediska nákladů a rychlosti programování, tak také svazující, pokud jde o dlouhodobou udržitelnost a rozvoj. Platí totiž pravidlo, že většinu problémů již před námi někdo řešil a výsledek jeho snažení si můžeme, díky svobodným licencím, snadno importovat do vlastního díla. Velký kus práce tak za nás udělají druzí a programování ve světě open source je často přirovnáváno k puzzle – stačí najít vhodné kousky a dobře je pospojovat. Skutečně nové práce je často nutné odvést relativně málo.

Na tento princip tvorby by vývojáři v případě open source technologií měli pamatovat již v počátcích tvorby aplikací – na jedné straně proto, aby si ušetřili práci s tvorbou algoritmů či celých knihoven, které již někdo vytvořil, ale také proto, aby byli schopni jednotlivé balíčky efektivně do aplikace začleňovat. Školní poučky o tom, jak by měl vypadat dobrý programátorský postup zde získává zcela jasný význam. Modularizace je nutná pro efektivní práci s již hotovými výtvory, poznámkování pro pochopení algoritmů a jejich sdílení s dalšími vývojáři, efektivita kódu je při modulární výstavbě kritickou komponentou atp.

Silné stránky a možnosti

Některé silné stránky jsme již naznačili. Obecně je možné je rozdělit do dvou velkých skupin. První jsou výhody pro vývojáře a druhé pro koncové uživatele. Přitom je třeba říci, že výhody pro koncového uživatele jsou z hlediska vývoje softwaru také důležité, protože by měly být součástí marketingové strategie a celkové komunikace se zákazníky.

Pokud jde o výhody pro programátora, tak prvním významným momentem je možnost používat cizí otevřené kódy a řešení. Nejde o prosté kradení nápadů či postupů druhých, ale o skutečnost, že řada funkcí se v programech opakuje. Příkladem mohou být algoritmy pro hledání jednoslovných předložek, které potřebuje každý textový editor, aby nezůstávaly na koncích řádků nebo třeba slovníky pro kontrolu pravopisu. Podobných příkladů lze najít desítky. To co by jedna společnost vyvíjela měsíce či roky je často k dispozici v hotové, odzkoušené formě.

Tato skutečnost umožňuje v relativně velice krátkém čase vybudovat i značně robustní aplikace s velkým množstvím funkcí a nástrojů, které jsou umně poslepovány dohromady. Současně tyto možnosti vedou vývojáře k tomu, aby kód tvořili přehledně a pochopitelně, právě s ohledem na možný import cizích komponent. Dodržování pravidel programování je tak významným vedlejším efektem, který může v konečném důsledku přinášet velké finanční úspory.

S rychlostí vývoje souvisí také možnost zapojení komunity. Pokud je pro ni projekt zajímavý a vývojář s ní umí pracovat, lze zásadním způsobem akcelerovat vývoj softwaru i v oblastech, které ještě nejsou hotové. Řada lidí se připojí k projektu proto, že dělají dobrou či zajímavou věc, ale také mohou být motivováni skutečností, že daný problém budou někdy potřebovat vyřešit také. Týmová spolupráce je proto levnější a rychlejší také pro ně samotné.

Ač jedním z nejčastěji udávaných slabých míst otevřeného softwaru bývá považována bezpečnost, zkušenost ukazuje, že je situace jiná. Díky průhlednosti kódu je chybovost proprietárního i otevřeného softwaru nejhůře stejná. Pokud tedy nedochází k vývoji aplikací s krizovou důležitostí z hlediska bezpečnosti, jako je vojenský či bankovní software, lze říci, že bezpečnost patří mezi klady.

Určitou výhodou může být také internacionalizace projektů. Díky otevřenosti kódu a propojeností velkých nástrojů na organisaci vývoje, dochází často k tvorbě mezinárodních vývojových komunit. Prostorová vzdálenost nepředstavuje žádný problém a propojení odborníků z různých částí světa může přinášet řadu benefitů. Tím nejdůležitějším je, že v mezinárodním prostředí je obvykle větší šance na nalezení osob, které mají stejný problém. Tým pro vývoj se tak vytváří snáze, než v menších centrech. Mezi další výhody pak patří různorodost přístupů k řešení problému, jeden z důležitých aspektů efektivního vývoje, odlišnost v dílčích potřebách a požadavcích na aplikaci, což může pomoci ke komplexnímu řešení nebo snadná lokalizace softwaru do národních jazyků zainteresovaných vývojářů.

Nezanedbatelné jsou také výhody pro koncové zákazníky. Předně je to snížení závislosti na jednom dodavateli. Do velké míry může ve vývoji či úpravách systému pokračovat kdokoli jiný, aniž by bylo třeba hradit náklady na vývoj již hotových komponent. Jde o relativně důležitou skutečnost, kterou není možné marginalizovat. S tím souvisí také možný transport dat mezi aplikacemi. Díky jasné struktuře a otevřenosti lze snadno vyměnit jednu aplikaci za druhou, což v případě proprietárního softwaru není běžně možné.

Pro uživatele je důležité také to, že si může aplikaci kdykoli upravit, rozšířit či pozměnit podle vlastního přání a potřeb. Je relativně obvyklé, že komerční firmy financují open source vývoj komponent do systému, který potřebují a používají. Díky této možnosti úpravy softwaru podle aktuálních potřeb lze zásadním způsobem zvyšovat efektivitu. Nejčastějším a nejhlasitějším argumentem je pak cena samotného softwaru, která je nulová a platí se jen za přidané služby či data.

Slabé stránky a možné problémy

Jedním z nejvíce diskutovaných problémů či nedostatků v oblasti vývoje open source aplikací je otázka ceny. Pokud máme aplikaci, jejíž vývoj stále nemalé množství času a peněz, může snadno dojít k tomu, že ji převezme někdo jiný a bude nabízet za nižší cenu, neboť přijde o nutnost počátečních investic, nebo sám produkt zlepší, takže bude konkurenceschopnější, nežli původní program. Na to zcela přirozeně navazuje problém, kolik je vlastně možné si za takový software účtovat a jaké další služby k němu musí být připojené, aby měl vůbec šanci na funkční obchodní model.

Takto vytvořená expozice problému jednoznačně identifikuje jednu z možných obav. Je ale třeba říci, že svět softwaru se stále více posouvá do oblasti služeb. Uživatelé jsou stále více zvyklí platit za obsah nebo servis, než za samotný bázový produkt. Příkladem může být úspěšně fungující operační systém Android, který je bezplatný, ale společnosti Google generuje nemalý zisk především za prodej aplikací, hudby, filmů a knih v integrovaném obchodu s aplikacemi. Podobným příkladem by mohl být třeba Amazon a řada dalších firem.

Bez zdarma dodaného softwaru by nebylo možné prodávat další služby či obsah, které mají velkou přidanou hodnotu. S trochou nadsázky tento posun představuje podobnou diferenci, jaká je mezi výrobcem válcovaného plechu a automobilu. Marže i přidaná hodnota automobilky je přirozeně mnohem vyšší.

Podobným častým řešením jsou například otevřené cloudové aplikace, které získávají prostředky především z pronájmu výkonu svých serverů. Samotný software je zdarma, může si jej nainstalovat každý, ale provoz vlastního serveru není laciný, je nutná správa, konfigurace. Proto řada subjektů volí pronájem, na kterém vývojářské společnost vydělává, aniž by ztratila výhody vývoje open source aplikací. Ač to na první pohled může působit podivně, právě takové modely patří a zřejmě i budou patřit mezi nejúspěšnější. Rizikem může být nezvládnutí marketingu či nastavení finanční náročnosti.

Druhým velkým rizikem může být do značné míry upravená role vývojáře, který nejen programuje, ale musí se starat také o to, aby jeho projekt fungoval po všech dalších stránkách. Komunikace s dalšími lidmi si zabere nemálo času, takže řada těch, kteří s programováním v open source začínali, dnes již nepíše téměř vůbec, a přesunula se do role manažerů.

Nejde přitom jen o problémy související se ztrátou oblíbené činnosti, ale především o nároky na dovednosti a znalosti, které většina programátorů nemá. To co se učí manažeři pět let ve škole, je

třeba zvládnout nějak rychleji a intuitivněji. V takovém případě se většina lidí neubrání řadě chyb, které mohou poškozovat celý programátorský záměr. Organizace virtuálních týmu, dobrá komunikace s testery i vývojáři, nutnost motivovat všechny, kdo se na vývoji podílejí, může při nedobrém osobnostním profilu programátora či absence manažerských dovedností nakonec celý vývoj paradoxně brzdit a zpomalovat. Při názorových rozporech pak často dochází k rozpadům či štěpení týmů.

Problémem jsou také licence. Ač v zásadě sledují velice podobné ideály, v drobnostech se liší a nejsou obecně příliš kompatibilní. Nelze ani říci, že by existoval nějaký jednoznačný žebříček, který by je řadil podle striktnosti, takže v případě, že potřebuje programátor kombinovat řešení různých osob, může se dostat do složitých licenčních problémů. Také volba vlastní licence se tak ukazuje nikoli jako marginální záležitost, ale jako to, co může mít velký dopad na praktický vývoj. Licencí jsou přitom desítky a není možné se v nich vždy snadno zorientovat. Samozřejmostí by měla být znalost těch základních, ale v případě exotických jako jsou ISC License (ISC) Multics License (Multics) či NASA Open Source Agreement 1.3 (NASA-1.3) je velice složité vyhledat a právně zajistit vzájemnou kompatibilitu. Konzultace s právními odborníky je v takovém případě zcela nezbytná a často nákladná, což opět může pozdržet vývoj či zveřejnění nové verze systému.

Nezanedbatelným rizikem je také možnost, že nedojde k vytvoření komunity, která by se na vývoji a testování softwaru podílela. V takovém případě padá velká část výhod, které s sebou toto vývojářské paradigma přináší.

Aktuální projekty

Open source stal masivním hnutím, ke kterému se dnes připojují stovky či tisíce vývojářů a aplikací. Z těch nejznámějších lze vyzvednout například dvojici operačních systémů Linux a Android. Linux je dnes nejrozšířenějším operačním systémem pro servery, ale hojně se užívá také ve vestavěných zařízeních či chytrých televizích. Společné pro všechny distribuce je jádro, nad kterým pak běží grafické prostředí (například Gnome, KDE atp.), ovládaní a ovladače či aplikace. Řada uživatelů pak má také Linux nainstalovaný na svých pracovních stanicích.

Android nabízí zcela jiný pohled na open source vývoj. Stojí za ním Google, který jej organizuje, financuje, hledá pro něj koncová zařízení, stará se o marketing atp. Přínos komunity je zde spíše menší, ale otevřenost usnadňuje vývoj a zlepšuje obraz celého projektu. Oproti rozdrobeným linuxovým distribucím je zde jedna jasná linie, kterou garantuje jedna společnost.

Dalším známým příkladem je OpenOffice.org což je kancelářský balík, který má konkurovat MS Office. Poté, co byla společnost Sun koupena Oracle, došlo k postupné stagnaci projektu a jeho fragmentaci. Dnes je vývoj zase obnoven díky odštěpení části vývojářů a přispěvatelů k projektu LibreOffice. Jde o pěknou ukázku toho, kdy svoboda licence a fragmentace může projekt zachránit.

Známé je také prostřední pro e-learning Moodle, které patří mezi nejpoužívanější a nejoblíbenější na světě. Zdrojem financí je pronájem samotného názvu produktu pro školící a vzdělávací aktivity. Servis, který chce názvu Moodle využívat, musí platit. Díky masovému rozšíření jde opět o velice zajímavý a životaschopný model financování.

Příkladem z druhé strany spektra může být grafický editor GIMP, který měl konkurovat Photoshopu. Ač jde o funkční, robustní a relativně známý nástroj, potýká se s neustálými problémy – velice malá

vývojářská komunita, pomalý vývoj, špatné grafické rozhraní, absence obchodního modelu atp. Ani dobrá myšlenka a nesmírně kvalitní produkt tak nemusí nutně znamenat úspěch.