Szerződés sz.	Munka csomag	Szállítás	Szállítás dátuma
030776	WP1	D1.5	2007-08-31

CASCADOSS

Nemzetközi vízeséssszerű program fejesztése környezeti alkalmazásokhoz használható nyílt forráskódú térinformatikai és távérzékelési szoftverekhre

EGYEDI TÁMOGATÁSI INTÉZKEDÉS

1.2.4.2.2-ES PRIORITÁS: ÚJ MÓDSZEREK FELTÁRÁSA A TRANSZNACIONÁLIS TECHNOLÓGIA TRANSZFER NÉPSZERŰSÍTÉSÉRE ÉS TÁMOGATÁSÁRA

NYFSZ üzleti modellek gyűjteménye és elemzése

A projekt kezdő időpontja: 2007. 01. 05.

Időtartam: 24 hónap

A dokumentumra vonatkozó információk

Projektre vonatkozó információk

Dokumentum jellege *	R
Vezető résztvevő	SADL / KULeuven
Becsült ember hónap	2
Terjesztési szint *	CO
Szállítás dátuma projekt hónap	4

***Jelleg: R = Jelentés; P = Prototípus; D = Bemutató; O = Egyéb;**
Biztonsági szint: PU = Nyilvános; PP = A program más résztvevőire korlátozott (beleértve a Bizottsági Szolgálatokat; RE = A konzorcium által meghatározott csoportra korlátozott, pl. a Projekt Érdekcsoport (beleértve a Bizottsági Szolgálatokat); CO = Bizalmas, csak a konzorciumi tagok számára (beleértve a Bizottsági Szolgálatokat).

Dublin Core Metaadatok

Cím	Nyílt forráskódú üzleti modellek gyűjteménye és elemzése
Azonosító	D1.5_hu_Business_Models.odt
Szerző	Karel Maesen (KULeuven)
Társszerző(k)	
Kiadó	Cascadoss konzorcium
Kiadás dátuma	2009. 01. 19.
Téma	Üzleti modellek
Formátum	Nyitott dokumentum formátum
Nyelv	Magyar

Verzió történet

Verzió	Dátum	Szerző(k)	Leírás
1.0	2007. 03. 12.	Karel Maesen	Első verzió

Vonatkozó dokumentumok

Tartalomjegyzék

1 Bevezetés

Ez a jelentés a szabad/nyílt forráskódú szoftver - Free/Open Source Software (F/OSS) – (továbbiakban NYFSZ) üzleti modelljeinek áttekintését és elemzését mutatja be. Az elemzés arra a kérdésre összpontosít, hogy a vállalkozások számára hogyan teremt érdekes üzleti lehetőségeket a nyílt forráskód. A NYFSZ sajátossága, hogy a forráskódja nyílt és szabadon hozzáférhető. Annak ellenére, hogy a NYFSZ előállítók következésképpen nem realizálhatnak jövedelmet a licencekből, a konstrukció előnyöket biztosít mind a szoftver előállítók, mind pedig a fogyasztók számára, és így új üzleti lehetőségeket hoz létre. Ez a jelentés azt vizsgálja meg, hogy mik ezek a lehetőségek, ezek hogyan viszonyulnak a NYFSZ jellemzőihez, különös hangsúlyt fektetve Európa térinformatikai közösségeire.

A jelentés elsődleges célja, hogy a NYFSZ-ról információkat nyújtson, oktatási anyagként szolgáljon; nem pedig a meglévő tudás továbbfejlesztése. Ez tehát egy kísérlet a vonatkozó tudományos, műszaki és üzleti irodalom szintézisére. Az összefoglalás koherens áttekintést kíván adni arra vonatkozóan, hogy a nyílt forráskód mit jelent a vállalkozások számára.

2 Mi a NYFSZ?

2.1 A NYFSZ definíciója

A Szabad Szoftver Alapítvány - Free Software Foundation (FSF) – a szabad szoftvert tömören a következőképp definiálja: „A felhasználó a szoftvert szabadon futtathatja, másolhatja, terjesztheti, tanulmányozhatja, megváltoztathatja és továbbfejlesztheti” (http://www.fsf.org/licensing/essays/free-sw.html). A Nyílt Forráskód Kezdeményezés - Open Source Initiative (OSI) – a nyílt forrás definícióját a http://www.opensource.org/docs/osd veboldalon teszi közzé. Az OSI meghatározása a legszélesebb körben használatos definíció a Nyílt Forráskódra vonatkozóan, és a szoftverfejlesztő közösség erős támogatását élvezi. A mi céljaink szempontjából az OSI és az FSF definíciója közötti eltérés elenyésző. Mindkét definíció megegyezik azokban az alapvető jogokban, amelyeket a szoftver készítőjének biztosítania kell az engedélyesek (felhasználók) számára ahhoz, hogy a termék NYFSZ-nek minősüljön. Rosen a következők szerint sorolja fel ezeket a jogokat (Rosen 2004):

1. Az engedélyesek bármilyen célra szabadon használhatják a nyílt forráskódú szoftvereket.

2. Az engedélyesek anélkül másolhatják és terjeszthetik szabadon a nyílt forráskódú szoftvereket, hogy szabadalmi díjat kellene fizetniük az engedélyező számára.

3. Az engedélyesek anélkül készíthetik és terjeszthetik szabadon a nyílt forráskódú szoftverek származékos termékeit, hogy szabadalmi díjat kellene fizetniük az engedélyező számára.

4. Az engedélyesek szabadon férhetnek hozzá a nyílt forráskódú szoftverek forráskódjához és használhatják azokat.

5. Az engedélyesek szabadon kombinálhatják a nyílt forráskódú és egyéb szoftvereket.

Rosen felsorolása összeegyeztethető mind az FSF, mind pedig az OSI NYFSZ definíciójával.

Az NYFSZ sajátos jellemzői, mint fejlesztési és terjesztési modell, továbbá mint üzleti modellek alapja, éppen azon a tényen alapulnak, hogy a szoftver létrehozója megadja ezeket a jogokat a felhasználóknak.

2.2 A NYFSZ és a szellemi tulajdonjog

A NYFSZ nem közkincs szoftver. A közkincs szoftver senkinek sem a tulajdona, következésképpen senki sem vezethet be korlátozásokat arra vonatkozóan, hogy hogyan és mire használják. A legtöbb NYFSZ viszont nem közkincs szoftver, hanem magánszemélyek vagy cégek tulajdonát képezik. A szerzői jogon keresztül a szerzők megtehetik, hogy kötelezettségeket fogalmazzanak meg a szoftver engedélyeseire vonatkozóan, és jellemzően élnek is ezzel a lehetőséggel. Vannak korlátai annak, hogy egy NYFSZ forráskóddal mit tehetünk. Például a legtöbb licenc tiltja a kód egyes részeinek egyszerű, forrásmegjelölés nélküli kivágását és beillesztését (copy-paste).

A NYFSZ jellegzetes tulajdonságai abból a tényből adódnak, hogy a szoftverrel kapcsolatos szabadságjogok a szerzői jogok feletti licencként kerülnek átadásra. A NYFSZ licenc széleskörű felhasználási és újraelosztási jogokat biztosít az engedélyeseknek. Ezek a jogok olyan széleskörűek, hogy a tulajdonjoghoz váltak hasonlatossá. Az engedélyes valójában nem birtokolja a forráskód szerzői jogát, és nem rendelkezik azzal, csak jogában áll használni, módosítani vagy terjeszteni azt – ezek olyan jogok, melyek hagyományosan kizárólag a tulajdonos számára vannak fenntartva (Scott 2006). Ugyanilyen fontos azonban, hogy a NYFSZ licencek bizonyos jogi kötőerővel bíró követelményeket fogalmazhatnak meg. Ezek a követelmények biztosítják a NYFSZ fejlesztők számára munkájuk intellektuális integritásának

védelmét (pl.: forrásmegjelölések), a “copyleft” lehetővé tétele, és védelmet nyújtanak a hozzájárulók munkájának kisajátítása ellen.

2.3 A NYFSZ és a kölcsönös licencek

Egyes NYFSZ licencek azt a követelményt támasztják, hogy a NYFSZ szoftvert használó vagy az azon alapuló szoftvereket szintén NYFSZ licenc alatt kell terjeszteni. Ez az egyik legfontosabb jellemzője a GPL licencnek – ami az egyik legnépszerűbb nyílt forráskód licenc. Az ezen jellemzővel bíró NYFSZ licencek a copyleft (Stallman 1984), erős (Scott 2006) vagy kölcsönös (Rosen 2004) licenc elnevezést viselik. Ahogy Richard Stallman megfogalmazza, „a *copyleft általános módja egy program vagy egyéb munka szabaddá tételének, miközben kötelezővé tesszük, hogy a program minden módosított és kiterjesztett verziója is szabad legyen*” (Stallman 1984, kiemelés tőlünk). A GPL (GNU General Public License), az MPL (Mozilla Public License) és a CPL (Common Public License) közkedvelt kölcsönös licencek.

Az ezt a jellemzőt nélkülöző NYFSZ licenceket gyenge (Scott 2006) vagy akadémiai licenceknek (Rosen 2004) nevezik. A BSD, az Apache vagy az Artistic jó példák az akadémiai licencekre.

Az akadémiai és a kölcsönös licencek közötti különbség tükrözi az NYFSZ értékéről alkotott véleményekben megnyilvánuló alapvető eltéréseket. Az akadémiai licencek szószólói, mint például a BSD vagy az Apache közösségek, a szoftverből köztulajdont akarnak létrehozni. Más fejlesztők ezt a szoftver köztulajdont forrásként használhatják, de nem elvárás, hogy cserében valamilyen hozzájárulást tegyenek a köztulajdon javára. Az akadémiai licencek lehetővé teszik, hogy a köztulajdonból származó szoftvert szellemi tulajdont képező szoftver előállítására használják. A kölcsönös licencek azonban előírják, hogy akik a köztulajdonban lévő szoftverből valamilyen munkát származtatnak, az így nyert munkát helyezzék is vissza köztulajdonba.

A Lesser GPL (LGPL) Licenc érdekes köztes helyet foglal el a kölcsönös és az akadémiai licencek között. A licenc maga kölcsönös abban az értelemben, hogy a forráskód bármely módosítását közre kell adni, amennyiben a módosított szoftver terjesztésre kerül. Az LGPL így garantálja, hogy a szoftver továbbfejlesztései nyílt forráskódú köztulajdonban maradjanak. Viszont ha a szoftvert egy nagyobb rendszer részeként vagy könyvtárként használják, a licenc semmilyen kikötést sem tartalmaz arra vonatkozóan, hogy erre a nagyobb rendszerre milyen licenc vonatkozik. Az LGPL kölcsönös licencként funkcionál, amikor a szoftvert kiterjesztik, vagy módosítják; és akadémiai licencként funkcionál, amikor a szoftvert egy nagyobb rendszer összetevőjeként használják.

2.4 A NYFSZ mint szoftverfejlesztési modell

A 2.1 pont alatt felsorolt szoftver szabadsági jogok nagymértékben befolyásolják a szoftverfejlesztés mikéntjét. A következőkben a NYFSZ fejlesztési modell számos olyan jellemzőjét tárgyaljuk, melyek annyira hatékonnyá teszik.

2.4.1 A közösség által működtetett fejlesztések

Mivel a forráskód mindenki számára nyitott, nincsen a felhasználókat és a fejlesztőket aszerint elkülönítő határvonal, hogy kinek van hozzáférése a szoftver forráskódjához és a megvalósítás részleteihez, és ki az, aki előtt ezek az információk titkosak. Ennek az elkülönítésnek a hiánya a felhasználói közösséget feljogosítja arra, hogy a szoftverfejlesztési folyamatban aktívabb szerepet vállaljon. Ezért egy NYFSZ projekt közössége jellemzően a teljes spektrumot felöleli, a hétköznapi felhasználóktól kezdve az ún. hozzájáruló felhasználókig (akik a folyamathoz teszteléssel, visszajelzéssel, hibajelentésekkel és dokumentációval járulnak hozzá), egészen a fejlesztőkig (akik a forráskódot írják).

Mivel a NYFSZ projekt körüli közösség ilyen jogosítványokkal rendelkezik, a vezető fejlesztői csoport idővel szervesen megváltozhat. A felhasználók és/vagy más fejlesztők különféle okokból akarhatnak aktívan részt venni magában a kódírásban (hogy az általuk kedvelt jellemzők megvalósuljanak, hogy az őket zavaró hibákat eltávolítsák, vagy egyszerűen csak hogy valami érdekes időtöltést találjanak vasárnap délutánra). Ha úgy ítélik meg, hogy a NYFSZ projekt licence kellően védi saját érdekeiket, lehet, hogy önként együttműködnek a kezdeti fejlesztővel vagy fejlesztőkkel. (Természetesen az eredeti fejlesztőktől függ, hogy befogadják-e őket. A legtöbb érett NYFSZ projektben létezik egyfajta folyamat, amin az önkénteseknek végig kell menniük, mielőtt társfejlesztőként elfogadnák őket).

Ha a vezető fejlesztői csoport hozzájárulókat és fejlesztőket akar vonzani, akkor nem csak a forráskódnak (termék), hanem a fejlesztésnek (folyamat) is nyitottnak és átláthatónak kell lennie (Fogel 2006, pp.36ff). Ez azt jelenti, hogy a legtöbb NYFSZ projekt sokkal több vonatkozásban nyitott, mint a forráskód: tervezési dokumentumok, tervezési és megvalósítási kérdésekre vonatkozó műszaki megbeszélések, stb. is nyilvánosak és szabadon hozzáférhetőek mind a felhasználók, mind a fejlesztők számára.

Nagy hatású *A katedrális és a bazár* (*The Cathedral and the Bazaar)* című esszéjében Eric Raymond számos, sikeres NYFSZ projektekből leszűrt következtetést sorol fel. Ezek közül több közvetlenül összekapcsolható olyan közösség létezésével, amely felhatalmazást nyert a fejlesztési folyamatban való aktív részvételre (Raymond 1998). A következtetések a következők:

“Minden jó szoftvermunka a fejlesztő személyes viszketésének vakargatásával kezdődik”

Sok NYFSZ projekt a felhasználó-fejlesztő egy meglévő problémájából indul ki. Ezeket a projekteket ezért nem a piac vagy a technológia működteti, hanem a felhasználó: egy olyan valódi felhasználói igényből indulnak ki, amelyet nem lehet a szoftverpiacon létező ajánlatokból megfelelően kielégíteni (a felhasználó számára elfogadható áron). Ilyen körülmények között a felhasználó-fejlesztők dönthetnek úgy, hogy egy NYFSZ projektet kezdeményeznek. Az ilyen eredetű NYFSZ projektek gyakran sokkal jobban a felhasználói közösség valós igényeire szabottak, mint számos üzleti versenytárs.

„A jó ötletek után mindjárt a legjobb dolog a felhasználók jó ötleteinek felismerése. Időnként az utóbbi még jobb.”

A felhasználók/fejlesztők, akik egy adott NYFSZ projekt iránt érdeklődnek, saját tapasztalataikból származó továbbfejlesztési ötletekkel járulhatnak hozzá a munkához. Pl. a GIS világa számos tapasztalt és jól tájékozott felhasználót számlál, akiknek a meglátásai értékesek lennének a GIS rendszerek fejlesztésének irányításában. A szoftverfejlesztést befolyásolni tudják azáltal, hogy e-mail listákon és vitafórumokon keresztül tudásukkal hozzájárulnak a szoftverfejlesztéshez. Sőt, mivel a forráskód is a rendelkezésükre áll és sok fejlesztési információ is azonnal elérhető, arról is meggyőződhetnek, hogy milyen mértékig vették figyelembe a fejlesztők az ő hozzájárulásukat. Ez a fajta visszaigazolás nagyon ösztönzően és motiválóan hat a felhasználókra, de nagyon nehéz megvalósítani zárt forráskódú fejlesztési modellekben.

„A felhasználók társfejlesztőkként való kezelése a legkevésbé rázós út a gyors kódfejlesztés és a hatékony hibaelhárítás felé”

„Ha elég nagy béta-tesztelő és társfejlesztő bázisod van, szinte minden probléma gyorsan arcot ölt, a megoldás pedig magától értetődő valaki számára.” vagy („Elég sok szempár tükrében minden hiba jelentéktelen”).

Mindkét következtetés abból a tényből ered, hogy a felhasználók közül néhányan (még ha csak a kisebbség is) annyira jól informálttá válik a szoftver részleteivel kapcsolatban, hogy elkezd a kódfejlesztéshez hozzájárulni és a projekttel kapcsolatos problémákat megoldani. A felhasználók, akik megfelelően motiváltak egy új jellemző kifejlesztésére, ezt meg is tehetik, a kódot pedig átadhatják a fejlesztőknek. Ha a fejlesztők úgy találják, hogy ez a kódbázis számára értékes kiegészítés, be fogják építeni. Ezzel a módszerrel a szoftver olyan

jellemzőket kaphat, melyekre nem is számítottak a projektfejlesztők (és a terveikben sem szerepeltek). Nagyon gyakran ez a jellemző olyan valami, amit a felhasználó saját használatára fejlesztett és úgy dönt, hogy a közösség rendelkezésére bocsátja azt. Még fontosabb és még gyakoribb azoknak a felhasználóknak az esete, akiknek a szoftver használata során problémákat és hibákat kell vizsgálniuk. Mivel hozzáférésük van a forráskódhoz, a műszakilag kifinomultabb felhasználók ezt gyakran arra használják fel, hogy megvizsgálják és beazonosítsák a problémát. Ez sokkal specifikusabb probléma leírásokhoz és megoldási javaslatokhoz vezet.

A sikeres NYFSZ projektet körülvevő közösség természetesen nem homogén. És nem minden felhasználó megfelelően alkalmas vagy érdeklődő ahhoz, hogy aktív szerepet vállaljon egy NYFSZ projektben. Mockus et al. (2000) tanulmányozta az Apache vebszerver fejlesztési folyamatot és a közösség által megvalósított hozzájárulást. Az Apache szerverrel kapcsolatos következtetések a következők:

- 10-15 vezető fejlesztő írta meg a kódnak több mint 80%-át.
- A vezetőknél egy nagyságrenddel nagyobb csoport javítja ki a hibákat, és még egy nagyságrenddel nagyobb csoport tesz hibajelentéseket.

Hogy az adatokat összefüggéseikben lássuk: az írás időpontjában Mockus et al. körülbelül félmillióra becsülte az Apache Szerver installációk számát. A hibajelentéseket küldő magánszemélyek teljes száma hozzávetőleg 3000 volt. Ez az installációk számának kevesebb, mint 1%-a.

Ezek az eredmények és mások által végzett kutatások azt mutatják, hogy a NYFSZ projektet körülvevő közösség nagymértékben tagozódik (Gosh és Prakash 2000; Gosh et al. 2000; Hunt és Johnsson 2002, Healy és Shussman 2003). A vezető fejlesztők kis csoportját körülveszi a hozzájárulók/tesztelők nagyobb csoportja. A második szinten aktív felhasználók helyezkednek el, akik hibajelentéseket küldenek, valamint fórumokon és levelező listákon keresztül választ adnak a felhasználók kérdéseire. Vitathatatlanul a legnagyobb csoportot azok a felhasználók alkotják, akik a szoftver passzív fogyasztói maradnak. A térinformatikai projektek terén - mint pl. a JTS, a PostGIS, a GeoTools és a GRASS - szerzett saját tapasztalataink szintén alátámasztják ezeket a megállapításokat.

Amint a fentiek is mutatják, a teljes felhasználói közösségnek csak egy meglehetősen kis szegmense az, amelyik aktív szerepet vállal a projektben. Mindazonáltal, számszerűen ez még mindig sokkal több, mint ahányan jellemzően egy velük összemérhető szellemi tulajdont képező szoftver fejlesztésében részt vesznek. Ugyanilyen fontos az a tény, hogy a hozzájárulók különféle háttérrel rendelkeznek, mind műszaki, mind kulturális értelemben. Kombinálva a teljesítményelvűséggel, amely gyakran magától értetődő a NYFSZ fejlesztések esetében (lásd alább) ez előny, mert az ötleteknek és megvalósításoknak túl kell élniük a különféle szemponton alapuló vizsgálatokat és egyenrangú szemlézéseket. És végül a legtöbb hozzájáruló és a fejlesztői mag önmagát választja ki a fejlesztési feladatokra, képességei, tudása és érdeklődése alapján. Pl. a GeoTools és a PostGIS esetében úgy tűnik, hogy a legtöbb hozzájáruló jó képességű szoftverfejlesztő, akiket szakmai okokból érdekel a GIS. Ez a motiváció és elkötelezettség magas szintjét eredményezi.

2.4.2 Nyitott fejlesztés és információ sűrűség

A NYFSZ fejlesztési modell egyik fontos előnye, hogy lehetővé teszi a külső hozzájárulást (az eredeti vezető fejlesztői csoporton kívülről érkező hozzájárulás). De ehhez, ahogyan már korábban rámutattunk, teljesen átlátható, jól dokumentált, nyitott fejlesztési folyamatra van szükség. Ennek az oka kézenfekvővé válik, ha figyelembe vesszük, mivel jár egy projektben az első hozzájárulás átadása:

1. A hozzájárulónak fel kell állítania egy megfelelő összeállítási környezetet.
2. Létre kell hoznia, majd tesztelnie kell saját hozzájárulását.

3. Meg kell győznie az eredeti fejlesztői csoportot, hogy ellenőrizzék a munkáját, fogadják el és vegyék fel a kódbázisba.

Ha a fejlesztési folyamat nem nyitott vagy átlátható, a potenciális hozzájárulóknak az 1. és 3. lépés során jelentős nehézségekkel kell szembenézniük. Először is, az összeállítási környezet felállítása elég összetett feladat lehet. Ha a fejlesztési folyamat nem jól dokumentált, a megfelelő összeállítási környezet felállítására vonatkozó információ jellemzően elavult vagy hiányos lehet (az eredeti fejlesztőknek már meg van a saját összeállítási környezetük és nincs szükségük erre az információra). A 3. lépés még ennél is problémásabb. Abban az esetben, ha a fejlesztési folyamat nem nyitott, a potenciális hozzájáruló számára nem garantált, hogy a hozzájárulását elfogadják. Ezt okozhatja az, hogy a hozzájárulás nem illik az útitervhez, nem felel meg valamelyik, a fejlesztő által nem ismert irányelvnek, vagy egyszerűen, hogy az eredeti fejlesztők számára ez a munka érdektelen. A 3. lépésnél jelentkező problémák a legkárosabbak, mert rövid idő alatt aláássák a projekt hitelét, mint valóban nyitott, a közösség által működtetett projekt.

Az olyan projektek, amelyek nem kellőképpen nyitottak, el fogják veszíteni a külső hozzájárulásokat, mivel a potenciális hozzájárulók azt fogják érezni, hogy kevés esély van a hozzájárulásuk elfogadtatására, legalább valamelyest elfogadható időn belül és energiaráfordítással. Tehát ha egy projekt külső hozzájárulókat és új fejlesztőket akar vonzani, akkor legalább az alábbiaknak eleget kell tennie:

1. A kódfejlesztési folyamatokat világosan dokumentálni kell (ki kell építeni az eszközöket, tesztelést, kódolási stílust, stb...).

2. Érthetően tegyük közzé a projekt célkitűzéseit, vízióját, fejlesztési filozófiáját és útitervét.

3. Közösen vitassuk meg az útitervet, a projekt jelenlegi helyzetét és az eddigi előrehaladást.

Megfordítva is igaz: azon projektek, melyeknek teljesen átlátható, nyitott a fejlesztési folyamata, jó a projekt dokumentációja is. Minden, amire egy újonnan csatlakozónak szüksége van ahhoz, hogy elkezdje a fejlesztést, és hogy megértse a projekt céljait és irányvonalát, már mind jól dokumentált és könnyen elérhető. Zárt forráskódú projektekben a dokumentáltság ilyen szintjét esetenként nehéz elérni.

Noha azt láthatjuk, hogy a valóban nyitott NYFSZ projektek általában meglehetősen jól dokumentáltak fejlesztési szempontból, a dokumentáció mégsem egy dokumentum készletből áll. Ehelyett inkább elszórtan lelhető fel levelezési listákban, wiki oldalakon, a kibocsátást nyomon követő rendszerekben és még a forráskódban is.

2.4.3 „Elágaztathatóság”

A NYFSZ által biztosított jogok egyik velejárója a projekt elágaztatására vonatkozó szabadság. A fejlesztők egy csoportja elágaztathatja a projektet a kód lemásolásával és a másolaton alapuló konkurens projekt elindításával. Közismert példák az Emacs elágaztatása az Emacs és Xemacs projektekre, a XFree86 elágazgatása a XFree86 és a X.org-ra. A GIS világban példa erre az eredeti Jump projekt elágaztatása az OpenJump felé.

Az elágazgatást általában „rossz dolognak” tartják: a munka megkettőzését jelenti, a fejlesztői és felhasználói közösségeket megosztja, és a NYFSZ közösségen belül belharcokat eredményezhet. Az elágazgatás általában terheket is ró a NYFSZ felhasználókra. El kell dönteniük, hogy az eredeti projektben maradnak, amit komolyan gátolhat a közösség egy részének elpártolása, vagy inkább átállnak az elágaztatott verzióba, ami az erőforrás és az érdeklődés hiánya miatt kihalhat. Ezért a NYFSZ elágaztathatósága üzleti kockázatokat jelent a felhasználók számára.

Mindazonáltal az elágaztathatóság kockázata a NYFSZ projektvezetők számára egyben fontos ösztönzést is jelent arra, hogy fogékonyak legyenek felhasználóik, hozzájárulóik és fejlesztőik szempontjaira. Ha ez nem így van, a közösség tagjai az elágaztatás mellett dönthetnek, és új projektet kezdhetnek. Ahogy Fogel (2006, p.88) fogalmaz: „A reprodukálhatóság maga után vonja az elágaztathatóságot; az elágaztathatóság pedig maga után vonja a konszenzust." Így az elágaztathatóság garanciaként működik, csakúgy, mint a vásárlói elégedettség az üzleti életben.

Amennyiben az eredeti projekt kölcsönös licenc alapján működik, az elágaztatás is viszonylag „jóindulatú". Az eredeti licencet át kell vinni az új licencre, és így az elágazás minden változtatása és továbbfejlesztése rendelkezésére áll az eredetinek is. Ez segít biztosítani azt, hogy az elágaztatás csak nagyon alapos és súlyos indokok miatt történik meg, nem pusztán üzleti vagy önös érdekek miatt.

2.4.4 Érdemeken alapuló fejlesztés

A projektmenedzsment és a vezetés tekintetében fontos különbség van a nyitott és a zárt szoftverfejlesztési folyamat között. Ahogyan már korábban jeleztük az elágaztathatóság tárgyalása során, a NYFSZ projekt vezetője nem rendelkezik abszolút ellenőrzéssel a projekt fölött. Fel kell állítaniuk egy projektirányítási struktúrát, amelyhez jellemzően hozzátartozik a tevékenység részletes leírása, a konfliktuskezelési eljárás, valamint az az eljárás, melynek során az új fejlesztőket felvehetik a vezető fejlesztő csoportjába (olyan fejlesztők, akik módosításokat menthetnek el a projekt adattárában). Ahhoz, hogy megvalósítható és fenntartható legyen, a NYFSZ projekteknek olyan irányítási struktúrát kell létrehozniuk, amely igazságos és motiváló azok számára, akik a legtöbb energiát fektetik bele. Ez azt jelenti, hogy a legtöbb NYFSZ projektvezetés az érdemeken alapul: azoknak, akik értékes hozzájárulásokat tesznek, beleszólásuk van a projekt irányába, a hozzájárulás nagyságrendjével arányosan. Mondanunk sem kell, hogy az ilyen, érdemeken alapuló vezetés nagyon motiváló a jó fejlesztők számára. Arra is jó garancia, hogy a műszaki kitűnőség és/vagy a végfelhasználók elvárásai ösztönözzék a fejlesztést, nem pedig az üzleti megfontolások.

2.5 A NYFSZ mint szoftverterjesztési modell

A NYFSZ szoftverterjesztési modell is (Olsen 2006). A NYFSZ a szoftvercsomag lehető legszélesebb körű terjesztését teszi lehetővé, nagyon alacsony költséggel és minimális marketinggel. Valójában vannak olyan cégek, akik csak emiatt választják a NYFSZ üzleti modellt, a NYFSZ fejlesztési modellt viszont nem veszik át (vagy csak egy minimális fokig).

Az alábbiakban felsorolunk néhány okot, amiért a NYFSZ-t jellemzően széles körben terjesztik:

- Mivel minden NYFSZ licenc engedélyezi a terjesztést forrás vagy bináris formátumban, a szoftvert a NYFSZ projekt tulajdonosaitól eltérő ügynökök is terjeszthetik. Ennek legszélesebb körben ismert példája a Linux rendszermag. A Linux részéről ez a nagy piaci részesedés a szerver piacon valószínűleg nagyrészt annak a ténynek köszönhető, hogy cégek és szervezetek széles skálája végezte a terjesztést (Debian, Slackware, Mandrake, Red Hat, SUSE, Gentoo, IBM, ...).

- Mivel a NYFSZ szoftvereket le lehet tölteni, és tesztelni lehet költségek nélkül, időkorlátoktól mentesen, és anélkül, hogy e-mail-en vagy telefonon kapcsolatba kellene lépnünk az eladóval, a leendő felhasználók számára sokkal könnyebb tesztelni és megismerni a szoftvert.

- Mivel a forráskód nyílt, a leendő felhasználók jobban tudják mérni a szoftver minőségét. Ez különösképpen igaz, amikor a szoftver nyílt forráskódú fejlesztési modell alapján kerül kifejlesztésre. Például a nyitott NYFSZ fejlesztési modellekben a hibalisták mindenki számára szabadon megtekinthetőek, ami lehetővé teszi, hogy a leendő felhasználók láthassák, mennyi hiba van, azok milyen gyorsan és mennyire professzionálisan kerülnek kiküszöbölésre. Ez a nyitottság táplálja a szoftver iránti

bizalmat. Még akkor is, ha a vizsgálat eredménye vegyes, a leendő felhasználónak pontos képe van arról, hogy mit kap. A szellemi tulajdont képező rendszereknél általában nem ez a helyzet, ahol csak akkor tudjuk meg, mit kapunk, amikor már sok pénzt és időt szántunk rá.

- Sok fejlesztő szeret a NYFSZ-rel dolgozni szoftverfejlesztés közben az alacsony költségek miatt, – még akkor is, ha a kifejlesztett szoftver szellemi tulajdont képező szoftveren fog futni. Ez gyakran átcsap a termelési környezetre is: miután megfelelő bizalmat szereztek ezek iránt a NYFSZ összetevők iránt, ezeket az összetevőket fogják javasolni gyártási felhasználásra is. (Pl. a Java vebalkalmazásokat gyakran fejlesztik a NYFSZ Tomcat alkalmazás szerver felhasználásával, majd pedig gyártásba BEA WebLogic vagy IBM WebSphere alkalmazás szervereken küldik).

- Amikor a szoftverfejlesztőknek könyvtárat vagy egy összetevőt kell választaniuk, gyakran a NYFSZ-t részesítik előnyben, mert az nem teljesen fekete doboz. A forráskód rendelkezésre állása azt jelenti, hogy a hibákat és problémákat nyomon tudják követni a könyvtár vagy az összetevő forráskódjáig (pl. egy hibakereső programban).

2.6 A NYFSZ mint üzleti modell

Amikor a NYFSZ nélkülözhetetlen egy üzleti modellhez, akkor nyílt forráskódú üzleti modellről beszélünk. Ha megvizsgáljuk a szoftver értékláncot (lásd 4.2.2 pont), megértjük, hogy a NYFSZ licenc hatékonyan akadályozza meg, hogy egy fejlesztő közvetlenül kisajátítsa szoftverfejlesztési munkája értékét. Az értéklánc egyéb jövedelemgeneráló tevékenységeire, mint például a képzés, támogatás és tanácsadás, ez nincs hatással. Ebben az értelemben a NYFSZ üzleti modellek nem jelentenek maradéktalan továbblépést a hagyományos modellektől (lásd még Young (1999) és Behlendorf (1999) a NYFSZ és a hagyományos szoftver üzleti modellek közti hasonlóságokról).

Mindazonáltal a sikeres szoftver üzleti modell kulcsának hagyományosan a szoftvertermékből történő közvetlen pénznyerést tekintik. A szoftverfejlesztést a magas állandó költségek és gyakorlatilag nulla határköltségek jellemzik. Zárt forráskódú modellben a vállalatok a licencekből tehetnek szert jövedelmekre, és így nagyon magas megtérülést generálnak, feltéve, hogy a vásárlói bázis elég nagy ahhoz, hogy az állandó költségek megtérüljenek. Így foglalja össze Olsen (2006) a licencekből származó jövedelmek jelentőségét:

> A szoftver licenc jövedelmek jó jövedelmek – mivel egy szoftverből újabb másolatot készíteni és azt licenccel ellátni gyakorlatilag további költségek nélkül lehet, a vállalkozások és a pénzügyi piacok szeretik a licencből származó jövedelmeket.

A NYFSZ üzleti modellekkel kapcsolatban az a kérdés merül fel, hogy vajon megfelelően kompenzálható-e a licenc jövedelem kiesése. A jelentés további része ezzel a kérdéssel foglalkozik.

3 A NYFSZ gazdaságtana

A NYFSZ viszonylag új módja a szoftver-előállításnak. Ebben a fejezetben gazdasági alapjait fogjuk röviden tárgyalni. Olyan tagadhatatlan érveket sorolunk fel, melyek alapján elhihetjük, hogy a NYFSZ módszerű szoftver-előállítás fenntartható és kézzelfogható gazdasági előnyöket nyújt. Véleményünk szerint a téma kifejtése megteremti a szükséges hátteret a NYFSZ üzleti modellek megértéséhez. A gazdasági tudományos szakirodalomban a NYFSZ számos rejtvényt ad fel. Elsőként a *motivációs rejtvényt* vizsgáljuk meg. Miért dolgoznak fejlesztők olyan szoftvereken, melyek értékét nem tudják a maguk számára kisajátítani? A NYFSZ licenc séma a szoftvert nem versengővé teszi, a fogyasztásból pedig senkit sem lehet kizárni. Ez valójában azt jelenti, hogy a NYFSZ-t a közjavak közé sorolhatjuk. Ez a következő kérdéshez vezet: *Magán szereplők miért működnek együtt közjavak előállításában, és hogyan birkóznak meg a potyautasokkal*?

3.1 A motivációs rejtvény

A motivációs rejtvény a következő: A NYFSZ fejlesztők miért fordítanak olyan sok időt és energiát olyan termékre, amelyet lényegében ingyenesen osztogatnak el? A hagyományos gazdasági szempontokat figyelembe véve ez a viselkedés csak akkor tekinthető racionálisnak, ha a belőle származó előnyök meghaladják a fejlesztő költségeit. A kérdést akkor úgy kell feltennünk, hogy mik azok az előnyök, amelyekkel az egyéni fejlesztők számolhatnak. Lakhani és Wolf (2005) kétfajta motivációt különböztet meg: belső és külső motiváció. A motiváció belső, amikor magától a tevékenységtől elválaszthatatlan. A fejlesztőt magának a tevékenységnek az aspektusai motiválják. A külső motiváció a tevékenységnek elválasztható következménye. Lakhani és Wolf a belső motivációkat tovább osztják az élvezeten alapuló és a kötelezettségen/közösségen alapuló motivációkra. Ami a külső motivációkat illeti, átveszik Lernes és Tirole (2002) azon javaslatát, mely szerint a külső előnyök közvetlen és késleltetett megtérülésűek lehetnek. A közvetlen előnyök közé tartoznak a következők:

- Egy vállalat azért fizet, hogy NYFSZ projekteken dolgozzunk (pl. amikor a Red Hat vagy az IBM mérnökei a Linux-on dolgoznak)
- Közvetlen felhasználói és így kedvezményezettjei vagyunk a NYFSZ-nek, melyen a fejlesztő dolgozik

A késleltetett előnyök közé tartoznak a következők:

- Karrier előmenetel (jelzés a munkaerő piacon)
- Készségek javítása az együttműködés és az egyenrangú szemlézés által

Lakhani és Wolf a sourceforge.net-en hosztolt NYFSZ projektek egyedi fejlesztőit vizsgálta. Vizsgálatuk 2001-ben kezdődött. A legfontosabb okok, amit a fejlesztők részvételük magyarázataként említenek, hogy szükségük van a szoftvertermékre, amin dolgoznak (58%), és a NYFSZ fejlesztés által nyújtott intellektuális ösztönzés (45%). A harmadik legfontosabb motiváló tényező a lehetőség saját szoftverfejlesztő képességeik tökéletesítése. Az a meggyőződés, hogy a szoftvereknek nyitottnak kellene lenniük, szintén fontos közösségi alapú motiváció (33%). Ezen eredmények fényében, Lakhani és Wolf arra a következtetésre jut, hogy „nem létezik egyetlen, domináns magyarázat arra, hogy egy egyéni szoftverfejlesztő miért dönt az NYFSZ projektben való részvétel és hozzájárulás mellett” (p.19).

Amikor a szerzők azon tényezőket vizsgálták, amelyek meghatározzák, hogy a fejlesztő milyen energiákat hajlandó a NYFSZ–ra fordítani, azt találták, hogy csak az alábbiak számítanak jelentős tényezőnek (fontossági sorrendben):

- Élvezeten alapuló belső motivációk, a kreativitás érzésének formájában
- Külső motivációk fizetség formájában

- Kötelezettségen/közösségen alapuló belső motivációk.

A belső motivációk empirikus kutatása arra mutatott rá, hogy általánosságban a külső előnyöknek, mint például ha fizetnek valakinek, negatív hatásuk van a belső motivációkra. Lakhani és Wolf meglepő módon arra a következtetésre jut, hogy ez a negatív kapcsolat a NYFSZ esetében nem áll fenn. A részvételükért megfizetett fejlesztők nagy csoportja esetében (a minta 40%-a) az a tény, hogy fizetik őket, nem csökkenti a kreativitás érzetét és/vagy a közösségi szerepet.

3.2 Közjavak előállítása magáncégek által

A 2001-es Lakhani-Wolf tanulmányban a NYFSZ fejlesztők körülbelül 40%-a fizetést kapott a hozzájárulása fejében. Figyelembe véve olyan cégek azóta növekvő részvételét a NYFSZ fejlődésében, mint például a Red Hat, az IBM és a Novell, biztonsággal feltételezhetjük, hogy ezen csoport relatív mérete az elmúlt néhány év során jelentősen nőtt. Ez mutat rá következő kérdésünkre: *Magáncégek miért működnek közre közjavak létrehozásában, és hogyan birkóznak meg a potyázás problémájával*?

A vállalatok vagy úgy vesznek részt a NYFSZ-ben, hogy fejlesztői erőforrásokat biztosítanak a közösségi alapú projekt számára, vagy pedig maguk indítanak el NYFSZ projekteket. Az első esetre példa a Red Hat, amely jelentősen hozzájárul a GCC-hez (GNU Compiler Collection – a GNU/Linux legfontosabb fordítóprogram készlete). A második esetre jó példa a Maemo. A Maemo egy Linux alapú NYFSZ fejlesztési platform a Nokia tablet PC-hez. A Maemo-t a Nokia hozta létre azzal a nem titkolt szándékkal, hogy a Nokia Tablet számára egy jó minőségű alkalmazások kifejlesztésére képes fejlesztői közösséget támogasson.

A cégeket erre javarészt külső motivációk késztetik. Vagy érdekes üzleti lehetőséget látnak a NYFSZ-ben és átvesznek egy NYFSZ alapú üzleti modellt, vagy költséghatékonysági okok miatt, esetleg taktikai és stratégiai okokból teszik ezt. Az első csoportra példa a Red Hat vagy a Novell, akik NYFSZ alapú üzleti modelleket vettek át. Ezeket a modelleket bővebben az 5. fejezetben tárgyaljuk. A második csoportot a rendszer integrátorok jellemzik. A rendszer integrátorok azért kedvelik a NYFSZ-t, mert a segítségével nyereségüket növelhetik, a közvetlen költségek csökkentése révén. A harmadik csoportra, akik stratégiai vagy taktikai okokból tesznek hozzájárulásokat, a fentebb említett Maemo projekt a példa.

Újabb keletű kutatások arra engednek következtetni, hogy a NYFSZ-t használó cégek között is van néhány, akiket szintén *belső* motiváció késztet arra, hogy erőforrásaikból a NYFSZ-re áldozzanak (Rossi és Bonaccorsi, 2006). Ebben az összefüggésben a belső motivációt a NYFSZ közösség értékeivel való azonosulás jelenti. Ezek a cégek azért támogatják a NYFSZ-t, mert kötelességüknek érzik, hogy a közösségnek is visszajuttassanak valamit. Ahogy várható is, Rossi és Bonaccorsi arra a következtetésre jutott, hogy ez a cégek egy tiszteletreméltó kisebbségére igaz (35%). A szerzők meggyőződése, hogy ezek a vállalatok „közösségi hozzáállásukat valószínűleg azon alapítóiktól örökölték, akik korábban magánemberként részt vállaltak [NYFSZ] programozásban és szenvedélyüket hivatássá emelték.” (p.105).
Ezen belsőleg motivált cégek létezése ellenére világos, hogy a legtöbb vállalat opportunista és önző okokból foglalkozik a NYFSZ-rel. Sok megfigyelő tart attól, hogy ezek a vállalatok „többet fognak kivenni, mint amennyit betesznek”. Pontosabban attól tartanak, hogy a vállalatok megpróbálják a lehető legtöbb értéket kisajátítani a NYFSZ projektből (pl. kiegészítő szolgáltatások nyújtása révén), miközben a lehető legkevesebb erőforrást szentelik a projektre. Ez a „potyautas promléma” esete. A Wikipedia ezt így írja le: “potyautasok azok a szereplők, akik az erőforrásokból többet fogyasztanak, mint a rájuk eső méltányos rész, vagy kevesebb támogatást nyújtanak, mint az előállítás költségéből méltányosan rájuk eső rész. A potyautas probléma azt a kérdést veti fel, hogyan lehet a potyázás előfordulását megelőzni, vagy legalább negatív hatásait korlátozni.”
(forrás: http://en.wikipedia.org/wiki/Free_rider_problem).

Általánosságban a potyautas magatartást azért tartjuk ártalmasnak, mert aláássa azon szereplők motivációját, akik a terhekből a maguk méltányos részét viselik. A legújabb kutatások azonban azt mutatják, hogy a NYFSZ viszonylag immúnis a potyautas

magatartással szemben. Rossi és Bonaccorsi, a cégek NYFSZ-hez való hozzájárulását vizsgáló tanulmányában arra a következtetésre jut, hogy „nyílt forráskódú közösségek bizonyos tagoknak megengedik, hogy sokkal többet vegyenek el, mint amennyit adnak, feltéve, hogy a legalapvetőbb tagsági szabályokat nem sértik meg. ... A [közbirtokosságra], a közjavak biztosítására és a potyautasságra vonatkozó szakirodalom valószínűleg eltúlozta a kisszámú nem-közreműködők esetleges ártalmas szerepét, feltételezve, hogy az őáltaluk mutatott viselkedés elkerülhetetlenül elharapózik. Ez nem feltétlenül igaz" (Rossi és Bonaccorsi, 2004).

Úgy véljük, hogy a NYFSZ-nek a potyázással szembeni viszonylagos immunitása a NYFSZ számos jellemzőjéből adódik. Az első és talán legfontosabb az, hogy a NYFSZ egy harmadikféle alternatívát kínál sok vállalat számára a hagyományos „csináld-vagy-vedd meg" dilemmára (Bessen 2006). A nagyon speciális igényekkel rendelkező vállalatok együtt tudnak működni más vállalatokkal és egyéni fejlesztőkkel olyan „platformon", amit azután könnyen testre lehet szabni saját speciális céljaiknak megfelelően. Ennek a legkiemelkedőbb példája a Linux. A Linux-ot használják mobiltelefonok operációs rendszeréhez (Motorolla), szórakoztató elektronikához (TiVO), és használják számítógép berendezés forgalmazók is (IBM). A Linux modularitása teszi lehetővé a felhasználások eme széles skáláját. Erős és stabil „platformot" biztosít, ami kiindulópontként szolgál az ezen cégek által igényelt testreszabott fejlesztésekhez. Mivel az együttműködésben részt vevő vállalatok hasznát látják a jó minőségű alapterméknek, motiváltak az együttműködésben azért, hogy biztosítsák az alaptermék folyamatos rendelkezésre állását és minőségét. A potyázás nem opció ezen vállalatok számára, mivel rendkívül speciális igényeik vannak: nem valószínű, hogy bármely más vállalat vagy magánszemélyek egy csoportja pontosan azt a terméket állítaná elő, amire nekik szükségük van. Baldwin és Clark (2003) szintén úgy érvel, hogy a NYFSZ projektekben való potyázás korlátok közé szorul, ha a termék erősen moduláris jellegű. A „moduláris" kifejezést a szerzők úgy értik, hogy a szoftver alaptermékhez könnyen lehet egyedi jellemzőket hozzáadni, más projekt résztvevőktől függetlenül és inkrementálisan. A nagyon széles potenciális jellemző-készlettel bíró, erősen moduláris szoftverek csökkentik a további jellemzők kifejlesztésének költségeit, miközben a potyautas viselkedést nem vonzóvá teszik: mivel a potenciális jellemzők hatalmas készletéből egy viszonylag kis fejlesztői csoport is ki tudja válogatni a megvalósítandókat, a potyázók nemigen számíthatnak arra, hogy a fejlesztők a potyautas számára fontos jellemzőt valósítják meg.

Azt a tételt, hogy a modularitás korlátozza a potyázás által okozott károkat, általában a szoftverek és különösen a NYFSZ két további jellegzetessége is megerősíti. Az egyik az, hogy a „platform" vagy „infrastruktúra" kód valójában nem feltétlenül nagyon erőforrás-igényes. Például egy audio interjúban Mark Shuttleworth, a népszerű Ubuntu Linux forgalmazás multimilliomos alapítója azt mondta, hogy „szükség esetén évtizedeken keresztül" tudta volna finanszírozni az Ubunti projektet saját zsebből. Hasonló a helyzet az Apache vebszerverrel is. Mockus, Fielding és Herbsleb az Apache projektről szóló tanulmányában arra a következtetésre jutott, hogy mindössze 15 vezető fejlesztő 80%-ban járult hozzá a forráskód előállításához (2000). Ezek a viszonylag kis csapatok képesek a platformot megfelelő állapotban tartani, a többiek pedig kis, inkrementális jellemzőkkel járulnak hozzá a munkához, amik együttesen növelik a teljes termék értékét. Ez arra is enged következtetni, hogy a platformon való hozzájárulás költsége sok vállalat számára megfelelően alacsony a teljes termék értékéhez viszonyítva, ezért a legtöbb vállalat nem fog potyázni: az esetleges kedvezőtlen sajtó és a rossz közösségi kapcsolatok egyszerűen nem érik meg a potyázásból származó költségmegtakarítást.

A másik jellemző, ami ellenállóvá teszi a NYFSZ-t a potyázás okozta károkkal szemben az, hogy a szoftvereket általában erős és kedvező hálózati hatások jellemzik. Egy adott szoftver értéke nő, ahogyan a szoftver felhasználóinak száma növekszik. A NYFSZ esetében még a potyázó vállalatok is jó hatással vannak a szoftver megértésére és használatára. A termék népszerűségének növekedése az egyéni fejlesztők megnövekedett motivációját vonja maga után (belső okokból, mint például az eredmény feletti büszkeség, de külső okokból is, így jelezve például az egyén programozói képességeit a szélesebb közönség felé). Így a potyázás negatív hatásait csökkentik a kedvező hálózati hatások.

Amennyiben a NYFSZ-re kölcsönös licenc vonatkozik, mint pl. a GPL, maga a licenc is csökkenti a potyázás okozta károkat. Noha a potyázók valaki más munkájának a gyümölcsét élvezik, nem vehetik azt ki a közösből, és nem alakíthatják üzleti termékké. Így a potyautasok nem tudnak túl sok kárt okozni, mert nem tudnak az eredetivel versengő terméket létrehozni.

Az utolsó ok, ami miatt a potyázás nem annyira káros a NYFSZ esetében, az, hogy a tudás nem kézzelfogható vagyon. Azon vállalatoknak és egyéni fejlesztőknek, akik hozzájárulnak a NYFSZ-hez, olyan mélységű tudásuk van a termékről, amivel a potyázók nem rendelkeznek. Noha elméletben minden elérhető és betekinthető mindenki számára, a gyakorlatban mély tudásra csak úgy lehet szert tenni, ha dolgozunk a terméken, ha elkezdjük fejleszteni azt. Valójában a NYFSZ szolgáltatások ügyfelei is tisztában vannak ezzel, és így inkább olyan vállalat szolgáltatásait veszik igénybe, amelyik maga is valóban hozzájárult a NYFSZ-hez, valamelyik potyázó versenytársa helyett.

Ezzel nem azt állítjuk, hogy a potyázás nem okoz gondokat a NYFSZ számára. A nagymértékű potyázás ronthatja egy NYFSZ projekt esélyét a sikerre. De úgy látszik, hogy itt a potyázás kisebb problémát jelent, mint más területeken.

4 A NYFSZ üzleti modellek leírása

4.1 Az üzleti modell összetevői

A téma kifejtésének céljaira az üzleti modellek struktúrájának egyszerűsített megfogalmazását vesszük át, ami Osterwalder (2004) és Stähler (2002) munkáin alapul. Ezen megfogalmazás szerint, az üzleti modell a következő összetevőkből áll (Osterwalder 2004, p. 31).

Összetevő	**Felteendő kérdések**
Értékelőny	Milyen értéket teremt a vállalat az ügyfelei és partnerei számára?
Termék/szolgáltatások	Mit értékesít a cég?
Az értékteremtés architektúrája	Hogyan és milyen konfiguráción keresztül teremtődik az érték? A vállalat hol helyezkedik el az értékláncban?
Jövedelem modell	A vállalat hogyan keres pénzt?

4.2 Az értékteremtés architektúrája

A NYFSZ üzleti modellek tárgyalása során hangsúlyozottan rávilágítunk a NYFSZ-nek az architektúra részeiben játszott megkülönböztető szerepére. Ezen összetevők Stähler's általi leírását a következő elemekre egyszerűsítjük le: a piacterv, az elhelyezkedés a szoftver értékláncban és a terjesztési modell (interfész a vállalat és ügyfelei között).

4.2.1 A piacterv

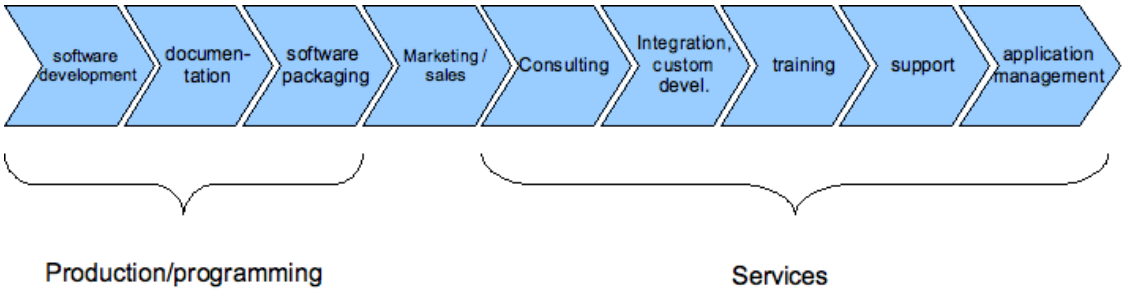
A piacterv azokat a piacokat határozza meg, amelyeket a vállalat az üzleti modellben megcéloz. Ezeket a piacokat a földrajzi elhelyezkedés, típus (vállalatok közötti: Business-to-Business vagy B2B, vállalatok és fogyasztók közötti: Business-to-Consumer vagy B2C) alapján lehet szegmensekre osztani, illetve aszerint, hogy magán- vagy közszféráról van szó. A következő felosztásokat javasoljuk:

- Földrajzi szegmentáció:
    - Helyi piac: az ügyfelek földrajzi értelemben közel találhatóak és/vagy a vállalattal közös kulturális vagy nyelvi hátérrel rendelkeznek.
    - Globális piac: az ügyfél földrajzi elhelyezkedése nagymértékben irreleváns az üzleti modellre nézve.
- Típus szegmentáció:
    - B2C – vállalatok és fogyasztók közötti: az üzleti modellben megcélzott fogyasztók az ajánlott termék vagy szolgálatás végfelhaszánlói
    - B2B – vállalatok közötti: a fogyasztók más vállalatok, amelyek a terméket vagy a szolgáltatást saját ajánlataik inputjaként használják.
    - B2G – vállatok és a kormány közötti: az ügyfél egy állami szférába tartozó szervezet.

- Szoftvertermék szegmentáció (Berlecon Kutatás 2002):
    - Vállalkozói megoldások: A modell olyan ügyfeleket céloz meg, akiket az üzleti eljárásaikat támogató megoldások érdekelnek (Ügyfélkapcsolat menedzsment, Vállalkozói erőforrás tervezés)
    - Csomagolt tömegpiaci szoftver: A modell olyan ügyfeleket céloz meg, akiket az olyan felhasználói szoftverek érdekelnek, amelyek nagyon általános informatikai szükségleteket elégítenek ki, mint például a szövegfeldolgozás, képfeldolgozás vagy táblázatkezelés.

4.2.2 A szoftver értéklánc

Az értéklánc azokat a lépéseket írja le, amelyek az inputokat hozzáadott értékkel bíró outputtá változtatják. A szoftver értéklánchoz a Bercelon Kutatás (2002) modelljét vesszük kiindulási pontnak.



Ebben az értékláncban a lépések a következők:

- Szoftverfejlesztés: a szoftver elemzése, tervezése, programozása és tesztelése.
- Dokumentáció: a dokumentáció megírása (API dokumentáció, hivatkozási kézikönyv, felhasználói útmutatók, oktató anyagok, segédanyagok, GYIK, ...)
- Szoftvercsomagolás: felhasználóbarát csomagolás készítése a szoftverhez; a szoftver párosítása egyéb csomagokkal.
- Marketing/értékesítés: a szoftver marketingje, az értékesítés lezárása, a széleskörű alkalmazás népszerűsítése, terjesztés.
- Tanácsadás: a szoftverre vonatkozó tanácsadás nyújtása.
- Integráció/személyre szabott fejlesztés: a szoftver integrálása az ügyfél rendszereibe, felhasználó-specifikus igényekre szabása
- Képzés: a szoftverre vonatkozó képzés felhasználás vagy a testreszabás közben
- Támogatás: a végfelhasználók támogatás (telefon, e-mail), installációs és aktualizálási támogatás, hibaelhárítás
- Alkalmazás menedzsment: az ügyfél szoftveren alapuló alkalmazásainak működési menedzsmentje.

Az üzleti modell tárgyalása során jelezni fogjuk, hogy az üzleti modellben az értéklánc melyik lépésén van a hangsúly. A hangsúlyt a gyártás/programozás vagy a szolgáltatás lépések fogják kapni. Azokat a (valós vagy potenciális) üzleti modelleket, amelyek a marketing és az értékesítés lépésekre koncentrálnak, nem vizsgáljuk. Noha lehetséges ilyen üzleti modelleket tervezni, meggyőződésünk szerint az ilyen üzleti modellek nem igazán függenek a NYFSZ fejlesztési és terjesztési modell speciális jellemzőitől.

4.2.3 A terjesztési modell

A terjesztési modell azt írja le, hogy a vállalat hogyan szándékozik ügyfeleit elérni.

5 NYFSZ üzleti modellek

A következő fejezetben idealizált „tiszta formában létező” üzleti modelleket mutatunk be. A legtöbb vállalat ezeket a tiszta üzleti modelleket keveri, hogy fenntartható üzleti modellt hozzanak létre. Számos vállalat pedig ezeket a modelleket a hagyományos zárt forráskódú üzleti modellekkel vegyíti.

5.1 Az üzleti modellek áttekintése

Az alábbi táblázat az általunk a piacon megfigyelt, valamint a szakirodalom által beazonosított modellek áttekintését adja.

Model name	References	Alternative names	description	Prototype companies
Dual Licensing	Olsen 2006, Daffara 2007, Koenig 2004	Twin licensing	Companies make their F/OSS available under a non-reciprocal license	Sleepy Cat, MySQL
Support Sellers	Hecker 1999, Daffara 2007, Koenig 2004	Product specialists, Software provider (GPL)	Companies that provide paid support for F/OSS products they developped.	MySQL, Alfresco, Talend, Compiere
Third-party Support Seller	Krishnamurthy 2005	Third-party service provider	Companies that provide paid support for products they do not themselves develop.	EnterpriseDB, SpikeSource
Platform providers	Daffara 2007, Krishnamurthy 2005	Distributor	Companies bundle several F/OSS products into a complete platform. The company guarantees the quality of the integrated platform.	Red Hat, Novell, SpikeSource, SourceLabs
Consulting	Daffara 2007		Companies that provide consultancy w.r.t. F/OSS products based on their knowledge or experience of working with F/OSS	
Software as a Service	Koenig 2004	Hosted strategy	F/OSS is used to provide access to revenue-generating online services	Google,Salesforce.com
Added-Value providers	Krishnamurthy 2005	Software provider (non-GPL)	Companies that create proprietary software derived from F/OSS.	Apple, EnterpriseDB
Accessorizing	Hecker 1999		Selling services or physical items related to F/OSS (books, hardware, ...).	O'Reilly, Manning, SourceForge, CollabWeb
Loss Leader	Hecker 1999, Daffara 2007	Split OSS/commercial products	no-charge F/OSS product is used as a loss leader for traditional commercial software	Xen Source, SugerCRM, JasperSoft
Widget Frosting	Koenig 2004, Hecker 1999	Optimization Strategy	Selling a combined offering of high-value soft- and/or hardware components together with low-cost F/OSS offerings. The low-cost F/OSS enable higher pricing for the high-value components	Oracle, IBM

5.2 A kettős licenc modell

A kettős licenc modell esetében a szoftvertermék két különféle licenc alatt érhető el:

- Egy kölcsönös nyílt forráskódú licenc, amely arra kötelezi a vevőket, hogy saját termékeiket szintén a kölcsönös licenc alatt bocsássák ki, amennyiben a termék saját szoftvertermékükben felhasználásra kerül.
- Egy üzleti licenc, amely felszabadítja a felhasználót azon kötelezettség alól, hogy kölcsönös licenc alatt bocsássa ki termékét.

Röviden: a vevő vagy vállalja a viszonosságot azzal, hogy hozzájárul a köztulajdonban lévő szoftverhez, vagy fizet a fejlesztőnek.

5.2.1 Értékelőny

Az ügyfél haszna az, hogy a saját ajánlatába beépítheti a NYFSZ-t, anélkül, hogy nyílt forráskódúvá kellene tennie azt.

5.2.2 Termék/szolgáltatások

A szoftvertermék bináris és/vagy forráskód.

5.2.3 Az értékteremtés architektúrája

5.2.3.1 A piacterv

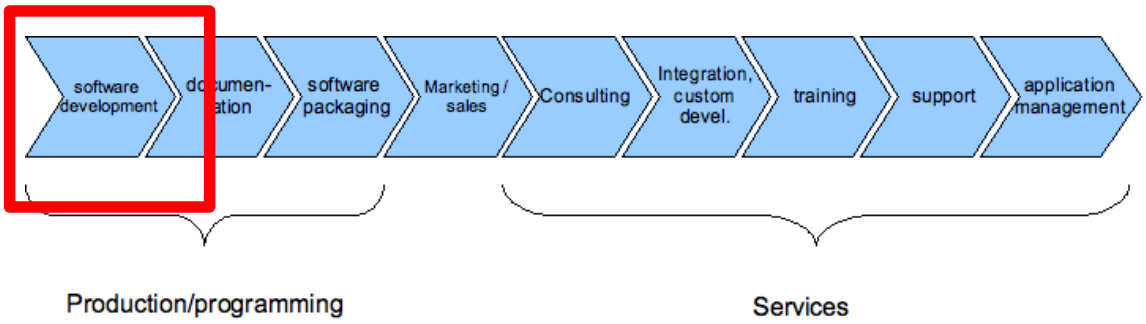
Földrajzi szegmentáció: globális

Típus szegmentáció: B2B

Szoftvertermék szegmentáció: nem releváns

5.2.3.2 Szoftver értéklánc

Annak a vállalatnak, amelyik ezt az üzleti modellt használja, különösen az értéklánc szoftverfejlesztési lépésére kell koncentrálnia. A jövendőbeli ügyfelek számára a termék értéke a termék minőségétől és a benne rejlő lehetőségektől függően nő.



Meg kell jegyeznünk, hogy az ezt a modellt használó vállalatoknak különösen figyelniük kell arra, hogy termékük hogyan működhet egy nagyobb szoftvertermék vagy rendszer részeként.

5.2.3.3 Terjesztési modell

A NYFSZ terjesztési modell (Hiba: A hivatkozás forrása nem található pont) széles körben ismertté teszi a terméket fejlesztői körökben, és segít a terméket felismerhető márkává tenni a szoftverekkel foglalkozó szakmai körökben.

5.2.4 A jövedelem modell

A jövedelem modell az üzleti licencek hagyományos licenc díjazásán alapul. A jövedelem modell megegyezik a megfelelő zárt forráskódú vállalatok által használt modellekkel.

5.2.5 Az NYFSZ licenc szerepe a modellben

Ez a modell a NYFSZ terjesztési modellt alapvetően arra használja, hogy a terméket hitelképes jelöltté tegye a vevő saját (szellemi tulajdont képező) termékének technológiai inputjaként.

Ha működtetni akarjuk, kölcsönös licencre van szükség (akadémiai licenc alatt, mint amilyen például a BSD licenc, a leendő vevőt nem lehet rávenni, hogy az üzleti licencet válassza). Végül a modell megköveteli, hogy a vállalat rendelkezzen a szoftver szerzői jogával. A gyakorlatban ez azt jelenti, hogy maga a szoftver nem alapulhat kölcsönös licenc alatti külső NYFSZ forráskódra.

5.2.6 Példák a modellt alkalmazó vállalatokra

A MySQL AB, amelyik a népszerű MySQL adatbázist gyártja, a beágyazott Berkeley DB adatbázist gyártó Sleepycat Software. Sleepycat-et nemrégiben vásárolta fel az Oracle.

5.2.7 A modell erősségei és gyengeségei

Erősségek: magas megtérülési rátájú licenc jövedelmeket biztosít.

Gyengeségek: a forráskód teljes tulajdonjogát teszi szükségessé.

5.3 A Támogatás-értékesítő modell

Ebben a modellben a NYFSZ terméket előállító vállalat támogató szolgáltatásokat kínál a termék felhasználóinak. A modell azon az előtételen alapul, hogy a szoftver létrehozói a legalkalmasabbak a támogatás nyújtására, mert ők az alkotók.

5.3.1 Értékelőny

A felhasználók bizonyos csoportjai számára a NYFSZ átvételét akadályozza a formális támogatás hiánya. Ez különösen így van a nagyvállalati és a közszférába tartózó ügyfelek esetében. Ezek az ügyfelek általában inkább olyan vállalatoktól szerzik be technológiájukat, amelyek az esetleges hiányosságokért és hibákért számonkérhetőek. Ebben a modellben a vállalat biztosítja, hogy a NYFSZ felhasználók ugyanolyan szintű támogatást kapnak, mint amit az üzleti alapú zárt forráskódú vállalatok biztosítanak.

5.3.2 Termékek/szolgáltatások

- Ez általában egy szabványosított támogatási szolgáltatáscsomag a NYFSZ termékhez. A csomag tartalmazhat e-mail és telefonos támogatást; vebrendszeren keresztül működő automatizált támogatást (pl. a biztonsági frissítésekről automatikus értesítés küldése); távolsági frissítési szolgáltatásokat. Általában hozzátartozik egy Szolgáltatási Szint Megállapodás is, amely meghatározza a hibajelentésekre való reagálás maximális idejét. Erre a szolgáltatás kínálatra jó példa az MySQL ajánlat, mely elérhető a https://shop.mysql.com/enterprise/ címen.

- Képzési ajánlatok

5.3.3 Az értékteremtés architektúrája

5.3.3.1 A piacterv

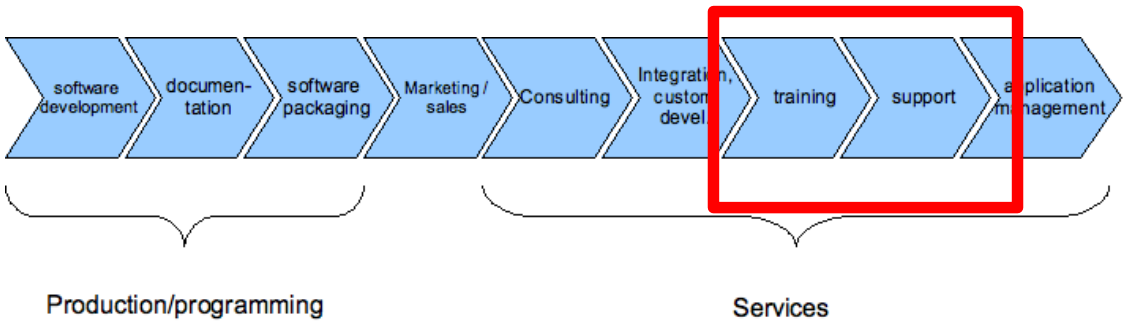
Földrajzi szegmentáció: globális

Típus szegmentáció: B2B és B2G.

Szoftvertermék szegmentáció: vállalkozói megoldások

5.3.3.2 Szoftver értéklánc

Az ezt a modellt alkalmazó vállalatok az értéklánc képzés és támogatás lépéseire helyezik a hangsúlyt.



A szoftverfejlesztés önmagában nem generál jövedelmet, mivel a terméket tisztán NYFSZ-ként kínálják. Mindazonáltal nagyon fontos, mert ez alapozza meg a hitelességét annak az állításnak, hogy a nyújtott szolgáltatás az elérhető legjobb.

5.3.3.3 Terjesztési modell

A NYFSZ terjesztési modellt telepítések nagy számának a létrehozására használják. A telepítések számát lehet a későbbiekben szolgáltatástámogatási jövedelem céljára felhasználni.

Ez az üzleti modell jelentős marketinget igényel, mert a felhasználókat meg kell győzni arról, hogy megéri fizetni az üzleti támogatásért.

5.3.4 A jövedelem modell

Szabvány szolgáltatási csomagokat kínálnak Szolgáltatási Szint Megállapodás formájában vagy pedig (jellemzően) a támogatásra való éves, fix árú előfizetéstor. Ez utóbbi modell a legfontosabb.

5.3.5 A NYFSZ licenc szerepe a modellben

Ez a modell a kölcsönös és az akadémiai licencek esetében is működik. Nincs szükség a teljes kódbázis tulajdonjogára.

A NYFSZ kétféle módon játszik fontos szerepet ebben a modellben. Mint fejlesztési modell, a NYFSZ lehetővé teszi, hogy a vállalat gyorsan és viszonylag korlátozott befektetéssel létrehozzon egy terméket. Terjesztési modellként segíti a vállalatot egy nagyobb felhasználói bázis megalapozásában Ennek a felhasználói bázisnak egy része a fizetett támogatási szolgáltatások piacáról kerül ki. Szembeszökő példa erre a JBoss.com (most már a Red Hat része), amely a JBoss alkalmazás szervert hozza létre. Három év működés után a JBoss körülbelül 30%-os piaci részesedést ért el. Hat év után több mint 60 millió USD jövedelemre tett szert, nagyrészt támogatási szerződésekből.

5.3.6 Példák a modellt alkalmazó vállalatokra

MySQL AB (adatbázis rendszerek), Alfresco (Tartalom menedzsment rendszerek), Talend (ún. Extract-Transform-Load – adatok kivonására, átalakítására és tárolására szolgáló - eszközök), Compière (Vállalkozási erőforrás-tervezés), JBoss (Java alkalmazás szerverek; most a Red Hat egy részlege). Ezek a vállalatok NYFSZ termékeket hoztak létre és a támogatási szolgáltatások értékesítésével tettek szert jövedelemre.

A JotSpot a Dojo könyvtár és sok más Web 2.0 technológia területén kínál képzéseket, melyek létrehozásában közreműködött.

5.3.7 A modell erősségei és gyengeségei

Erősségek:

- A jövedelem modell magas árréseket tesz lehetővé, mivel a költségek nem emelkednek a jövedelemmel arányosan (a szabvány csomagokat fix áron kínálják).
- Ismétlődő bevételek, mivel a legtöbb felhasználó meghosszabbítja a támogatási szerződést.
- A kód tulajdonjoga kevésbé fontos; csak a hatékony műszaki vezetésnek és a know-how-nak van jelentősége.

Gyengeségek:

- Jelentős üzleti erőfeszítésekre van szükség ahhoz, hogy a vevőket az ajánlat értékéről meggyőzzék.
- Az értékelőny csak azon ügyfelek számára vonzó, akik a terméket üzleti szempontból kritikus rendszerekben használják és jó minőségű támogatásra van szükségük.
- Feszültségben áll a NYFSZ szellemével. A NYFSZ közösség azt várja a fejlesztőktől, hogy bizonyos fokú támogatást nyújtsanak, és a hibákat hárítsák el. Ha a NYFSZ közösség nem fizető tagjai úgy érzik, hogy az ő problémáik a háttérbe szorulnak a fizető tagokéval szemben, elvesztik a termék iránti érdeklődésüket és bizalmukat. Másrészről viszont, ha a közösség nagy és aktív, a közösség általi támogatás minősége elejét veheti a legtöbb felhasználó professzionális támogatás iránti igényének. Ezt említik az egyik okként, amiért a népszerű Postgresq adatbázisnak nincs Támogatásértékesítője.
- A belépés határai alacsonyak: ugyanazokat az ajánlatokat külső vállalatok is megtehetik (lásd a Külső szolgáltatás nyújtó modellt). Verseny esetén a vállalat egyetlen versenyelőnye a márka („a … gyártóinak támogatásával”) és a technológia feletti ellenőrzés.

5.4 A Külső támogatásértékesítő modell

Ez lényegében megegyezik az előző üzleti modellel, azzal a kivétellel, hogy a támogatási szolgáltatásokat kínáló vállalat nem a tulajdonosa (a szerzői jog tulajdonosa) a támogatott NYFSZ-nek. Még az sem szükséges, hogy a vállalat a vezető fejlesztői csoport tagja legyen.

Ezt a modellt gyakran használják, amikor az NYFSZ termék vagy semelyik egyéni fejlesztő vagy fejlesztő cég tulajdonát sem képezi, vagy amikor a forráskód tulajdonosa nem képes professzionális támogatási szolgáltatások nyújtására (pl. amikor inkább egy magánszemélyről és nem egy nagyvállalatról van szó).

5.4.1 A NYFSZ licenc szerepe a modellben

A forráskód szabad rendelkezésre állása teszi ezt a modellt megvalósíthatóvá. Viszont a NYFSZ drámaian lecsökkenti a belépési korlátot azon vállalatok számára, amelyek versengő NYFSZ szolgáltatásokat akarnak kínálni.

Ezen modell előfeltételeit szintén a NYFSZ biztosítja. A NYFSZ közösség hozza létre a szoftvert és az installált bázist. A formális, professzionális támogatás hiánya azután létrehozza ezen támogatási szolgáltatások iránti piaci keresletet.

5.4.2 Példák a modellt alkalmazó vállalatokra

A Spikesource (http://www.spikesource.com) számos nyílt forráskódú csomaghoz kínál támogatási szolgáltatásokat a tartalom menedzsment, ügyfélkapcsolat menedzsment és együttműködés területén. Ezen szolgáltatások célközönségét a vállalati vagy a közszférába tartozó ügyfelek alkotják. Az egyik ajánlatuk a Drupal Tartalom Menedzsment Rendszert veszi célba. A Drupal-t számos egyéni szoftverfejlesztő alkotta meg, s a közösségből jelentős hozzájárulást kaptak a forráskódhoz. Senki sem mondhatja magáénak a Drupal forráskódot, a fejlesztést pedig nem ellenőrzi semmilyen üzleti szervezet. A Drupal esetében a Spikesource biztosítja azokat a professzionális támogatási szolgáltatásokat, amikre a nagyvállalatoknak szükségük van.

5.4.3 A modell erősségei és gyengeségei

A modell erősségei és gyengeségei ugyanazok, mint a támogatás értékesítési modell esetében, azzal a kivétellel, hogy a belépési korlátok még alacsonyabbak. A szolgáltatás ajánlatot a versenytársak könnyen lemásolhatják. Amikor ez előfordul, a vállalat nem tudja magát olyan színben feltüntetni, mintha a termékkel kapcsolatban privilegizált helyzetben lenne, mint ahogyan ez a támogatásértékesítő modell esetében van. Ez arra ösztönzi a vállalatot, hogy aktívan és láthatóan vegyen részt a szoftverfejlesztésben.

5.5 A platformszolgáltató modell

A vállalat több NYFSZ terméket fog össze egy teljes megoldásba vagy platformba. A vállalat minőségi garanciát vállal arra, hogy a kiválasztott terméket működőképesek együtt. Olyan vállalatok használják ezt a modellt, mint a Red Hat és a Novell, amelyek üzleti Linux terjesztést kínálnak.

Ez a modell általában a (Külső) támogatásértékesítő modellel keveredik. Először is azért, mert sokkal könnyebb egy teljes megoldást (platformot) támogatni és annak hibáit elhárítani, mivel nagyobb ellenőrzést biztosít a működési környezet felett. Másodsorban pedig, az értékelőny hangsúlyosabb az ügyfél számára, ha ugyanattól a beszállítótól tudja beszerezni a platformot és a vonatkozó támogatást is.

5.5.1 Értékelőny

Az ügyfél számára az értéket az jelenti, hogy szakemberek előválogatják és tesztelik a NYFSZ összetevőket, amelyek együttesen alkotják a számára érdekes megoldást vagy platformot. A vállalat igazolja, hogy az általa összeállított NYFSZ összetevők együtt működőképesek. Ennek a terméknek a hiányában az ügyfeleknek sok időt és energiát kelljen arra fordítaniuk, hogy a különféle NYFSZ összetevők simán tudjanak együttesen működni. És nem lenne garancia a minőségre.

5.5.2 Termék/szolgáltatások

Előre konfigurált, tesztelt és optimalizált NYFSZ termékek kötege, amelyek együttesen egy megoldást vagy platformot biztosítanak.

5.5.3 Az értékteremtés architektúrája

5.5.3.1 A piacterv

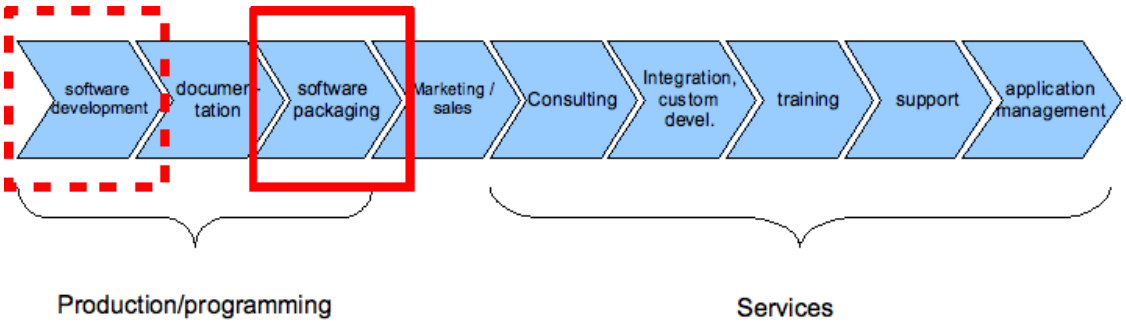
Földrajzi szegmentáció: globális

Típus szegmentáció: B2B, B2G

Szoftvertermék szegmentáció: Vállalkozói megoldások

5.5.3.2 Szoftver értéklánc

Ebben a modellben a hangsúly a szoftvercsomagoláson van. A szoftverfejlesztés is fontos annak biztosítására, hogy a számos összetevő együtt is működőképes legyen. Ehhez tesztelésre, hibaelhárításra, valamint továbbfejlesztett és optimalizált verziók készítésére van szükség.



5.5.3.3 Terjesztési modell

A modell előfeltételezi, hogy a NYFSZ termékeket már használják ebben a kombinációban, és hogy ennek piaca is van. Ennek ellenére jelentős marketing munkára van szükség, az összerakott megoldás előnyeinek és rendelkezésre állásának népszerűsítésére.

5.5.4 Jövedelem modell

Általában a licenc díj. Az üzleti modell azonban általában a támogatásértékesítő modellel kombinálódik. Ebben az esetben a licenc díj az összeállított termékkel együtt a támogatási szolgáltatásokhoz való hozzáférést is fedezi.

5.5.5 A NYFSZ licenc szerepe a modellben

A NYFSZ teremti meg a keresletet, amit ez a modell megcéloz: szoftverek sokasága áll rendelkezésre, amikor is a felhasználó számára nem a szoftverhez való hozzáférés okoz problémát, hanem hogy kiválogassa, mely ajánlatok hordoznak számára értéket, és milyen elemek fognak tökéletesen működni együtt.

Ez a modell a kölcsönös és az akadémiai licencek esetében is működik. Viszont amennyiben a NYFSZ összetevőkre kölcsönös licenc vonatkozik, a vállalatnak minden, az összetevőn megvalósított továbbfejlesztést közre kell adnia.

5.5.6 Példák a modellt alkalmazó vállalatokra

A Red Hat és a Novell biztosítják a vállalkozói felhasználáshoz a Linux terjesztést (Red Hat Enterpise Linux, SUSE Linux Enterprise).

A SourceLabs az SASH-t biztosítja: egy teljes optimalizált szoftvercsomag a Java-alapú vállalkozói alkalmazásokhoz. A következő NYFSZ termékekből áll: Spring, Axis, Struts és Hibernate. A SourceLabs számos hibát kijavított és sok továbbfejlesztést valósított meg. Fontos tényező, hogy a SourceLabs gondoskodik arról, hogy ezen termékek legmegbízhatóbb termékverziói jelenjenek meg saját termék kombinációikban. A szlogenjük a következő: „Többé ne pazarolja idejét saját támogatással, integrálással és teszteléssel.”.

A Spikesource ezt a modellt a Külső támogatásértékesítő modellel kombinálja. Például a Drupal ajánlatuk neve: Drupal SpikeIgnited. A termék adatlapján ez szerepel: „a Drupal

SpikeIgnited megoldást a SpikeSource tesztelt, integrált és igazolt nyílt forráskódú infrastruktúra elemei jellemzik. A SpikeSource ezen összetevők minden verzióját értékeli, a kombinációkat integrálja és konfigurálja, hogy jól működjenek a Drupal-lal, valamint teszteli őket ..., hogy a konfiguráció biztosan megfeleljen az Ön gyártási igényeinek." A Drupal ajánlat Drupal tartozékokat is tartalmaz, mint például a MySQL adatbázis és a LDAP.

5.5.7 A modell erősségei és gyengeségei

Erősségek:

- Licenc jövedelmet biztosít

Gyengeségek:

- A hozzáadott érték túl kicsi lehet ahhoz, hogy vonzó legyen az ügyfelek számára. Ez a másik oka annak, amiért általában a Támogatásértékesítő modellel kombinálják.
- Nagy erőfeszítésekre van szükség ahhoz, hogy a piacon a minőségbiztosítás hitelképes legyen.

5.6 A Tanácsadási modell

A vállalat több NYFSZ termékre vonatkozóan nyújt tanácsadási és testreszabási szolgáltatásokat. Minden bizonnyal ez a legszélesebb körben elterjedt modell.

5.6.1 Értékelőny

Az ügyfelek hozzáférést nyernek a NYFSZ-szel kapcsolatos szakértelemhez és szoftverfejlesztési képességekhez.

5.6.2 Termék/szolgáltatások

Tanácsadás, testreszabás és egyedi szoftverfejlesztési szolgáltatások.

5.6.3 Az értékteremtés architektúrája

5.6.3.1 A piacterv

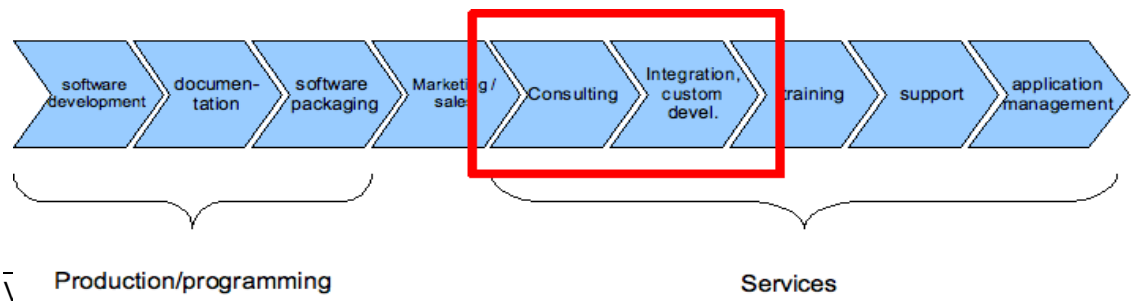
Földrajzi szegmentáció: helyi, a földrajzi közelségtől és a közös nyelvi/kulturális háttértől függ.

Típus szegmentáció: B2B, B2G

Szoftvertermék szegmentáció: Vállalkozói megoldások

5.6.3.2 Szoftver értéklánc

A vállalatok az értéklánc tanácsadási és integráció/személyre szabott fejlesztés lépéseire koncentrálnak.



5.6.3.3 Terjesztési modell

A terjesztési modell ugyanaz, mint bármely más tanácsadó cég esetében: ügyfélkapcsolat menedzsment, hirdetés, pályázás nyitott vagy meghívásos tendereken.

5.6.4 Jövedelem modell

A szolgáltatásokat általában idő és eszköz alapon értékesítik. A személyreszabott fejlesztésekre gyakran szabott árral szerződnek.

5.6.5 A NYFSZ licenc szerepe a modellben

A Tanácsadási modellben a NYFSZ szerepe, hogy alacsonyabbá teszi a belépési korlátot. A tanácsadó cégeknek nem szükséges költséges szoftvertermékekbe vagy fejlesztési eszközökbe befektetniük. A forráskód, valamint a szoftver részleteire vonatkozó (általában) nagy mennyiségű publikált információ rendelkezésre állása azt is jelenti, hogy sokkal könnyebb szakértői szintű tudásra szert tenni a termékkel kapcsolatban.

5.6.6 Példák a modellt alkalmazó vállalatokra

Az Accenture, a Red Hat, a Unisys és az IBM néhány azok közül a nagy márkanévvel rendelkező vállalatok közül, amelyik a NYFSZ-re vonatkozóan tanácsadói szolgáltatást nyújtanak. A modell a helyi piacokat megcélzó KKV-k körében is nagyon elterjedt. A legtöbb KKV a Linux-ra vagy más „infrastruktúra” NYFSZ-re koncentrál.

5.6.7 A modell erősségei és gyengeségei

Erősségek:

- A modell nagyon alacsony induló költséget és befektetést igényel.

Gyengeségek:

- A jövedelem egyenesen arányos a munkaerő ráfordítással, ami azt jelenti, hogy a haszonkulcs mérsékelt marad.

5.7 A szoftver mint szolgáltatás modell

Ebben a modellben a NYFSZ-t egy veben keresztül elérhető alkalmazási szolgáltatás létrehozására használják fel. Az ilyen rendszereket “Software as a Service” (SaaS) – „Szoftver mint szolgáltatás” címkével illetik.

5.7.1 Értékelőny

Kifinomult alkalmazás funkcionalitást kínálnak alacsony költségek mellett. Az ügyfél számára az a tény is kedvező, hogy a SaaS-hoz nincs szükség IT adminisztrációra vagy szerver hardver erőforrásokra.

5.7.2 Termék/szolgáltatások

Hozzáférés a szoftverhez, mint szolgáltatás (SaaS).

5.7.3 Az értékteremtés architektúrája

5.7.3.1 A piacterv

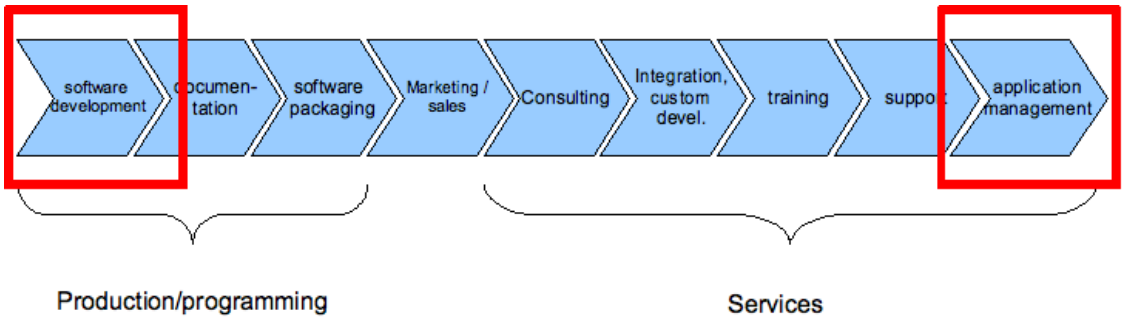
Földrajzi szegmentáció: globális

Típus szegmentáció: többnyire B2B és B2G. A legtöbb ügyfélközpontú SaaS (mint például a flicker és a YouTube) az ügyfél számára ingyenes és a hirdetéseken keresztül jövedelmet generál (ebből a szempontból a B2B is).

Szoftvertermék szegmentáció: Vállalkozói megoldás

5.7.3.2 Szoftver értéklánc

A SaaS-hoz a vállalatnak ki kell fejlesztenie a szoftverszolgáltatást. A szoftverfejlesztéshez szükséges befektetés jelentős, mert a biztonságnak, a megbízhatóságnak és a skálázhatóságnak felbecsülhetetlen értéke van az ajánlat hitelessége szempontjából. Emiatt a vállalatnak az alkalmazásmenedzsment lépésbe is be kell fektetnie: megbízható szerver infrastruktúra, hálózatosodás, 24/7 alkalmazás menedzsment,...



5.7.3.3 Terjesztési modell

Ahhoz, hogy ez a modell sikeres legyen, a vállalatnak képesnek kell lennie egy erős márka kiépítésére. A vállalatnak meg kell győznie az ügyfelet arról, hogy rábízhatja az adatokat.

5.7.4 Jövedelem modell

Az ügyfél általában havi díjat fizet az alkalmazási szolgáltatásokhoz való hozzáférésért.

5.7.5 A NYFSZ licenc szerepe a modellben

A NYFSZ-t nagyon gyakran használják fel SaaS szolgáltatások kiépítésének az alapjául. A legtöbb kölcsönös licenc, beleértve a GPL-t is (a v3 és az azt megelőző verziók is) nem szabnak meg kötelezettségeket arra vonatkozóan, miszerint a SaaS-t megvalósító szoftvereket nyílt forráskóddal kellene megjelentetni. A SaaS biztosítását nem tekintik szoftverforgalmazásnak (ezt nevezik a Saas kiskapunak).

5.7.6 Példák a modellt alkalmazó vállalatokra

A legismertebb cég, amely ezt a modell alkalmazza, kétségkívül a Google. Egy másik példa a Salesforce.com az ügyfélkapcsolat menedzsment, valamint a Smallthought az adatbázis menedzsment területén.

5.7.7 A modell erősségei és gyengeségei

Erősségek:

- Erős jövedelem modell
- Az értékelőnyt könnyű elmagyarázni
- A „SaaS kiskapu” azt jelenti, hogy a vállalatnak a szolgáltatás szoftverét nem kell megosztania.

Gyengeségek:

- Erős márkára van szükség a sikerhez
- Jelentős előzetes befektetést igényel az alkalmazásfejlesztés és az infrastruktúra kiépítése területén.

5.8 A hozzáadott érték nyújtási modell

Ebben a modellben a vállalat egy NYFSZ-ből származó szellemi tulajdont képező szoftvertermékét származtat.

5.8.1 Értékelőny

Az alkalmazás funkcionalitása, amennyiben az nincs jelen abban a NYFSZ-ben, amiből a terméket származtatták.

5.8.2 Termék/szolgáltatások

Szoftvertermék bináris kódok és végfelhasználói licenc.

5.8.3 Az értéklánc architektúrája

5.8.3.1 A piacterv

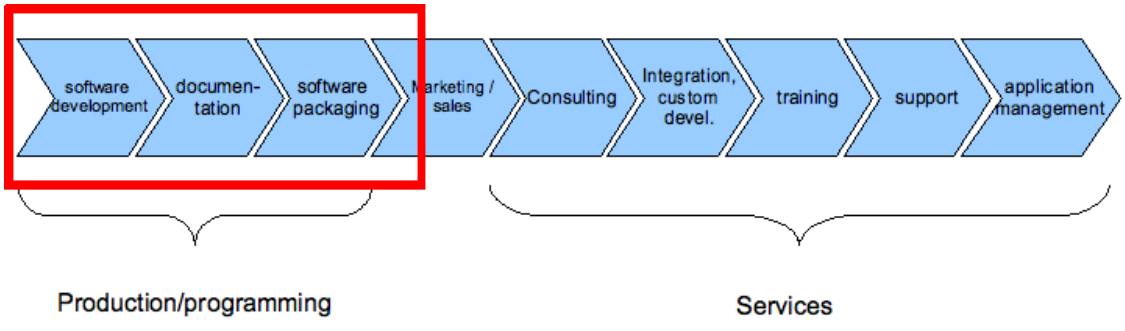
Földrajzi szegmentáció: globális

Típus szegmentáció: B2B, B2G, B2C

Szoftvertermék szegmentáció: nem releváns

5.8.3.2 Szoftver értéklánc

A szoftvercég a hagyományos szoftvergyártókhoz hasonló pozíciót foglal el az értéklláncban.



5.8.3.3 Terjesztési modell

A vállalat ugyanolyan terjesztési modellel rendelkezik, mint a hagyományos szoftvergyártók.

5.8.4 Jövedelem modell

A jövedelem modell ugyanolyan, mint a hagyományos szoftvergyártók esetében: licenc jövedelem.

5.8.5 A NYFSZ licenc szerepe a modellben

Ez a modell a már korábban létező NYFSZ használatával ér el költségcsökkentést. Annak érdekében, hogy a teljes termék magántulajdonban maradhasson, a NYFSZ-nek az akadémiai BSD stílusú licencet kell alkalmaznia.

5.8.6 Példák a modellt alkalmazó vállalatokra

Az Apple ezt a modellt használja OS X operációs rendszer családjához. Az Apple az OS Y magjaként a FreeBSD-t használta fel, és ehhez adott hozzá szabadalmaztatott jellemzőket, mint például a Quartz 2D grafikus technológia, a Carbon alkalmazási környezet és az Aqua grafikus felhasználói interfész. Az Apple választása azért a FreeBSD-re esett, mert az a BSD akadémiai licenccel járt együtt, ami nem kötelezte az Apple-t ezen technológiák nyílt forráskódúvá tételére, s így az egész csomagot értékesíteni tudta.

5.8.7 A modell erősségei és gyengeségei

Erősségek:

- Jól ismert és bevált üzleti modell
- Magas haszonkulcsot lehetővé tévő licenc jövedelem modell

Gyengeségek:

- Kizárólag akadémiai licencekkel működik
- Ugyanaz, mint a zárt forráskódú licenc modellek esetében
- Jelentős márkaépítést igényel, melynek segítségével az ügyfeleket rá lehet venni a szoftver beszerzésére
- A hozzáadott értéknek jelentősnek kell lennie, mert különben a NYFSZ projektek ingyenesen tudják kínálni ugyanazt a funkcionalitást.

5.9 A kiegészítő modell

A vállalat a NYFSZ termékekhez való fizikai kiegészítőket árusít. Ezek közül a legfontosabbak a műszaki könyvek és kézikönyvek.

5.9.1 Értékelőny

Egy műszaki kiadó alapesetét figyelembe véve, az értéket a NYFSZ termékre vonatkozó első osztályú műszaki információkhoz való hozzáférés jelenti. A NYFSZ dokumentációk gyakran kaotikus volta teszi ezeket a könyveket értékessé a felhasználók számára, különösen, ha az egyik vezető fejlesztő írja őket.

5.9.2 Termék/szolgáltatások

Kézikönyvek, műszaki szakirodalom.

5.9.3 Az értékteremtés architektúrája

5.9.3.1 A piacterv

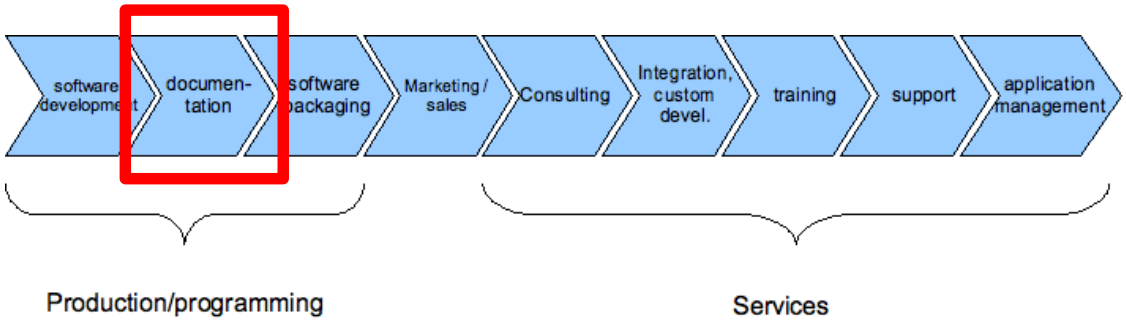
Földrajzi szegmentáció: globális

Típus szegmentáció: többnyire B2C

Szoftvertermék szegmentáció: nem releváns

5.9.3.2 Szoftver értéklánc

A vállalat az értéklláncnak kizárólag a dokumentáció lépésére koncentrál.



5.9.3.3 Terjesztési modell

Ugyanaz, mint bármely más könyvkiadó esetében. Ha a könyvet az egyik vezető fejlesztő írja, a kiadónak meg van az az előnye, hogy a könyvet különösen mértékadónak fogják tartani.

5.9.4 A jövedelem modell

Könyvértékesítésből származó jövedelem.

5.9.5 A NYFSZ licenc szerepe a modellben

A NYFSZ licenc nem játszik szerepet. Ehhez az üzleti modellhez a NYFSZ az előfeltételeket teremti meg, nem csak úgy, hogy a műszaki könyvek témáját biztosítja, hanem bizonyos esetekben a konzisztens, jó minőségű dokumentáció hiánya miatt is.

5.9.6 Példák a modellt alkalmazó vállalatokra

Az O'Reilly és a Manning két olyan kiadó, melyek mindig különös figyelmet szenteltek a NYFSZ tematikájú könyvekre.

5.9.7 A modell erősségei és gyengeségei

Erősségek:

- Hagyományos, jól ismert kiadói modell

Gyengeségek:

- Bizonyos NYFSZ projektek esetében a változás üteme olyan gyors, hogy a könyvek hamar elavulnak.

5.10 A veszteségvezető modell

Ebben a modellben a szoftvercég a NYFSZ terméket a szellemi tulajdont képező termék veszteségvezetőjeként és piac pozícionálójaként kínálja. A szellemi tulajdont képező termék további magas értékű funkcionalitást kínál. Az elképzelés az, hogy az ezen funkciókat igénylő ügyfelek hajlandóak is megfizetni őket. A veszteségvezető modellt gyakran használják a zárt forráskódú szellemi tulajdont képező szoftverek üzleti modelljében.

5.10.1 Értékelőny

Az értékelőny ugyanaz, mint a hagyományos szoftvercégeknél. A modell csak a terjesztési modellt változtatja meg, az értékelőnyt nem.

5.10.2 Termék/szolgáltatások

NYFSZ a veszteségvezető szoftvertermék bináris kódokhoz és végfelhasználói licencek az után követő termékekhez.

5.10.3 Az értéklánc architektúrája

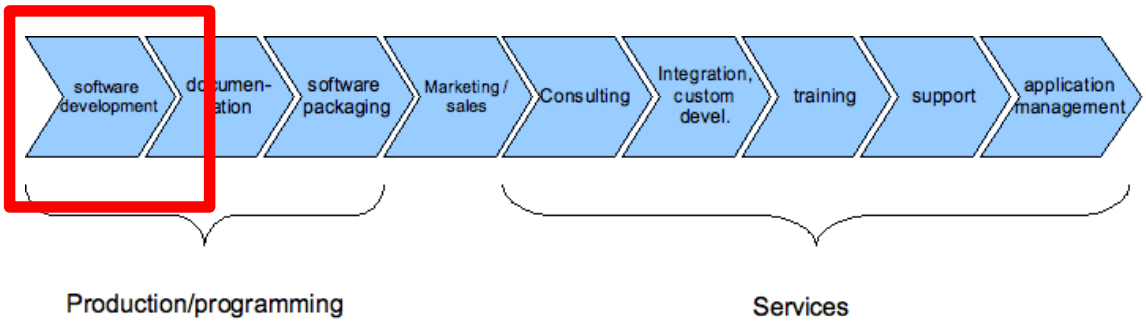
5.10.3.1 A piacterv

Földrajzi szegmentáció: globális

Típus szegmentáció: B2B, B2G, B2C

Szoftvertermék szegmentáció: nem releváns

5.10.3.2 Szoftver értéklánc



5.10.3.3 Terjesztési modell

A veszteségvezető generálja a piaci igényt a szellemi tulajdont képező termékekre. A veszteségvezető NYFSZ jellege segíti a széles körben történő terjesztést és segít a vállalat, valamint a szoftver szakmai közösségen belül a szellemi tulajdont képező termék márkaépítésében.

5.10.4 Jövedelem modell

A veszteségvezetőt követő szellemi tulajdont képező szoftvercsomagokból származó licenc (és támogatási) jövedelmek.

5.10.5 A NYFSZ licenc szerepe a modellben

A NYFSZ nem nélkülözhetetlen ahhoz, hogy a veszteségvezető modellel piaci részesedést érjünk el. Valójában a legtöbb szoftvercég, amelyik a veszteségvezető modellt alkalmazza, a veszteségvezetőt levédeti (pl. Acrobat Reader, QuickTime Player, ESRI ArcExplorer, ...). Akkor mások miért teszik nyílt forráskódúvá? Íme néhány ok:

- A termék nyílt forráskódúvá tétele azt jelenti, hogy az OSS Platformszolgáltatók, mint pl. a Linux Red Hat vagy a Canonical (Ubuntu) a veszteségvezetőt bevonhatják a terjesztésbe. Ez a veszteségvezető széles körű terjesztését teszi lehetővé.

- Amikor a veszteségvezető nem osztozik a hozzáadott értékű szellemi tulajdont képező után követő termékekkel a kódon, a NYFSZ csökkentheti a veszteségvezető fejlesztési költségét.

- Számos esetben a veszteségvezető egy platform termék, amelyik generikus és/vagy alacsony értékű alkalmazásszolgáltatást kínál. A vállalat azáltal tesz szert jövedelemre, hogy ehhez a platformhoz nyújt szellemi tulajdont képező kiterjesztéseket vagy beépíthető modulokat. A platform NYFSZ jellege előnyöket biztosít a kezdeti fejlesztésnél (egyéb NYFSZ termékek beemelésével) és a széleskörű terjesztésnél.

Az utolsó ok a veszteségvezető egy gyakori alkalmazását mutatja be, amit platform veszteségvezető modellnek fogunk nevezni.

5.10.6 Példák a modellt alkalmazó vállalatokra

A VMWare nyílt forráskódúvá tette az alap virtualizációs alaptermékeit 2006-ban és ezeket üzleti termékeik veszteségvezetőjeként használta fel. A nyílt forráskódú veszteségvezetők alkalmazására az motiválta őket, hogy népszerű Linux terjesztésekbe építsék be őket.

Több vállalat kínál szellemi tulajdont képező, üzleti beépülő modulokat a sikeres NYFSZ IDE Eclipse-hez.

5.10.7 A modell erősségei és gyengeségei

Erősségek:

- Jól ismert jövedelem modell (szellemi tulajdont képező szoftverből származó licence jövedelem).

Gyengeségek:

- Egy veszteségvezető nyílt forráskódúvá tétele általában kevés ösztönzést jelent a közösség számára a fejlesztésében való aktív részvételre. A nyílt forrás csak a terjesztés és/vagy az alacsonyabb fejlesztési költség szempontjából jelent előnyt.

- A platform veszteségvezető modellben nehéz a helyes egyensúlyt megtalálni a között, hogy mi legyen a NYFSZ platformban, és mit tartsunk meg szellemi tulajdont képezőnek. A platform termék körüli közösség minden funkcionalitást a NYFSZ platformba akar áttenni, a vállalat pedig bizonyos funkcionalitásokat szellemi tulajdont képezőnek akar megtartani. Ebben az összefüggésben a közösségen belüli kapcsolatok menedzselése nagyon fontossá válik. (Az Eclipse beépülő modulok esetében a legtöbb szellemi tulajdont képező beépülő modulokat gyártó vállalat végül emiatt tette nyílt forráskódúvá azokat.) A platform veszteségvezető modell akkor működik a legjobban, amikor a szellemi tulajdont képező kiterjesztések nagyon magas értéket képviselnek, amelyeket nem könnyű kifejleszteni a közösségben, vagy szabadalmakkal terhelni (pl. Microsoft Exchange beépülő modulok a NYFSZ csoportos használatú termékekhez).

5.11 A zárvány modell

Ebben a modellben (angolul Widget Frosting Model) a vállalat egy teljes megoldást értékesít az ügyfélnek, amely jellemzően mind hardvert, mind pedig szoftvert tartalmaz. A megoldás az ár csökkentésének eszközeként tartalmaz NYFSZ terméket is, anélkül, hogy a vállalat profitja csökkenne. Ezt a modellt leginkább szellemi tulajdont képező termékek forgalmazói (SAP, Oracle) és Rendszer integrátorok alkalmazzák (EDS, IBM, Accenture, ...).

5.11.1 Értékelőny

Teljes hardver és szoftver megoldás alacsonyabb költséggel.

5.11.2 Termék/szolgáltatások

Teljes vállalkozói megoldások.

5.11.3 Az értékteremtés architektúrája

5.11.3.1 A piacterv

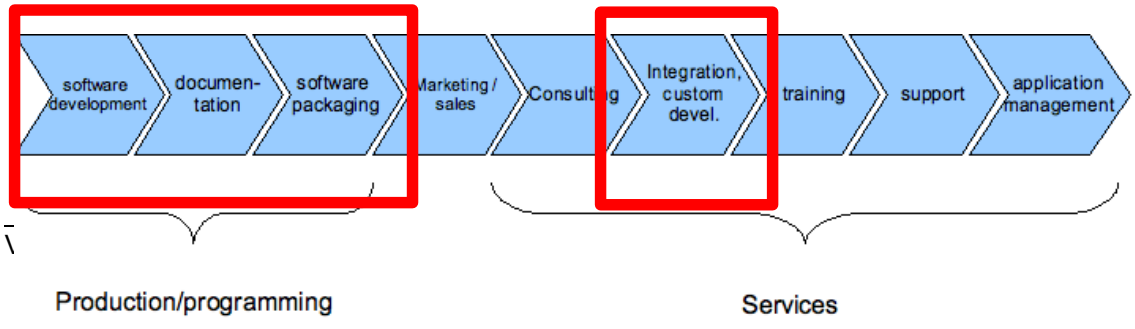
Földrajzi szegmentáció: globális

Típus szegmentáció: B2B, B2G

Szoftvertermék szegmentáció: Vállalkozói megoldások

5.11.3.2 Szoftver értéklánc

A zárvány modell a hagyományos szellemi tulajdont képező szoftver értékesítők és a Rendszer integrátorok modelljének egy változata. Az értékesített elsődleges termék egy szellemi tulajdont képező szoftver vagy hardver ajánlat, vagy pedig egy rendelésre készült megoldás.



5.11.3.3 Terjesztési modell

A terjesztési modell itt a rendszer integrátorok modellje: ügyfélkapcsolat menedzsment, hirdetés, pályázás meghívásos vagy nyílt tendereken.

5.11.4 Jövedelem modell

A jövedelem modell a tejes megoldás rögzített árán alapul

5.11.5 A NYFSZ licenc szerepe a modellben

A NYFSZ jó minőségű, alacsony költségű szoftver összetevőket biztosít, amelyek csökkentik a megoldás költségeit.

5.11.6 Példák a modellt alkalmazó vállalatokra

Amikor az Oracle-t arra kérték, hogy egy hardvert is tartalmazó adatbázis megoldást szállítson, a megoldást a Linux OS-re alapozta. Ez lehetővé tette, hogy a vállalat az Operációs Rendszert törölhesse, mint ajánlata költségtételét.

5.11.7 A modell erősségei és gyengeségei

Erősségek:

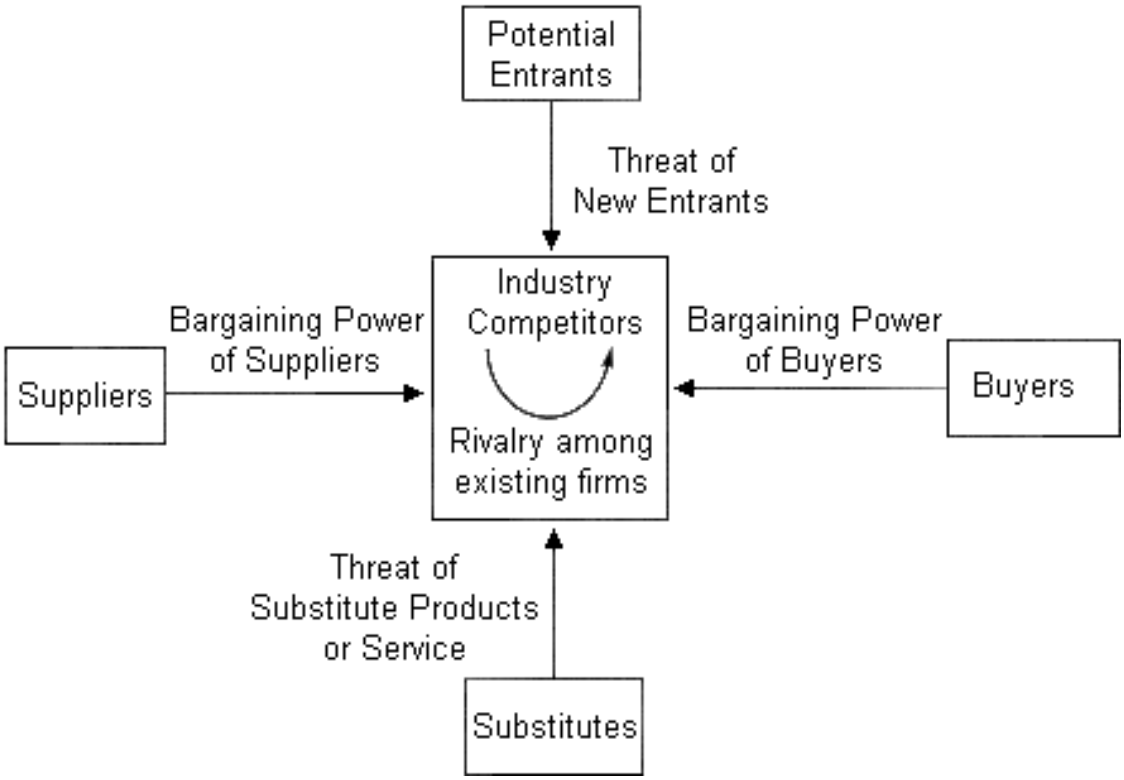
- A költségcsökkentésnek csak egy részét kell az ügyfélnek továbbadni. A haszonkulcsok magasabbak lehetnek, mint ahol NYFSZ helyett szellemi tulajdont képező szoftvert használnak alternatívaként.

Gyengeségek:

- Ez a modell csak Rendszer integrátorok és hardver vagy szoftver értékesítők számára elérhető.
- A megoldás vevőjének képesnek és hajlandónak kell lennie arra, hogy NYFSZ összetevőkkel dolgozzon. Ez a gyakorlatban azt jelenti, hogy az ügyfélnek professzionális támogatási lehetőségeket kell ajánlani a NYFSZ összetevőkre vonatkozóan, vagy magának a vállalatnak kell a teljes megoldáshoz támogatást nyújtani, beleértve a NYFSZ összetevőket is.

6 A NYFSZ hatása a szoftverpiacra

Porter öt erő modellje a piac szerkezetét kívülről befelé mutatja be.



A modell a következő erőket különbözteti meg:

- Az új belépők fenyegetése: mennyire könnyű vagy nehéz a versenytársak számára a piacra lépés?
- A beszállítók alkuereje: Mennyire erős a beszállítók pozíciója?
- A vevők alkuereje: Mennyire erős a vevők pozíciója?
- Meglévő cégek közötti versengés: Mennyire erős a verseny a piacon jelenlévő cégek között?
- Helyettesítő termékek fenyegetése: A terméket vagy szolgáltatást mennyire könnyű mással helyettesíteni?

A következőkben a NYFSZ hagyományos szoftvertermékek és szolgáltatások piacán történő felbukkanásának hatásait vizsgáljuk ennek a modellnek a felhasználásával. Azért különböztetjük meg a termék és szoftver piacokat, mert a szolgáltatások nyújtója általában nem egyezik meg a szoftvertermék értékesítőjével.

6.1 Az új belépők fenyegetése

A NYFSZ drámaian csökkenti a szoftvertermék piacra történő belépés korlátjait, mert a NYFSZ szoftver megalkotója a már létező NYFSZ-t beépítheti termékajánlatába. Erre a Red Hat az egyik példa. A Red Hat körülbelül évi 60 millió dolláros befektetéssel létre tud hozni egy

vállalkozás-kész Operációs Rendszert, mert a NYFSZ ökoszisztéma becslések szerint 1 milliárd USD fejlesztést biztosít.

A szoftverszolgáltatás piacon a NYFSZ ugyanilyen hatást fejt ki. A szoftver szabadon rendelkezésre áll és a forráskódot tanulmányozni lehet. A NYFSZ fejlesztés nyitott jellege azt is jelenti, hogy nincsenek privilegizált értékesítői információk, ami megakadályozná az új belépők szolgáltatás nyújtását.

6.2 A beszállítók alkuereje

A szoftvertermékek és szolgáltatások piacán a NYFSZ csökkenti a beszállítók erejét. A NYFSZ sok közös, szabadon elérhető szoftvert biztosít, ami korlátozza a már működő szoftverértékesítők erejét. Ha a szoftverértékesítők túl magas árakkal kezdenek dolgozni, a piaci szereplők vagy visszatérnek a NYFSZ alternatívákhoz, vagy BEFEJEZETLEN MONDAT!

6.3 A vevők alkuereje

A beszállítókra kifejtett hatás tükörképeként a NYFSZ a vevők alkuerejét erősíti. Számos nagy, a közszektorba tartozó szervezet illusztrálta ezt, amelyek a Linux-ra való áttéréssel fenyegetőztek, s így gyakoroltak nyomást a Microsoft-ra, a költségcsökkentés érdekében.

A szoftverszolgáltatási piacon a NYFSZ fejlesztési modell nyitottsága a szoftver képességeire, összetettségére és korlátaira vonatkozó információkat szabadon elérhetővé teszi. Ennek az információnak a segítségével a felhasználók pontosabban felmérhetik a szolgáltatási ajánlat értékét.

6.4 Versengés a meglévő cégek között

A szoftverszolgáltatási piacokon a NYFSZ a versenyt erősíti, mivel nagyon könnyű piacra lépni és beszállni a versenybe. A szellemi tulajdont képező szoftverek értékesítői intézkedéseik révén ellenőrizhetik, hogy kik ajánlanak a termékre vonatkozó szolgáltatásokat, azáltal, hogy Hozzáadott értékű viszonteladói partnerségeket és terjesztői hálózatokat hoznak létre. Az ESRI például egy földrajzi területen belül csak egy hivatalos terjesztőt engedélyez, és ezeknek a terjesztőknek monopol jogokat ad arra, hogy az adott területen támogatási szolgáltatásokat nyújtsanak. A NYFSZ termékek esetében a terjesztők ilyen ellenőrzése a szolgáltatási ajánlatokra vonatkozóan gyakorlatilag lehetetlen.

A szoftvertermék piacon a NYFSZ rendelkezésre állása csökkenti az olyan jelentős szellemi tulajdont képező szoftvercégek monopol erejét, mint például a Microsoft.

6.5 A helyettesítő termékek fenyegetése

A NYFSZ megjelenése egyes termékek és szolgáltatások esetében lehetővé tette esetleges helyettesítő termékek létrehozását. Ezek a helyettesítő termékek általában Saas formában jelentkeznek. A NYFSZ fontos ezen helyettesítő termékek megjelenése szempontjából, mivel alacsony költségen biztosítja a szükséges méretezhetőséget.

7 A nyílt forráskódhoz kötődő üzleti kockázatok

7.1 A licenceknek való megfelelés

A nyílt forráskód nem közkincs szoftver: licenccel jár együtt, amely kötelezettséget ró mindenkire, aki a szoftvert használja. Ha egy vállalat a nyílt forráskódra akarja üzletét alapozni, igazolnia kell, hogy a megközelítés összeegyeztethető-e minden, az általa használt nyílt szoftver licencével. Íme egy példa: X vállalat egy önkormányzat számára testreszabott szoftverfejlesztési szolgáltatásokat kínál. Rendszerint a testreszabott szoftverfejlesztést egy olyan ügyfél számára, aki a szoftvert csak a szervezeten belül használja, a GPL lehetővé teszi, mert nincs szó a szoftver terjesztéséről. De mivel a szoftvert sok ügyfele használhatná, az X vállalat más modellt alkalmaz. Előfinanszírozza a szoftverfejlesztést, és licence díjat fizettet az ügyfeleivel (az ügyfeleket szerződés kötelezi a szoftver megvásárlására). Más szóval, terjeszti a szoftverét, és hogy ezt jogszerűen tehesse a GPL kötöttségein belül, a vállalatnak is közzé kell tennie a forráskódját. Ez nyilvánvalóan tönkretenné az egész modellt. Ha a vállalat ki akarja használni a nyílt forrás előnyeit, vagy meg kell változtatnia modelljét, vagy el kell kerülnie minden, a GPL alá tartozó nyílt forráskódú szoftvert (vagy egyéb kölcsönös licenceket).

Azokban az esetekben, amikor a terméket nyílt forráskódú szoftvercsomagok keverékéből állítják elő, amelynek mind saját licenccel rendelkeznek, a licencnek való megfelelés problémája még tovább bonyolódik. Ha GPL, MPL és Apache licencünk is van, akkor melyik licence alapján lehet saját termékünket terjeszteni? Egyáltalán lehet ilyen terméket jogszerűen terjeszteni? A Szabad Szoftver Alapítvány szerint az Apache 2.0 licenc és a GPL 2.0 licenc egymással nem összeegyeztethető, így ezeket nem lehet egy terjesztésre szánt termékben kombinálni. (Az Apache Alapítvány azonban ezt a megállapítást vitatja).

7.2 Felelősség

A legtöbb nyílt forráskódú licenc tartalmazza a felelősség korlátozását, mely szerint a programot olyan állapotban kínálják, „ahogy van”, és hogy a létrehozók nem vállalnak felelősséget semmilyen, a program által okozott kárért. Európában a szolgáltatásaikhoz és termékeikhez nyílt forráskódot használó vállalatokat nem védik ilyen záradékok. Először is, mivel az európai fogyasztóvédelmi törvények felülírják az ilyen felelősség korlátozó klauzulákat, és másodsorban pedig, mert a nyílt forráskódú terméken alapuló üzleti szolgáltatás vagy termék elhatárolható a nyílt forráskódú terméktől, és így önmagában új felelősségeket vonhat maga után. Tehát mi történik, amikor egy X nyílt forráskódú csomagon alapuló terméket vagy szolgáltatást hozunk létre, és az X-ben lévő hibák vagy a rossz működés kárt okoz ügyfelünknek? Elvethetjük-e felelősségünket? Valószínűleg nem. Felelősségünk keletkezik, amikor üzleti ajánlatban használjuk fel a nyílt forráskódot, és ezt a felelősséget nem lehet a nyílt forráskódú szoftver létrehozóira hárítani.

7.3 Szabadalmak

Amikor az üzleti modell nyílt forráskódú szoftver létrehozását is magában foglalja, a legnagyobb jogi kockázatot a szabadalmakkal kapcsolatos pereskedés jelenti. A nagy szoftvercégeknek nagyon sok szabadalmuk van, ezért gyakorlatilag lehetetlen abszolút biztosnak lenni afelől, hogy a szoftver nem ütközik semmilyen szabadalomba. A szabadalmi perekben való védekezés költsége olyan magas, hogy kezdő vagy kisebb NYFSZ vállalatok nem képesek ezt a költséget magukra vállalni. A szoftver szabadalmak hatásának csökkentésére a NYFSZ közösség a NYFSZ licencekben a szabadalmakról is gondoskodik. A GPL 3. verziója például olyan rendelkezéseket tartalmaz, hogy az engedélyezőnek minden olyan szabadalom licencet biztosítania kell a felhasználóknak, amely a szoftver futtatásához, módosításához vagy terjesztéséhez szükséges. Ezen túlmenően, a szabadalom tulajdonosokat arra kérik, hogy ne támasszanak szabadalmi követeléseket a GPL más felhasználóival szemben, különben az összes GPL licence automatikusan érvénytelenítésre kerül. Ma még bizonytalan, hogy ezek az óvintézkedések mennyire fognak hasznosnak

bizonyulni. A tapasztalatok azt mutatják, hogy a NYFSZ felhasználóknak kevés félnivalójuk van a szabadalmi licencektől. Azért a NYFSZ alkotóknak meg kell vizsgálniuk azokat a szabadalmakat, amelyek hatással lehetnek a projektre. Figyelembe véve a szoftver szabadalmak számát, bonyolultságát és időnként homályosságát, remény sincs arra, hogy az ilyen jellegű kutatás meggyőző erejű lehet. Tőlünk telhetően mégis törekedjünk a vonatkozó szabadalmak beazonosítására. Minden beazonosított szabadalom, amely korlátozhatja a szoftvert, később körül tervezhető. A jelenlegi körülmények között ez a megközelítés a legjobb, amit remélhetünk.

8 Nyílt forráskódú üzleti modellek a gyakorlatban

Jelenleg az üzleti érték teremtése szempontjából a legfontosabb nyílt forráskódú modellek a Támogatásértékesítő és a Platformszolgáltató modellek. Mindként modell magas nyereség kulccsal kecsegtet, mert szabvány terméket ajánlanak rögzített áron. Ezek a modellek lehetővé teszik, hogy a NYFSZ projekt műszaki irányítása jövedelemgeneráló ajánlattá alakuljon. Emiatt a nyílt forráskódú piac vezető vállalatai ezeket a modelleket és variánsaikat használják alapul. Ezek között van a MySQL AB, az Alfresco, a Mulesource, a Red Hat, a Canonical és a Pentaho.

A támogatásértékesítő és a platformszolgáltató modellek természetes üzleti modellek a szoftver piacon, mert a NYFSZ projektek közösségi fókuszát kellően tiszteletben tartják. A közösségi hajtóerő a projekt műszaki fejlesztése szempontjából nagyon fontos: sok felhasználó, hozzájáruló és fejlesztő működik együtt olyan jó szoftverek létrehozásában, amelyek a felhasználók valós igényeire adnak választ. A NYFSZ *vállalati* felhasználói számára ugyanakkor kényelmetlen, hogy egy közösséggel kell bíbelődniük annak érdekében, hogy szoftvertámogatási szolgáltatásokhoz férjenek hozzá. Számukra kedvezőbb olyan vállalatközi kapcsolatok kiépítése, amely elszámolhatóságot, stabilitást és a támogatás megbízhatóságát biztosítja számukra. A Támogatásértékesítők és a Platformszolgáltatók azok az üzleti szervezetek, akikkel a vállalati felhasználók ilyen kapcsolatot tudnak kiépíteni. Szintén emiatt van az, hogy olyan vállalatok, mint a Spring21, az Alfresco és a Mulesource számos Global 2000 vállalatot számlálhatnak ügyfeleik között: az ilyen nagy, gyakran multinacionális cégek megértik a NYFSZ szoftver értékét, de szerződésben garantált, megbízható cégektől származó támogatásra tartanak igényt. (A dolog másik oldala pedig, hogy javarészt a nagyvállalatok alkotják a támogatásértékesítők piacát. A kis, mozgékony és rugalmas KKV-k számára nem olyan problémás a formális támogatási szerződések hiánya, és így nem is lehet őket könnyen rávenni arra, hogy ilyeneket vegyenek.).

Érdekes módon ezeknek a Támogatásértékesítőknek és Platformszolgáltatóknak az értéke nem elsősorban szellemi tulajdonukban, hanem sokkal inkább a márkanévben rejlik. Ez akkor vált világossá, amikor a múlt évben közzétették az Oracle Linux ajánlatot. Az Oracle elkötelezte magát, hogy létrehoz egy új Linux változatot, ami ugyanaz, mint a Red Hat terjesztés, csak alacsonyabb költségekkel. Az Oracle szerencsétlenségére az ajánlatuknak nagyon korlátozott sikere volt csak a piacon. Úgy tűnik, azért, mert az ajánlat még alacsonyabb áron sem túl hiteles. Senki sem tud jobb Red Hat Szervert létrehozni, mint maga a Red Hat. A vevők azért értékelik a Red Hat-et és termékeit, mert már bebizonyította, hogy képes jól integrált, tesztelt és teljes támogatással ellátott vállalkozás-kész operációs rendszert előállítani – ez az, amit a Red Hat márkanév képvisel. Az Oracle eddig még nem tudta meggyőzni a piacot arról, hogy ők ugyanolyan értéket és megbízhatóságot tudnak nyújtani, mint a Red Hat. A nyílt forráskódú vállalatok számára a márka értéke nagy jelentőséggel bír. Először is, egy projekt körül a közösség műszaki vezetése nagyon fontos, mert ez segít a márkaépítésben és segíti az ügyfelek felé a hitelesség megőrzését. A második következtetés az, hogy a Külső támogatásértékesítő modell nem lehet üzletileg fenntartható, ha létezik a projekt feletti vezetést fenntartó hiteles Támogatásértékesítő.

Noha a legérdekesebb nyílt forráskódú üzleti modell a Támogatásértékesítő vagy a Platformszolgáltató modell, a legszélesebb körben mégis a Tanácsadási modellt alkalmazzák. Ezt jól illusztrálja az a nagyszámú tanácsadó cég, amelyik a Linux-hoz kapcsolódó tanácsadó szolgáltatást kínál. A NYFSZ nyitottsága sok kis vállalkozó számára teszi lehetővé, hogy önálló szoftverforgalmazó vagy partner státusra vonatkozó tárgyalások nélkül, nagyon kis befektetéssel is el tudja kezdeni a szolgáltatást. A modell, mondhatni, mindenki számára elérhető, akinek van egy laptopja és internet hozzáférése. Az ilyen tanácsadás egyfajta ökoszisztémát hoz létre a NYFSZ projekt körül. A NYFSZ-re alapuló tanácsadások olyan szolgáltatásokat ajánlanak, amelyek a Támogatásértékesítő és a közösség által nyújtott szolgáltatások között foglalnak helyet. Olyan szakmai támogatási szolgáltatást tudnak biztosítani, amelyek pontosan a helyi piacra és ügyfelekre szabottak, de nem fogják a Támogatásértékesítőkhez hasonló magas árrést vagy globális kiterjedést elérni.

9 Alkalmazás a térinformatikai piacon

A térinformatikai piac egy piaci rés a szoftverpiacon. Nagy termék komplexitás és olyan felhasználói bázis jellemzi, amely erősen koncentrálódik a közszférában és a tudományos közösségben. Másrészről viszont nagy piaci erő van szoftvercégek kis csoportjának a kezében, ahol az ESRI egyértelműen piacvezető.

Ha megvizsgáljuk a NYFSZ vállalatokat ezen a piacon, nem találunk egyértelmű nyerteseket – olyan vállalatokat, amelyek a JBoss-hoz vagy a Red Hat-hez mérhetőek. Azok a cégek, amelyeket a leghamarabb a térinformatikai nyílt forráskóddal asszociálunk, mint a Refractions, a Vivid Solutions vagy a Lat/lon, KKV-k, és olyan üzleti modellt vettek át, ami gyakorlatilag megegyezik a Tanácsadási modellel (támogatási és fejlesztési szolgáltatások ad-hoc alapon). Ez a modell korlátozza az általuk elérhető növekedést, és azt is jelenti, hogy termékeik (mint a PostGIS vagy a JTS) népszerűségét sem hasznosíthatják saját bázisuktól távol eső területeken (Európa, Ázsia).

Mivel lehet magyarázni a Platformszolgáltatók és a (külső) Támogatásértékesítők hiányát a térinformatikai piacon? A közszektor – olyan nagy ügyfelek, akik értékelik a szerződéses jogviszony megbízhatóságát és stabilitását – által uralt piac a támogatásértékesítők megjelenése számára ideális. De akkor miért nem bukkant még fel egy ilyen cég sem?

Az első ok az, hogy az értékesítői kötöttségek a GIS területén olyanok, hogy (különösen) a nagy GIS felhasználók számára nagyon nehéz a NYFSZ GIS-re átváltani. Persze nem csak erről van szó, mert most fel kell tennünk a kérdést: Miért olyan nehéz átváltani a nyílt forráskódra? Lehetséges, hogy a GIS NYFSZ közösségnek nem sikerült olyan szoftver készletet biztosítani, ami összemérhető az üzleti szoftver készletekkel? Szerintem az irodai GIS-szel, a magas minőségű térképészeti és adatszerkesztő eszközökkel ez a helyzet. Az olyan eszközök, mint például a GRASS és a QGIS gazdag térelemzést és könnyen használható GIS megjelenítést biztosítanak, de nem közelítik meg az ArcGIS teljes gazdagságát mint térelemző, térkép előállító és adatszerkesztő eszköz. Más funkcionális területeken (API-k és könyvtárak programozása, téradatbázisok és vebtérképezés) a NYFSZ ajánlatok összemérhetőek vagy talán még jobbak is, mint a zárt kódú ajánlatok. Sajnos sok ügyfélnek irodai GIS-re, adatszerkesztő és magas minőségű térképészeti szoftverre van szüksége. Ők üzleti GIS forgalmazókhoz fordulnak az ilyen fajta szoftverekért, a GIS értékesítők pedig megpróbálják úgy kihasználni pozíciójukat, hogy maximálisan befolyásolják az ügyfél többi GIS infrastruktúráját is. (Annak, hogy az ArcGIS Szerkesztőnek földrajzi adatbázisra van szüksége, valószínűleg több köze van az ESRI azon vágyához, hogy a szellemi tulajdont képező formát alkalmazza az adatbázisra, mint a műszaki előírásoknak). Tehát ameddig nem létezik megbízható NYFSZ alternatíva az irodai GIS-hez, térképezéshez és adatszerkesztéshez, addig a GIS-hez nem tudnak illeszkedő NYFSZ szoftver rendszert biztosítani. Nézőpontom szerint ezt gyakorlatilag eleve kizár olyan vállalatokat a GIS-ből, mint a Red Hat.

Betűszavak és rövidítések

BI	Business Intelligence	Üzleti Intelligencia
BSD	Berkeley Software Distribution	Berkeley szoftverterjesztés
CPL	Common Public License	(egy licenc szerződés elnevezése)
CRM	Customer Relationship Management	ügyfélkapcsolat menedzsment
ERP	Enterprise Resource Planning	vállalkozói erőforrás tervezés
F/OSS	Free/Open Source Software	ingyenes/nyílt forráskódú szoftver (NYFSZ)
FSF	Free Software Foundation	Szabad Szoftver Alapítvány
GPL	GNU General Public License	(egy licenc szerződés elnevezése)
IDE	Integrated Development Environment	Integrált fejlesztési környezet
LGPL	Library (or Lesser) GNU General Public License	(egy licenc szerződés elnevezése)
MPL	Mozilla Public License	(egy licenc szerződés elnevezése)
OSI	Open Source Initiative	Nyílt Forráskód Kezdeményezés
OSD	Open Source Definition	Nyílt forráskódú szoftverek definíciója
SaaS	Software as a Service	Szoftver mint szolgáltatás
SLA	Service Level Agreement	Szolgáltatási Szint Megállapodás
SME	Small to Medium Enterprise	Kis- és közepes méretű vállalkozás (KKV)
UML	Universal Modelling Language	Univerzális Modellező Nyelv

Irodalomjegyzék

Baldwin, C. Y. and Clark K. B. Does Code Architecture Mitigate Free-Riding in the Open Source Development Model? Harvard Business School Working Paper, elérhető a http://www.people.hbs.edu/cbaldwin/DR2/BaldwinClark.ArchOS.Jun03.pdf internet címen, 2003.

Behlendorf B. Open Source as a Business Strategy. In DiBona, C., Ockman, S. and Stone M. (eds.), *Open Sources: Voices from the Open Source Revolution*, pp. 113-126. O'Reilly & Associates, Sebastopol, 1999.

Berlecon Research Gmbh. Basics of Open Source Software Markets and Business Models – FLOSS Final Report, elérhető a http://www.infonomics.nl/FLOSS/report internet címen, 2002.

Bessen, j. Free Provision of Complex Public Goods. In Open Source Software: Do Firms Practise What they Preach? In Bitzer, J. and Schröder Ph. (eds.): *The Economics of Open Source Software Development,* pp. 83-109. Elsevier, Amsterdam, 2006.

Daffara D. Business models in FLOS-based companies. Working paper, elérhető a http://opensource.mit.edu/papers/OSSEMP07-daffara.pdf internet címen, 2007

Fogel K. *Producing Open Source Software.* O'Reilly & Associates, Sebastopol, 2006.

Gosh, R. A. and Prakash, V. V. The orbiten free software survey. *First Monday,* 5(7), 2000.

Gosh, R. A., Glott, R., Kreiger B. and Robles, G. The free/libre and open source software developers survey and study – FLOSS Final Report, elérhető a http://www.infonomics.nl/FLOSS/report internet címen, 2002.

Healy, K. and Schussman, A. The Ecology of Open Source Software Development. Working Paper, elérhető a http://opensource.mit.edu/papers/healyschussman.pdf internet címen, 2003.

Hecker, F. Setting up shop: The business of open-source software. *IEEE Software*, 16(1): 45-55, 1999.

Hunt, F. and Johnson, P. On the Pareto distribution of sourceforge projects. In *Proceedings of the Open Source Software Development Workshop*, pp. 122-129, Newcastle, UK, 2002.

Koenig, J. Seven open source business strategies for competitive advantage. Elérhető a http://www.itmanagersjournal.com/feature/314 internet címen, 2004.

Krishnamurthy, S. An Analysis of Open Source Business Models. In Feller, F., Fitzgerald, B., Hissam, S. A. and Lakhani K. R. (eds.): *Perspectives on Free and Open Source Software,* pp.279-296. MIT Press, Cambridge USA, 2005.

Lakhani, R., Wolf, R.G. Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. In Feller, F., Fitzgerald, B., Hissam, S. A. and Lakhani K. R. (eds.): *Perspectives on Free and Open Source Software,* pp.279-296. MIT Press, Cambridge USA, 2005.

Lerner, J., Tirole, J. Some simple economics of open source. *Journal of Industrial Economics,* 50 (2): 197-234, 2002.

Mockus, A., Fielding, R.T. & Herbsleb, J. A case study of open source software development: the Apache server. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, pp. 263-272, Limerick, Ireland.

Olsen M. Dual Licensing. In DiBona, C., Cooper D. and Stone M. (eds). *Open Sources 2.0: The Continuing Evolution*, pp. 71-90. O'Reilly & Associates, Sebastopol, 2006.

Osterwalder, A. The Business Model Ontology. PhD Thesis, Université de Lausanne, 2004.

Raymond, E. S. The Cathedral and the Bazaar. *First Monday*, 3(3), 1998.

Rosen, L. *Open Source Licensing.* Prentice Hall, Upper Saddle River, 2004.

Rossi, C. Bonaccorsi, A. Contributing to the common pool resources in Open Source software. A comparison between individuals and firms. Working Paper, Sant'Anna School of Advanced Studies Institute for Informatics and Telematics (IIT-CNR), Pisa, Italy, 2004.

Rossi, C., Bonaccorsi, A. Intrinsic Motivations and Profit-Oriented Firms. In Open Source Software: Do Firms Practise What they Preach? In Bitzer, J. and Schröder Ph. (eds.): *The Economics of Open Source Software Development,* pp. 83-109. Elsevier, Amsterdam, 2006.

Scott, B. The Open Source Legal Landscape. Elérhető a http://opensourcelaw.biz/publications/papers/BScott_OSS_Legal_Landscape_060328.pdf internet címen, 2006.

Stähler, Business Models as a Unit of Analysis for Strategizing. Paper presented at the International Workshop on Business Models, Lausanne, Switzerland, 2002.

Stallman, R. What is copyleft? Elérhető a http://www.gnu.org/copyleft/copyleft.html internet címen, 1984.

Young, R. Giving it Away: How Red Hat software stumbled on across a new economic model and helped improve an industry. In DiBona, C., Ockman, S. and Stone M. (eds.), *Open Sources: Voices from the Open Source Revolution*, pp. 113-126. O'Reilly & Associates, Sebastopol, 1999.