Kodolányi János Egyetem
Gazdálkodási és Menedzsment Tanszék

A Szoftverfejlesztés Gazdaságtana

Konzulens:
Dr. Pitlik László
Informatika Tanszék

Készítette:
Györgydeák Balázs Péter
Gazdálkodási és menedzsment alapszak

Székesfehérvár, 2021

TARTALOMJEGYZÉK

TÁBLÁZATOK JEGYZÉKE ÉS ÁBRÁK JEGYZÉKE

1 BEVEZETÉS

1.1 Problémafelvetés, hipotézisek

A szoftver fejlesztés tervezesének gazdasági vetülete napjainkban még mindig ún. „gyerekcipőben" jár, noha az információs többletérték fogalma, szinte a kezdetektől létezik, ezért kifejezetten fontosnak érzem a téma pontosabb vizsgálatát (vö. 2.8, 3.2.5 és 3.3.5). Ennek elsődleges oka tapasztalataim szerint, hogy az Információs Technológiák (IT) jelentik pl. a legnagyobb pénzügyi vállalatok (melyek közül egy biztosítót egy középvállalat mellett dolgozatomban külön is elemzek) fő szolgáltatását, ahol a felhasználók online intézik ügyeiket, a belső rendszerek segítenek életben tartani a teljes céget. Mindemellett a COVID-19 pandémia okán hozott megszorítások mellett egy olyan munkavállalói környezetben szükséges szakértőket választaniuk, ahol vélhetően a legnagyobb a munkaerőhiány, azonban általában csak az „előre menekülés" stratégiája tarthatja őket életben.

Mivel manapság az IT szinte minden vállalati működésen belüli szakterület napi munkájára hatással van, legyen az marketing, döntéshozatal, gazdaság, HR, stb, így a szoftverfejlesztés gazdaságosságának vizsgálata a cégek számára elengedhetetlen (vö. 2.7, 3.2.4, 3.3.3-as pont). Azonban a vállalatok az informatikai változtatások esetében általában csak azok gazdasági-kompetitív előnyeire és nyereség-költség alapú vetületeivel foglalkoznak, azonban nem veszik számításba pl. az ezzel járó megfelelősséget (vö. Táblázat 5.: Vállatok kiértékelési táblájának „Nem megfelelő" oszlopa), mint az aktuális szolgáltatások minőségének változása, illetve csak és kizárólag a szoftvert fejlesztő [1] kódolók erőforrásainak bekerülési költségével számolnak, noha más tételek is léteznek, például a tervezés során igénybe vett Architect kollégák, Üzleti Elemzők vagy szolgáltatási szakértők munkaideje amelyekre az elemzések során (Táblázat 5.) is kitérek.

Dolgozatom célja, hogy a teljes szoftverfejlesztési folyamatot egy minél inkább mérhető, számszerűsíthető és szigorúan előre tervezhető egységes információs többletérték-becslő rendszerként mutassam be, mindamellett, hogy megvizsgálom, az

[1] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Developer

egyes quasi véletlenszerűen kiválasztott vállalatok, hogyan végzik ezen feladatokat és mennyire állnak készen a jövő hasonló kihívásaira. A szoftverfejlesztés folyamatának ezen újszerű, holisztikus megközelítésekor saját évtizedes személyes szakértői tapasztalataimra és természetesen a kapcsolódó szakirodalomra támaszkodom. A dolgozat ezáltal egyrészt a céges IT-stratégiák megalkotását támogató dokumentumként, másrészt a knuth-i elvek felé való elmozdulás lehetőségeit feltárni akaró kutatásként is felfogható, ahol a knuth-i elvárás egyszerű: tudás csak az, ami forráskódba átírható! Így a stratégia-alkotás absztrakciós szintje is ezen az elváráson keresztül vizsgálandó.

A dolgozat titkosításának indokai a következők:

- A vizsgált cégek esetében, mivel a folyamatokat sok esetben testre szabják a munkatársak vagy a partnerek felismerhetik az adott vállalatot, ami akár versenyhátrányt is okozhat számukra.
- Mivel az elemzések versenyelőnyt is jelenteni képes kritikákat/javaslatokat is tartalmaznak, amely a szerző magánvéleménye is, ezért azt nem kívánja megosztani más vállalatokkal és a piac többi szereplőjével sem.

1.2 Kutatási cél

Knuth szerint, ha „Mindent optimalizálsz, mindig boldogtalan leszel.” [2] Véleményem szerint ez azt jelenti, hogy mindig van gyorsabb, újabb, értékesebb, tehát mindig van lehetőségünk egy adott helyzetből, projektből többet kihozni. Kutatási célom tehát alapvetően két kérdés megválaszolására koncentrál:

- Mit is jelent valójában az információs többletérték? Miért van az, hogy egy információ valamely időpillanatban, egy piaci szereplőnek, míg egy másiknak kevesebb, mik azok a körülmények, amiket szükséges figyelembe vennünk ennek meghatározásakor?
- Hogyan alakítsuk a gazdaságossági számításaink egy innovációs folyamat során az információs többletérték függvényében, mire érdemes odafigyelni egy projekt tervezésekor és végrehajtásakor, végig szem előtt tartva a célként meghatározott érték létrehozását?

[2] Knuth Ervin Donald, 1981, 15.

Kutatásom során quasi szúrópróba-szerűen egy név nélküli multinacionális és egy középvállalat működését kívánom bemutatni, amelyek mind belső, mind külső fejlesztőkkel, projektmegoldásokkal dolgoznak, valamint egy segítségével a hazai IT szcéna ún. általános (de értelemszerűen nem reprezentatív) véleményét fogom elemezni, a két vizsgált cég munkavállalóinak reakcióival kiegészítve. A két (eltérő jellegű és piaci szegmenst reprezentáló) cég kiválasztása azért elegendő, mert a dolgozattal olyan (pl. a knuth-i elvet támogató) jelenségekre kívánok rámutatni, melyek egyszeri előfordulása is pl. best practice-ként, ill. ennek ellenkezőjeként értelmezhető, vagyis a statisztikai sokaságnak ilyen esetekben nincs értelme.

Célom mindezzel, hogy rávilágítsak az általános hiányosságokra, hogy ezek milyen bevétel kiesést, vagy plusz költséget jelentenek a vállalatok számára, valamint arra, hogyan rombolja a munkatársak motivációját és teljesítményét az egyes anomáliák bármelyike. Mindezek után, természetesen az eredmények fényében, megoldási javaslatokat kívánok felvázolni arra, miként lehet komplexebb módon kezelni a szoftverfejlesztés többletértéktermelő képességét, az e területen fellépő kockázatokat.

1.3 Kutatási módszerek

Kutatási módszertanom ötvözi az induktív és a deduktív módszereket, mivel egyrészt építkezem a szakirodalom alapjaira, amely részletesen taglalja a szervezési, tervezési, és szolgáltatás átadás-átvételi problémák egyes általános okait és következményeit, azonban sok helyen személyes tapasztalatokra, illetve általam hozzáférhető belső biztosítói és flottakezelői adatokra is támaszkodom, amelyet a kutatás során használok fel, ez az alapja a titkos kezelésre vonatkozó kérelmemnek.

A kutatás eredményeinek kidolgozásához az adatokat szekunder (vö. 8. Fejezet: Melléklet: Szekunder adatok katalógusa), módon gyűjtöttem, erre lehetőséget a munkáltatóm biztosított, amely nemzetközi szinten piacvezető gépjármű szoftvertechnikai adatbeszállító 54 márka, 3800 modellének adatát tartalmazza, több, mint 240.000 felszereltségi változat pontos meghatározásával, legyen az motortípus vagy méret, esetleg digitális klíma, valamint az ezek eredményeképp generált 9 millió alkatrészt és hozzátartozó cikkszámot és árat, gyári munka normaidőt és kiegészítő

adatokat. Ez a nagyvállalatot átfogó elemzésem során azt jelentette, hogy az információs többletérték meghatározásánál (amely tulajdonképpen a biztosító számára tervezhető költség- illetve kárkifizetés csökkentés) a 2019-es év adatait használtam dolgozatomban, 328.192 darab káresemény elemzésével, felhasználva a károsult gépjárművek javításához szükséges fényezési, munka és alkatrész összegeket, míg a középvállalatnál, amely egy hazai márkaszervíz, a várható plusz bevétel esetén az elmúlt 4 év 84.000 db, az adatbázisunkban található szervizesemény a tételeivel számoltam.

2 A VIZSGÁLAT SZAKIRODALMI HÁTTERE

Dolgozatom hátterét és motivációját az elmúlt 15 év információs technológiák (IT) szektorban töltött tapasztaltom adta, amelyből 5 évet a székesfehérvári, 2 évet a Helsinki-i IBM-ben (International Business Machines) töltöttem. Az első helyszínen, mint szoftverüzemeltetés csoportvezető, míg a másodikon projekt menedzserként, aki a legnagyobb észak-európai szerverterem felépítéséért felelt, mind technológiai, mind üzleti oldalon. Ezután 8 évet töltöttem egy gépjárműinformatikai vállalat szoftverfejlesztési és szolgáltatási vezetőjeként, amely biztosítóknak, szakértő irodáknak és autójavítóknak kínált kárrendezési és szervizeltetési megoldásokat. Ezen munkakörök betöltésekor szinte minden esetben az üzlet és a technológia közötti kommunikációt éreztem a legnagyobb hibajelenségnek, valamint a fejlesztés előkészítési fázisainak nem megfelelő koordinálását. Dolgozatomban ezekre a jelenség/problémák kapcsán próbálok kézzel fogható megoldási javaslatokat bemutatni, amelyek minden hasonló vállalat/intézmény dolgozói számára segítséget nyújthatnak.

Az elméleti háttér értelmezéséhez fontos előzetesen jellemeznünk a bemutatásra kerülő vállalatok alapvető karakterisztikáit, amelyet a „Táblázat 1.”-ben foglaltam össze, mivel ezen pontokra külön kitérek azok leírásakor:

Vállalat profilja	Vezető Biztosító	Gépjármű Márkaszervíz
Vállalatméret	130.000 munkavállaló	320 munkavállaló
Tulajdonos típus	Részvényesek - Tőzsde	KFT – 3 fő
Főtevékenység	Pénzügyi szolgáltatás	Gépjármű szolgáltatás
Tevékenység diverzifikálása	Diverzifikált – Utas-, élet-, nem-életbiztosítás, egyéb pénzügyi tevékenység	Diverzifikált – Javítás, szervíz, viszonteladás, flottakezelés
Változásokhoz való viszony	Lomha reagálás a változásokra, Startup felvásárlás	Trendkövető magatartás, nem elsődleges innovációs labor
Vállalat teljesítmény	Stabil – Nem növekvő (Koronavírus)	Stabil – Enyhe visszaesés (Koronavírus)
Piaci célok	Piaci részesedés növelés	Piaci részesedés megtartása, új piacok kiaknázása

Táblázat 1.: A két esettanulmány alap paraméterei (Forrás: Munkahelyi partnerek által megadott információk)

2.1 Szoftverfejlesztési metodológiák ismertetése és kritikája

Az alábbiakban részletesen kifejtem az egyes szoftverfejlesztési módszertanokat, azok előnyeit és hátrányait, valamint fontosabb vívmányait.

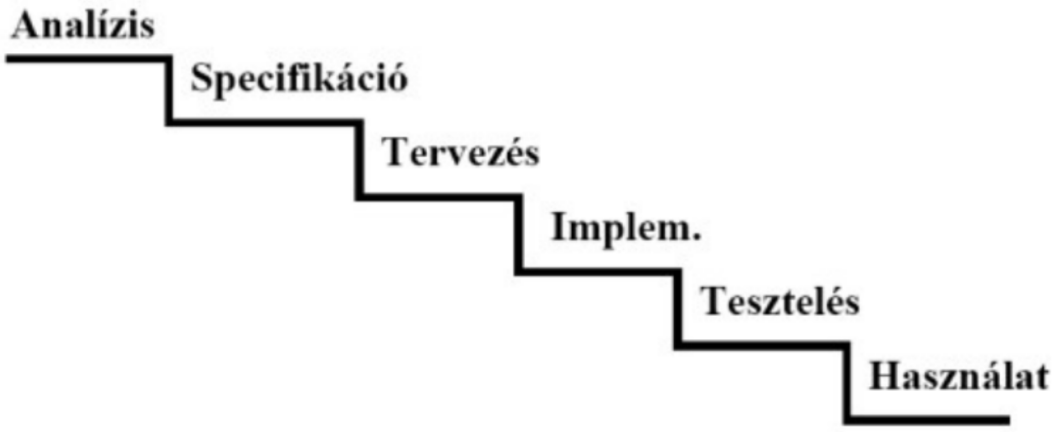
A fejezet célja a dolgozat elemzéseinek megalapozása. Ezen módszertanok elméleti hátterének megismerése azért fontos, mert az elemzés során egyértelműen kimutatható lesz, hogy a két kiválasztott vállalatnak milyen hátrányt és/vagy előnyt jelent a használatuk, mik azok a pontok, amelyek az esetükben testreszabást igényelnek.

A knuth-i elvekkel, vagyis az automatizálhatósággal való összevetés minden kulcsszó, jelenség, technológia, módszertan stb. esetén érzékelteti, milyen mértékben járul hozzá a szóban forgó gondolatvilág a hatékonysághoz, az információs többletérték-termeléshez. Minden olyan részjelenség, mely (még) nem automatizálható, jelentős hatékonysági kockázatként értelmezendő. S minden automatizálási lehetőség kapcsán felmerül a kérdés: megéri-e adott időhorizonton a fejlesztést végrehajtani (vö. vélelmezhető-e információs többletérték)?

2.1.1 Waterfall

A Waterfall[3] metodológia lényege (Ábra 2.), hogy a bejövő szoftverfejlesztési kérelmeket egységes rendszerben kezeli, azokat egy folyamat részeként, FIFO (First In, First Out) sorrendben hajtja végre. A folyamatot először D. Benington mutatta be 1956. június 29-én A „Symposium Advanced Programming Methods for Digital Computers” -en (Tudományos Tanácskozás a Fejlett Programozási Metódusokról a Digitális Számítógépekről) keretében. Később ezt a legtöbb kormányhivatal és nagyvállalat is átvette.

[3] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Waterfall



Ábra 2.: Waterfall modell – (Forrás: Wikipedia)

A metódus 6 lépcsőre osztja a fejlesztést

1. Analízis: Ebben a lépcsőben áttekintik a változtatási igényeket, meghatározzák azokat a célokat, amiket el kívánnak érni a fejlesztés során.
2. Specifikáció: Ahhoz, hogy a korábban meghatározott célokat elérjük szükséges a jelenlegi rendszer és az új funkciók[4] pontos megértése és elemzése.
3. Tervezés: Előzetes tervek kialakítása fejlesztésre, az architektúra pontos, előzetes kialakítása.
4. Implementáció: A szoftverkód megírása, amely a meghatározott célt eléri és követi a korábban kialakított dizájnt.
5. Tesztelés [5] : Az elvégzett változtatások tesztelése, megfelelő működés ellenőrzése.
6. Használat: A változtatások átadása az üzemeltetésnek, aki a továbbiakban karbantartja azt.

Előnyei minden vállalattípus esetében:

- Amennyiben az elemzés valóban pontos, már az előzetes elemzéseknél kiderülnek azok a hibák, amik sok esetben csak később merülnének fel.

[4] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Feature
[5] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Test scenario

- Nagy előnye, hogy minden esetben teljes mértékű dokumentáció készül a szolgáltatásról és annak terveiről.
- Amennyiben teljesen egyértelmű és nem változik a fejlesztés célja, még mindig nagyon jól működő rendszer.

Hátrányai minden vállalattípus esetében:

- Sajnos az esetek jelentős többségében a fejlesztés célja a megrendelő igényeinek folyamatos változása miatt nem fix, így sok esetben egyes pontokat újra felül kell vizsgálni.
- A dizájnerek, akik előkészítik a projektet, általában nem rendelkeznek a megfelelő információval a teljes megrendelői igénnyel kapcsolatban.
- A Waterfall nem alkalmas a legújabb technológiákban való fejlesztések követésére és irányítására, ahol egy-egy projektben több, mint 10 fejlesztő vesz részt, egy architektúrában pedig akár több százan is dolgozhatnak.
- A metódus nem alkalmas arra, hogy folyamatos fejlesztésű rendszereket üzemeltessünk benne, csak egy fix projekt végrehajtására.
- A Waterfall nem veszi figyelembe az egyes fejlesztések gazdaságossági hasznosságát, végül ez vezetett az Agile/Scrum alapú szoftverfejlesztés létrehozásához.

A knuth-i elvárások teljesíthetősége – vagyis az automatizálhatóság rétegei:

- A Waterfall rendszer elsődleges feladata az aktuális projektek leszállítása
- Nem holisztikus szemléletmódot használ a fejlesztői oldalon, így nem jellemző rá az automatizálás

2.1.2 Scrum/Agile[6]

„Successful use of Scrum depends on people becoming more proficient in living five values: Commitment, Focus, Openness, Respect, and Courage. The Scrum Team commits to achieving its goals and to supporting each other.” [7], ami magyarul azt fejezi ki, hogy a Scrum módszertan használatának sikere azon múlik, hogy a kollégák az öt alapérték szerint végzik-e munkájukat: elkötelezettség, fókusz, nyitottság, tisztelet és

[6] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Scrum/Agile
[7] Ken Schwaber & Jeff Sutherland, 2020, 4. – Saját fordítás

bátorság. A Scrum csapat tagjai mind elkötelezettek a közös cél elérése iránt és mindezt egymás támogatásával, közösen érik el.

Az Agile és a Scrum a legújabb szoftverfejlesztési módszertannak számít (2001). Maga az Agile egy úgynevezett módszercsoport, amely keretében a követelményeket és szolgáltatás szállításokat/megoldásokat önszerveződő csapatok együttesen intézik. Jellemzői a folyamatos fejlődés elősegítése, a megrendelői igényekhez való állandó alkalmazkodás képessége, a gyorsaság és a rugalmasság. Magát az Agilis metódus először „Agilis Szoftverfejlesztési Kiáltvány" -ban jelent meg, amelyet 2001. februárjában 17 szakértő jelentetett meg. Fontos azonban megemlíteni benne James Martin és James Kerr nevét, akik a 80-as évektől már hasonló rendszereket dolgoztak a DuPont vállalatnál.

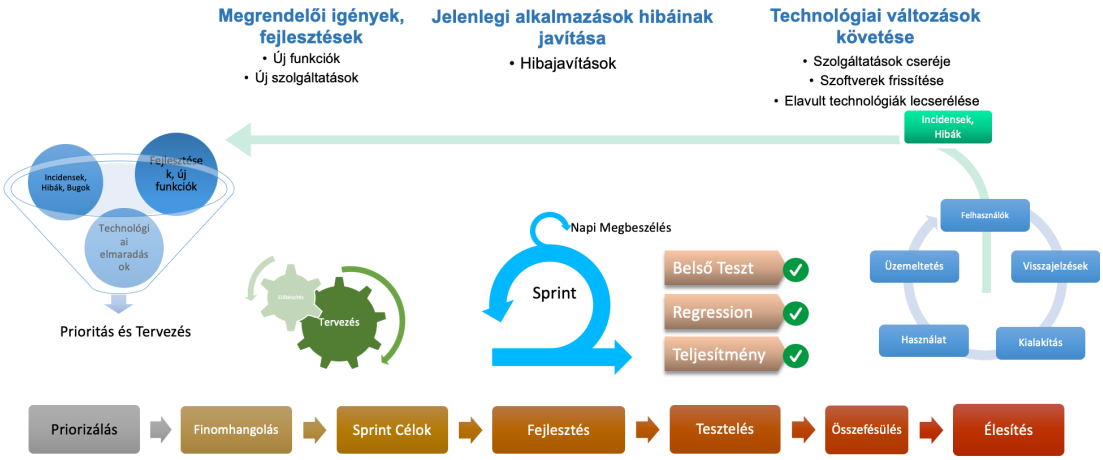
Az Agile módszertan egyik leágazása a Scrum, amelyet kifejezetten szoftverfejlesztésre találtak ki, illetve fontos része az ún. Commitment[8], ami azt jelzi az üzlet felé, hogy egy adott funkció mikorra kerül leszállításra. A Scrum rendszer szigorú határidőkkel és naptárrendszerrel dolgozik, amit a Scrum csapat minden tagja köteles követni és ami minden 2-4 hetes Sprintet [9]– fejlesztési ütemet – meghatároz.

A Scrum csapatban betöltött szerepek:

- Product Owner [10] : Elsődleges feladata a Megrendelő képviselete, aki a fejlesztői munkákat behozza a Scrum csapatban, ott egyezteti a fejlesztőkkel a pontos igényeket és kiosztja a feladatokat.
- Scrum Master[11]: A Scrum módszertan betartásáért és betartatásáért felel, a határidőket figyeli és fejleszti a csapattagokat, illetve a közös munkát. Szerepe nagyon sokrétű, ha kell menedzser, támogató munkatárs, mentor, kommunikációs tanácsadó, változás kezelő stb.
- Fejlesztő: A fejlesztők feladata kizárólag a kódolás és a funkciók, szolgáltatások előre meghatározott tervek szerinti leszállítása. A csapat önszerveződő, tehát az egyes kisebb feladatokat egymás közt osztják szét és folyamatosan tanulnak egymástól.

[8] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Commitment
[9] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Sprint
[10] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Product Owner
[11] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Scrum Master

Folyamatát jól összefoglalja az alábbi Ábra 3.:



Ábra 3.: Scrum folyamatábra – (Forrás: Scrum Page – saját fordítás és munka)

Jelmagyarázat:

- Színek jelentése:
  - Kék: Az üzleti, valamint a felhasználói visszajelzések eredményei, amelyek a fejlesztési folyamat előkészületei.
  - Zöld: A fejlesztők és az üzlet közös kommunikációnak eredménye.
  - Szürke – vörös átmenet: A szállítás folyamata.

A folyamat rövid összefoglalója:

1. Priorizálás: A vállalat egységes prioritást épít fel a gazdasági igényeinek megfelelően, azonban fontos változás, hogy ez az ún. Sprint[12] hosszának függvényében változhat, mivel azok előtt újabb sorrend alakulhat ki, a Sprint végére pedig a korábban vállaltakat a fejlesztőknek le kell szállítani. Így tud megvalósulni, hogy a vég számára legfontosabb új funkciók és feladatok mindig a lehető leghamarabb kerüljenek megoldásra, a „CI/CD[13]" folyamatos fejlődés, folyamatos szállítás jegyében. Az új funkciók fejlesztésére a vállalat IT erőforrásainak csak bizonyos részét foglalják le,

[12] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Sprint
[13] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: CI/CD

a maradék a fenti sávban megjelenített jelzett hibajavításokra és technológiai változások követésére van elhatárolva. A fontossági sorrend kialakításáért az IT-t és az üzleti oldalt összekötő kollégák felelnek, jellemzően a Projekt Menedzserek vagy a Solution Delivery Manager[14].

2. Finomhangolás: A finomhangolás során meghatározzák bejövő igények erőforrás igényeit, amelyeket ún. mandays[15]-ekre (fejlesztők által egy napi időszükséglet) osztanak fel, majd az előre meghatározott fejlesztőkollégáknak kiosztják azt, attól függően, hogy az adott Sprintben mennyi allokálható idejük van. Ez a lépés már teljesen a Scrum csapaton belül történik és a Scrum Master, valamint a Fejlesztési Architect [16]dolgozzák ki minden pontját.
3. Sprint Célok: Az adott Sprintre felkészülve a Scrum csapat meghatározza az elsődleges stratégiai célokat, legyen az egy új funkció átadása, vagy belső verseny az összes aktuális hiba kijavítására, vagy bármi egyéb, ami fontos a vállalat számára. Ez általában összefoglalja az adott Sprintben létező feladatokat egy egységes cél alá.
4. Fejlesztés: A fejlesztő kollégák elkészítik a megfelelő kódot az előre meghatározott célok alapján. Fontos megjegyezni, hogy sok esetben a specifikáció nem teljes, illetve kérdések merülnek fel, ezért a folyamatos kommunikáció a Scrum rendszer alapja. Erre vannak az ún. Sprint tervező megbeszélések, ahol az egyes feladatokat együtt beszélik meg a kollégák, valamint a Daily Standup[17], ami egy napi rendszerességű egyeztetés, ahol minden fejlesztő 5 percet kap arra, hogy jelezze, hol tart a munkájával és segítséget kérjen vagy adjon a munkatársainak. Az elkészült kódok a fejlesztési architecthez kerülnek, aki a teszt szerverre való élesítés[18] előtt átnézi őket, hogy megfelelnek-e a vállalat által felállított sztenderdeknek.
5. Tesztelés: A Sprint részeként az elkészült kódokat átadják a tesztelőknek, akiket a minőségbiztosítók a Scrum rendszerben. Manapság a tesztelők

[14] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Solution Delivery Manager
[15] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: MD – Mandays
[16] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Development Architect
[17] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Daily Standup
[18] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Deployment

automata regressziós (amely a felhasználó számára már korábban lefejlesztett funkciók megbízható működéséről is megbizonyosodik) tesztrendszereket használnak, amelyek, mint egy virtuális felhasználó, percek alatt elemzik a szoftvert, kipróbálnak minden funkciót és a megfelelő válasz esetén lépnek tovább, illetve jelzik a hibát a tesztelőnek. A tesztelésnek[19] alapvetően három fázisa van:

a. Unit teszt[20]: Ez az alap teszt, amelyet sok esetben már a fejlesztő maga elvégez, miközben kiélesíti az új programkódot egy belső szerveren, amelyen a teszt történik.
b. Service teszt: A programok közötti kommunikációt, adatátadást és egyéb a felhasználó számára nem látható funkciókat tesztel, amelyek szükségesek a program megfelelő működéséhez.
c. UI (User Interface – Felhasználói Felület) teszt: Végigmegy a szolgáltatás minden egyes felületén, kitölt minden elérhető mezőt, megnyom minden gombot és minden elérhető feladatot végrehajt vele amire a megfelelő választ várja.

Talált hiba esetén a tesztelő azt egy ún. bug [21]tikettként jelzi a fejlesztőnek, hogy még élesítés előtt azt ki tudja javítani. Amennyiben egy jelentős hibára nem derül fény a tesztelés alatt, azt egy külön folyamatban jelzik, ami Defect[22]-ként jelenik majd meg a rendszerben.

6. Összefésülés: Jellemzően nem a tesztelés után, hanem vele párhuzamosan zajlik, amikor az egyes fejlesztők által elkészített kódokat a vezető fejlesztő összefésüli és kiteszi a vevői teszt szerverre (UAT[23]), ahol egy újabb, teljes értékű teszt történik, a produktív rendszeren való élesítés előtt.
7. Élesítés: Az elkészült kód átkerül a belső üzemeltetéshez, akik a szerverekért és a már elkészült szoftverek folyamatos működésért felelnek, akik egy gombra való kattintással, - amennyiben kivitelezhető-, munkaidőn kívül felrakják a változtatásokat az éles környezetbe. Ott még lefut egy

[19] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Regression Test
[20] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Unit Teszt
[21] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Bug
[22] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Defect
[23] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Environment

automata teszt, amennyiben az is sikeres, az ún. „Deployment[24]” -et sikeresnek tekintik. Ha esetleg a későbbiekben olyan hiba merülne fel, amely akadályozza a megfelelő működést, egy sürgősségi ún. „hotfix” -et kell kiadni.

A Scrummal kapcsolatban fontos még kiemelni néhány, az alapvető fejlesztési folyamatot támogató részt:

- A Sprint közben hetente összeülnek a csapat tagjai és megbeszélik a Sprint aktuális állapotát, mit tudnak időben leszállítani és mit nem.
- Minden Sprint után tartanak egy ún. retrospektív elemzést az elmúlt Sprintről Sprint során tanultakat, megosztják egymással tapasztalataikat.
- 3 vagy 4 Sprintenként során van egy ún. „refinement session”, amelyen átnézik az elmúlt Sprintek során leszállított kódokat és átnézik, hol lehetne javítani a minőségen és a teljesítményen.
- A Scrum csapat munkájának minden esetben számokkal mérhetőnek kell lennie:
    - Backlog: Az aktuális Sprintben fejlesztések listája
    - Burn Down Chart: A Sprint során már elvégzett feladatok listája
    - Burn Up Chart: A még elvégzendő feladatok és a már elvégzett feladatok naponta frissített listája. Hány fejlesztői munkanapot sikerült elérni a Sprint alatt

Előnyei minden vállalattípus esetében:

- Azonnal alkalmazkodik a felmerülő vállalati, jogi és üzleti igényekhez.
- Költséghatékony, mivel a hibák már a tervezés során kiderülnek.
- Folyamatosan fejleszti önmagát és saját folyamatait a beépített visszajelző rendszerek miatt.
- A vállalat üzleti oldala számára is világos célokat fogalmaz meg és átlátható folyamat.
- Csapatépítő és önszerveződő, azaz nem függnek külső résztvevőktől, csak a bejövő kérelmek határozzák meg a munkájukat.

[24] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Deployment és Release

Hátrányai minden vállalattípus esetében:

- A fejlesztők számára jelentős frusztrációt okozhat a rengeteg kommunikáció és találkozó.
- Sokszor problémakerülő a Scrum módszertan, mert sokkal jobban figyel az új funkciókra és az üzleti megfelelősségre, mint a minőségre és a már létező szolgáltatások fenntartására, azok minőségének emelésére.
- Sok résztvevő számára mikro menedzsmentnek tűnhet a folyamatos számonkérés és mérés, valamint az utánkövetés.
- Mindenképp testre szabást igényel a vállalat profiljától függően és sokszor még így is demotiválhatja a kollégákat az állandó változás.

A knuth-i elvárások teljesíthetősége – vagyis az automatizálhatóság rétegei:

- A Scrum rendszerek holisztikus szemléletűek, így minden elemük a tervezésre és a kód újrafelhasználásra kell, hogy koncentráljon.
- Az ún. microservices-alapú rendszerek miatt a legapróbb részleteik automatizálhatók, mint például:
    - A fejlettebb rendszerek alkalmasak arra, hogy megkeressék a miniapplikációkat és a már megírt funkciók segítségével sokszor néhány sor megírásával és a korábban említettek meghívásával kialakítható egy teljesen új szolgáltatás.
    - Az elkészült kód automatikusan települ a meghatározott rendszerekre, nincs szükség emberi beavatkozásra.
    - A minőségellenőrök a belső és az éles tesztelések során is automata ún. regressziós és teljesítmény tesztet végeznek, amik esetében egy program a felhasználót imitálva figyeli a program futását és reakcióit.

2.2 A fejlesztés előkészítésében résztvevők bemutatása és felelősségi köreik határai, mértéke

A fejlesztések minden esetben alapos előkészítést igényelnek, mivel a vállalat központi folyamatain változtatnak, így akár jelentős bevételkiesést, vevői elégedettség visszaesést is okozhatnak. Ebben a pontban az előkészítési folyamatban részt vevő

kollégákon a folyamatban betöltött szerepeik sorrendjében haladok végig, azt feltételezve, hogy egy globális vállalat professzionális IT szolgáltatásával rendelkezünk.

A fejezet célja a dolgozat elemzésének értelmezéséhez: A vállalatok fejlesztési folyamatainak ismertetése során részletesen kifejtem, hogy az egyes pozíciók hiánya, illetve megléte milyen hátrányokkal és előnyökkel jár, illetve, hogy miért fontos a munkakörökhöz tartozó felelősségek pontos meghatározása.

2.2.1 Projekt Szponzor

A Szponzor felel a projekt mögött pénzügyi háttérért és az üzlet számára vállalt határidőkért, valamint a minőségi szolgáltatás leszállításáért. Amennyiben változás történik a projekt céljában, költségeiben, határidőkben, minden esetben szükséges értesíteni.

2.2.2 Üzleti Elemző[25]

Feladata a Projekt Szponzor által felvázolt cél eléréshez tartozó gazdaságossági számítások elvégzése. Figyelembe szükséges vennie a belső, pénzbeli erőforrás költségeket, a külső partnerekkel való szerződések értékeit, valamint piackutatás vagy egyéb módon meghatározni a projekt által generált várható bevételt, vagy költségcsökkenést. Ennek részletei a 2.7-es pontban kerülnek majd kifejtésre.

2.2.3 Project Management Office[26]

A PMO (Project Managament Office) tagjai az Projekt Menedzserek, akiknek feladata, hogy a Projekt Szponzor által meghatározott célkitűzést megvalósítsa az Üzleti Elemző által meghatározott keretek szerint. Mivel a nagyobb vállalatok összetett hierarchia szerint működnek, ezért az egyes feladatokra minden szerepkörből ki kell választani a megfelelő szakembert, egyeztetni vele részleteket, majd a megfelelő sorrendben azokat végrehajtatni.

[25] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Business Analyst/Üzleti Elemző
[26] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: PMO

2.2.4 Rendszerelemző[27] és Architekt

A rendszerelemzők rendelkeznek a vállalat teljes informatikai hátterének átfogó ismeretével, az architektek pedig tisztában vannak a legújabb trendekkel, amelyekkel segíteni lehet az adott projekt végrehajtását. A két pozíció feladata létrehozni a technikai dokumentációt, majd azt végrehajtásra átadni a fejlesztőknek.

2.3 Használt technológiák rövid leírása és kritikája

A vállalatok által használt technológiák kiválasztása az ún. architektek feladata, akik megvizsgálják a vállalat aktuális és várható, jövőbeni igényeit, a jogi és hatósági megfelelőséget, hogy milyen napi vagy egyéb fejlesztői háttértámogatással rendelkezik, a hozzátartozó liszenszköltségeket majd ez alapján döntést hozni.

A fejezet célja a dolgozat elemzésének értelmezéséhez: A használt technológiák befolyásolják a végrehajtható projektek körét, a komplexitást, a leszállítás hosszát, hogy a kollégák mennyire tudnak együtt dolgozni, illetve a későbbi üzemeltetést is.

2.3.1 Felhő[28] alapú szolgáltatások

A nagyvállalatok már a 2010-es évek közepén elkezdték átköltöztetni szolgáltatásaikat a felhő alapú szolgáltatásokba. Ennek lényege, hogy a szolgáltatások mögötti erőforrások automatizáltan és dinamikusan változnak, így nagyobb leterheltség sem okozhat problémát az ügyfelek kiszolgálásában. Előnye még, hogy nincs szükség a klasszikus értelemben vett szerverüzemeltetésre a szervezeten belül, mivel a fejlesztő saját magának tudja kialakítani a szükséges háttér rendszert, hátránya azonban, hogy egyedi programozási ismereteket kíván, amelyből még nem áll rendelkezésre megfelelő mennyiségű szakember a piacon, így magasabb költségen zajlik a projekt végrehajtása. A felhő alapú szolgáltatással szemben az egyedi szervereken[29] futó alkalmazások jelentik az elavultabb szolgáltatást, amik viszont könnyebben kontrollálhatók és nem igényelnek speciális kialakítást azért, hogy az egyes országok jogi feltételeinek megfeleljenek, melyek például sokszor elvárják,

[27] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: System Architect
[28] Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: AWS, mint példa
[29] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: On Premise

hogy ne kerülhessen ki semmilyen magánszemély adata olyan fizikai adathordozóra, ami az adott ország határain kívül fekszik. Ilyen problémák megoldására léteznek már az ún. publikus felhők mellett a nagyvállalatok számára kialakított egyedi, privát felhőszolgáltatások, amik áruk miatt még nem terjedtek el a KKV szektorban.

2.3.2 Kódbázis

Míg korábban egy szolgáltatást egy fejlesztő írt meg, manapság sokszor 8-10 fős csapatok dolgoznak annak egy funkcióján. Így szükséges volt egy egységes, online kódbázis fenntartása, amelyen egyszerre dolgoznak a kódot író kollégák a saját személyi számítógépükön fenntartott fejlesztői környezetben, majd, amikor elkészültek azzal, egy kérést indítanak a központi kódhoz, amit egy vezető fejlesztő áttekint és összefésül a teljes forráskóddal, így biztosítva a konzisztenciát és a megfelelő működést, minőséget. A technológiai háttértől függően ezen szolgáltatást a vállalatok vagy egyénileg oldják meg, ami jelentős többleterőforrást igényel, vagy a Bitbucket esetleg a GitHub vállalatok szolgáltatását veszik igénybe.

2.3.3 Fordítás

Szoftverfejlesztési szempontból két típusú nyelvet különböztetünk meg, az egyik azok a fajta programnyelvek, amelyek nem igényelnek gépi kódra való fordítást, tehát tulajdonképpen az adott számítógépen vagy magán a szerveren kerülnek lefordításra a gép által értelmezhető iránymutatásokra ezzel azt terhelve (az online rendszerek ezt jellemzően úgy oldják meg, hogy a kliens erőforrásait használják, tehát a honlap amit megtekintünk és amely dinamikusan változik, tulajdonképpen az ügyfél saját gépén fut), valamint azok a nyelvek, amelyek előzetesen lefordításra kerülnek a gép saját nyelvére, így nem igényelnek plusz erőforrást az ügyfél oldalán. Ez a különbség majd több helyen is megjelenik a költségeknél, illetve a vállalatok folyamatainak elemzése során.

2.3.4 Adatbázisok[30]

A szoftverek jelentős része manapság tulajdonképpen csak adatbázisok közti műveleteket végez, azok tartalmát dolgozza fel. Például, ha egy bankautomatánál fizetünk, akkor a számlánkon található összeget ellenőrzi egy algoritmus, amennyiben rendelkezésre áll az összeg, az adatbázison változtatásokat eszközöl, aminek eredményeképp a felvett összeggel csökkenti a még elérhető fedezetet. A folyamat természetesen ennél sokkal bonyolultabb, de lényegében csak és kizárólag adatbázis műveletek zajlanak és különböző applikációk kommunikálnak egymással. Ahhoz, hogy ezt a mennyiségű adatot megfelelő kontroll alatt lehessen tartani, a vállalatok több típusú adatbázist is használnak, amelyek mind összefüggnek egymással és hatást is gyakorolnak egymásra a köztes programok segítségével. A használt programozási nyelvtől, illetve a technológiai háttértől (felhő alapú-e vagy sem) függ az, hogy milyen típusú ún. relációs adatbázisokat hoznak létre. Ezek közül szükséges kiemelni a legújabb kor vívmányát, az „Big Data”-t, ami jobb esetben minden a vállalat életében létrejövő szoftveralapú tranzakció eredményét tárolja, ezáltal lehetőséget adva a vállalat kollégáinak arra, hogy mindig naprakész információval rendelkezzenek a vállalat egészét illetően. Különösen fontos ez egy globális óriás cég esetében, ahol sok esetben 80-100 ország pénzügyi adatait gyűjtik, majd teszik nyilvánossá a részvényesek számára, akik ezek alapján hoznak meg a vállalat szempontjából fontos gazdasági döntéseket.

2.3.5 Microservices

A felhőalapú rendszerek magukkal hozták az ún. Microservice alapú fejlesztési metodológiát, ami tulajdonképpen egy mini alkalmazás. Ennek lényege, hogy az egyes fejlesztők minden projektet funkciókra bontanak és azokra külön erőforrást allokálnak, majd egyedi programrészletet írnak rá. Ez többek között azért nagyon előnyös, mert a későbbiekben ezeket a microserviceket akár másik szolgáltatás is meghívhatja, vagy ha esetleg probléma van a megírt kóddal, nagyon könnyen visszanyomozható, hogy pontosan hol a hiba. A könnyebb érthetőség kedvéért vegyünk egy honlapot, ami online vásárlási lehetőséget kínál. Amikor úgy döntöttünk, hogy a digitális kosárba

[30] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: DB

már nem kívánunk mást tenni és a vásárlásra kattintunk, akkor egymás után meghívásra kerülnek a microservice-k, amiket egy központi szintén microservice hív meg a megfelelő sorrendben:

1. Egy microservice leellenőrzi, hogy az adott termék még mindig van-e raktáron (nem vásárolta-e meg azt időközben más felhasználó).
2. Egy másik, ha elérhető még az áru, azokat lefoglalja a vállalat adatbázisában, hogy azt más ne vásárolhassa meg.
3. A következő előkészíti az online fizetési rendszert és átküldi a banknak a megfelelő adatokat.
4. Szintén a következő átirányítja a felhasználót a bank oldalára, vagy egy beépülő modulban megjeleníti a bank honlapját, ahol megadhatja a tranzakcióhoz szükséges adatokat.
5. Ezután a bank visszajelez, hogy sikeres volt-e a fizetés, amit szintén egy microservice figyel, majd, ha megkapta az üzenetet meghívja a következő lépést.
6. Sikeres fizetés esetén a következő minialkalmazás előkészíti az árukat a kiszállításra, jelzi a munkatársaknak, hogy mit kell csomagolni és automatikus nyomtatja a címkét a szállítónak.
7. A következő értesíti a szállítót a és megadja a termék méretét és súlyát, hogy a megfelelő gépjárművet küldje.
8. Ezután következik a számla és a garancialevél elkészítése, ami jellemzően egy PDF fájl.
9. Az utolsó előtti microservice egy e-mail küldő alkalmazás, ami a vásárló részére elküldi a dokumentumokat.
10. Majd a legutolsó minialkalmazás megküldi a vásárló részére a szállító online link-jét, ahol pontosan követhető a szállítás aktuális állapota.

Az oka pedig annak, hogy miért érdemes minden szolgáltatást ilyen módon megírni? Gondoljunk csak bele, ha szeretnénk fejleszteni a webshopunk és reklamáció esetén lehetőséget biztosítani a vásárlónak, hogy újra bekérje a garancialevelet. Nem kell egy külön programot írnom számára, csak egy újabb microservice-t, ami meghívja a PDF generáló minialkalmazáson, elkészíti a garancialevelet, majd meghívni az e-

mail küldőt és már készen is vagyok, sőt, ha időben gondolkodtam, akkor csak kimásolom ezt a néhány sort a teljes vásárlást vezénylő minialkalmazásból és létre is jött a garancialevélküldő microservice. Ez a fajta fejlesztési rendszer nagyon sokat javíthat a nagyvállalatok piaci reakció idején, ami az utóbbi időben teret adott a startupoknak.

2.3.6 Robotok

Az utóbbi években kezdtek elterjedni a nagyvállalati szférában az automatizálás legújabb eszközei, a robotok. Ne (csak ill. alapvetően ne) fizikai robotként képzeljük el őket, hanem olyan alkalmazásokként, amelyek a felhasználó helyett végeznek el előre meghatározott automatizálható folyamatokat, például rákattintanak egy honlapon két gombra, kitöltenek egy mezőt, megvárják a választ rá, a válasz alapján további mezőket töltenek ki, míg végül egy teljes folyamatot végrehajtanak. Nem annyira finoman hangolhatók, mint egy programkód, amit egy szoftverfejlesztő készít, de alkalmasak arra, hogy egyszerűbb, monoton feladatokat végrehajtsanak ezzel is költséget csökkentve a vállalatnak. Létrejöttük elsődleges oka az informatikában jelenlévő munkaerőhiány, ami miatt az IT-s bérek olyan magasak, hogy egyszerűbb automatizálásokra nem érdemes erőforrás pazarolni.

2.4 Használt segédeszközök bemutatása leírása és kritikája

Mint minden a vállalaton belül található pozíciónak, a szoftverfejlesztésnek is léteznek saját segédeszközei, amelyek használata egyszerűsíti mind az üzlet, mind az IT-s kollégák napi munkáját. A következő fejezetben ezekről szeretnék néhány szót ejteni.

A fejezet célja a dolgozat elemzésének értelmezéséhez: Az informatikában a folyamatok betartása elengedhetetlen, hiszen egy rendszer összeomlását akár egy nem megfelelően elhelyezett pontosvessző is okozhatja, így az elemzések során lehet majd látni, hogy milyen hátrányokkal jár, ha a vállalat nem használja jól a megfelelő segédeszközöket.

2.4.1 JIRA[31]

A JIRA egy kifejezetten a szoftverfejlesztési folyamatokat támogató szolgáltatás csomag, amelyet az Atlassian nevű vállalat készített el és amit a legtöbb nagyvállalat használ. Támogatja az Agile alapú fejlesztést, minden szempontból testre szabható, egyedi feladatkörök határozhatók meg benne és szigorúan egyirányú folyamatok kialakítására alkalmas, ami segíti a kollégák munkáját. Hátránya legfeljebb csak annyi lehet, hogy a felhasználók jelentős része nem ismeri a szolgáltatás képességeit, így a legtöbb cég a képességeinek jelentős részét nem használja ki, csak dokumentálja az eseményeket, nem pedig kontrollálja. Nagy előnye, hogy lehetőséget ad arra, hogy összekössük a fejlesztés során használt egyéb programokkal, pl., ha a folyamat egy pontján akarjuk, a JIRA automatikusan utasítja az általunk használt külső programot, ami élesíti a legújabb programkódot, hogy azzal már ne is legyen munkánk. Fontos megjegyezni, hogy a JIRA-t nem csak a kódolással foglalkozó kollégák használhatják, hanem mindenki, aki részt vesz az informatikai fejlesztések bármely pontjában.

2.4.2 Confluence[32]

A Confluence szintén az Atlassian vállalat terméke, feladata pedig, hogy egységes és átlátható dokumentációs rendszer alkosson a JIRA-val együtt. A dokumentáció mindig sarkalatos pontja a fejlesztéseknek, mert egy kolléga átmeneti kiesése jelentős bevétel csökkenést, projektek csúszását okozhatja, ha nincs, aki átvegye tőle. Ez pedig az érthető és átlátható leírások nélkül nem megoldható.

2.5 KPI[33]-ok meghatározása és kritikája, sikeres és nem sikeres projektek megkülönböztetése

A szoftverfejlesztéshez tartozó metódusokat matematikusok és programozók dolgozták ki, így minden ponthoz megfelelőségi kritériumokat és pontosan meghatározott definíciókat hoztak létre. Ez azt jelenti, hogy nem csak a teljesítményt mérik, hanem a projekt egyes pontjait is egyértelműsítik, például, hogy mit jelent az,

[31] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: JIRA
[32] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Confluence
[33] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: KPI

hogy az adott kód tesztelésre készen áll, vagy, hogy mit jelent az, hogy a tesztelés sikeres stb. Jelen fejezetben csak a fejlesztéshez tartozó KPI-okat tekintjük át, a projektek előrehaladását és egyéb mutatóit mérő számításokat a „Gazdaságossági számítások és kritikájuk" pontban (2.7) fejtem ki.

A fejezet célja a dolgozat elemzésének értelmezéséhez: A fejlesztési projektek és az elkészült, használható kódok, valamint az egyes informatikus kollégák teljesítménye nagyon nehezen mérhető (Waterfall esetében egyáltalán nem), főleg, ha össze akarjuk hasonlítani őket egymással. Ezért az Agile alapú szoftverfejlesztés alapítói ezekre is kidolgoztak egy egységes rendszert. Ezeket változó minőséggel, de használjak a vállalatok, a nem megfelelő bevezetésük azonban torz képet adhat a vezetőknek, ezt részletesen kifejtem az egyes elemzések során. Alább néhány az általános mérési pontokból: Mindegyik mérőszám egy Sprintre vonatkozik Agile metódus használata esetén

- Átláthatóság: Vállalt feladatokat és leszállított feladatok aránya az előre meghatározott fejlesztői napok számában megadva.
- Produktivitás: A leszállított fejlesztői napok számossága.
- Hatékonyság: A leszállított feladatok munkanapokban mért számossága a sprintben lévő munkanapok számával osztva (torzíthatják a szabadságok).
- Minőség: A leszállított feladatokban talált, a minőségellenőrök által feltárt hibák számossága az adott időszakban – Illetve bizonyos esetekben hibajegyek és a fejlesztések aránya egy Sprintben.
- Bevételtermelés: A leszállított feladatok munkanapokban mért számossága a hozzájuk az üzleti elemző által meghatározott a vállalat számára várható plusz bevétel aránya.

2.6 Élesítés – Szolgáltatás átadás-átvétel

Amikor élesítésre kerül egy kód, akkor mindegyik típus esetében az elkészült kód összefűzésre kerül egy vezető fejlesztő által, aki azt felmásolja egy belső teszt szerverre, hogy azt a vállalat által foglalkoztatott vagy külsős minőségellenőr tesztelje. Ha megfelelt az elvárásoknak a változtatás, akkor a fejlesztők, vagy az alkalmazásüzemeltetők azt élesítik a vállalat szerverein.

A fejezet célja a dolgozat elemzésének értelmezéséhez: Az átadás-átvétel a szállítás egyik legfontosabb pontja, mivel ekkor készítik fel a rendszert a későbbi folyamatos használatra, ekkor derül ki, hogy mennyire fenntartható a szolgáltatás és elfogadható-e az eredmény a partner vagy a belső végfelhasználók számára.

2.6.1 Technikai dokumentáció

A vállalatok jellemzően nem rendelkeznek elég erőforrással a megfelelő dokumentáció elkészítéshez, vagy csak egy-egy változtatás dokumentálnak le. Ennek eredményeképp, ha valaki meg szeretné tudni egy szolgáltatás aktuális állapotát, először meg kell néznie az eredeti koncepciót, majd minden egyes változtatást átolvasnia. Azért, hogy ilyen probléma ne forduljon elő, mind a rendszerelemzőknek, mint az architektnek szükséges kiegészíteni a már létező dokumentumokat az aktuális változtatás eredményeivel. Ez a későbbiekben is segíti a közös munkát, mivel egységes álláspont alakul ki a rendszert illetően.

2.6.2 Felhasználói leírás

Amennyiben a változtatás, azaz a leszállított feladat változást hoz a felhasználói munkában, például újabb felület jelenik meg, funkciókkal egészül ki a szolgálatatás, bizonyos funkciókat más helyen érhetünk el, mindenképpen szükséges a felhasználói leírás aktualizálása ezzel is csökkentve az informatikus kollégák leterheltségét, mivel kevesebb beérkező hívás érkezik a partnerektől.

2.6.3 Hotline felkészítés

Minden változtatás hatással lehet a felhasználói élményre, amire szükséges felkészíteni a később a szolgáltatás élesben üzemelő kollégákat, hogy ha esetleg bármilyen reklamáció érkezik, azonnal tudjanak reagálni, a kisebb hibákat akár személyesen javítva, a nagyobb hibákat pedig felismerve, azokat továbbítani a fejlesztésnek a mélyebb változtatást igénylő feladatok végrehajtása érdekében.

2.6.4 Oktatás

Új szolgáltatás bevezetésekor, nagyobb folyamatbeli átalakuláskor mindenképp szükséges a végfelhasználók tájékoztatása vagy oktatása, nem csak a megfelelő használat érdekében, hanem azért is, hogy a felmerült kérdésekre egyértelmű válaszokkal szolgáljunk. Ezzel hosszútávon megint csak a saját munkájukat segítik az informatikusok, mivel jóval kevesebb lesz a beérkező hibajelzés, segítségkérés.

2.7 Gazdaságossági számítások és kritikájuk

Amennyiben nem törvényi megfelelősség indokol egy adott projektet, szinte mindegyik alapját képezi a gazdaságossági számítás, amely szembeállítja egymással a vállalat üzleti elvárásait, legyen az plusz bevétel vagy költségcsökkentés, valamint a feladat végrehajtásához szükséges erőforrásokat és költségeket.

A fejezet célja a dolgozat elemzésének értelmezéséhez: Az elemzésben több olyan helyzettel is megismerkedhetünk majd, amely során a vállalatok nem fektetnek megfelelő energiát az ún. CBA (Cost – Benefit Analysis) létrehozásába, vagy mert jelentősen alul becsülik a szükséges költségeket és erőforrásokat, vagy mert a piac nem megfelelő felmérése miatt téves bevételi terv került meghatározásra. Ebből a szempontból érdemes megjegyeznünk, hogy vállalatok belső fejlesztései az esetek jelentős többségében az EVM (Earned Value Management), tehát a „Megtermelt Értékek Módszere” típusú Projekt Menedzsment rendszert használják, amely megpróbál objektív és minden résztvevő számára érthető képet adni a projekt aktuális állapotáról, előrehaladásáról és várható eredményeiről. Az EVM három alappillére a „Tervezett érték” (Planned Value), a „Megtermelt érték” (Earned Value), valamint a Felmerült költségek (Actual Cost).

2.7.1 Tervezett érték - PV

A „Tervezett érték” feladata, hogy a projekt adott pontjában egyetlen számmal tudja megmutatni, hogy hol kell(ene) tartani a megtervezett munkának és ez alapján kerülnek kialakításra a főbb mérföldkövek a feladatok megtervezésekor. Ez a mutató a projekt során csak és kizárólag abban az esetben változhat, ha az elérni kívánt cél jelentős mértékkel változik. Esetünkben fontos megjegyezni, hogy a tervezett érték

meghatározása valójában dolgozatom egyik fő célkitűzése, az ún. „Információs Többletérték” eredményként realizálható összege, legyen az akár piaci előny, bevétel növekedés, profithányad javulás.

2.7.2 Megtermelt érték - EV

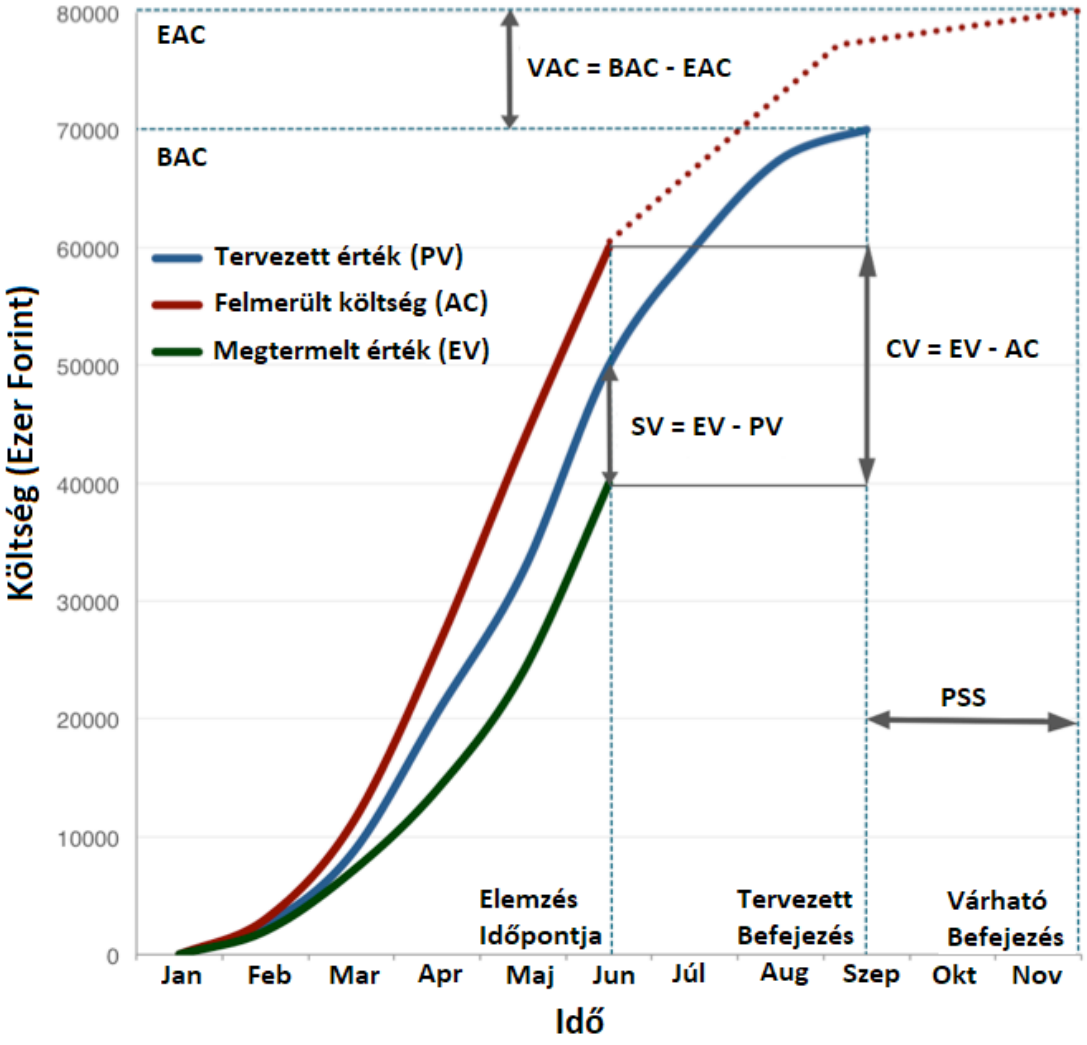
A „Megtermelt érték” a feladat előrehaladását mutatja egy adott időpontban, tehát az elvégzett munkát reprezentálja. Jellemzően a „Tervezett értékkel” közös diagrammon kerül bemutatásra, így jelezve, hogy mekkora a különbség a tervezett és a valóban elvégzett munkák közt, azaz a projekt teljesítményét mutatja.

2.7.3 Felmerült költség - AC

A „Felmerült költség” a projekt során igénybe vett erőforrások által ténylegesen elvégzett munka költségét mutatja egy adott időintervallumban. A költségek tartalmazzák mind az élőmunka, mind az egyszeri vagy folyamatosan felmerülő külső költségeket is.

2.7.4 Elemzés és előrejelzés

A projekt elemzésekor a korábban meghatározott értékeket vetik össze, így meghatározva azt, hogy a projekt mennyire sikeres és mekkora a befektetett munka megtérülése. Az alábbi ábrán ennek összefoglalóját látjuk:



Ábra 4.: EVM alapú projekt menedzsment változó értékei (Forrás: techsite.com, saját fordítás)

Jelmagyarázat:

- Y tengely: A költség (Ezer Forintban), illetve létrehozott érték pénzbeni összege
- X tengely: A projekt idővonala (Hónapok)
- Piros görbe: Felmerült költség (AC)
- Kék görbe: Tervezett érték (PV)
- Zöld görbe: Megtermelt érték (EV)
- PSS (Project Schedule Slippage): A Projekt várható csúszása

- SV = EV-PV (Schedule Variance): A „Tervezett érték” és a „Megtermelt érték” közötti különbség a projekt adott pillanatában. Megmutatja, hogy mennyivel van elmaradva projekt teljesítménye az előre eltervezettől
- CV = EV – AC (Cost Variance): Az projekt jelenlegi állapotáig „Megtermelt érték” és „Felmerült költség” különbsége, ami a projekt rentabilitását jelzi
- BAC (Budget at Completion): A projekt előre eltervezett teljes költsége
- EAC (Estimate at Completion): A projekt jelenlegi állapota alapján kalkulált várható költsége
- VAC = BAC – EAC (Variance at Completion): A projekt jelenlegi állapota alapján várható többletköltség

2.7.5 A gazdaságossági számítások értékeinek definíciója

A gazdaságossági számításoknál, minden esetben arra törekedtem, hogy a kiadásoknál a legrosszabb forgatókönyv szerint kalkuláljam, míg az elérhető nyereség/bevétel, illetve költségcsökkentésnél az optimális, nem pedig az valóságtól elrugaszkodott összegeket határozzak meg, így következő azok értékei a következőket tartalmazzák:

- Több éves megtérülésnél nem számoltam a kiadás oldalon a béremelkedéssel, illetve az általános inflációval, valamint bevétel oldalon a partnerek általi áremelkedéssel, mivel ezek egy része előre nem meghatározható, valamint jellemzően kiütik egymás pozitív és negatív hatásait. Ebből a szempontból fontos még megjegyeznem, egy nem feltétlen pozitív, azonban a számokat befolyásoló tényezőt: Tapasztalatom szerint a multinacionális cégek leányvállalatai jellemzően nem stratégiai, hanem taktikai döntéseket hoznak, azaz, bár egységes, központi vízióval rendelkeznek, a helyi entitások, még, ha kötelezik őket is arra, hogy a következő három évre tervezzen, csak az aktuális, vagy éppen kezdődő gazdasági évre koncentrálnak ott is arra, hogy pontosan a bónuszt jelentő növekedést és nyereséghányadot érjék el, s többet (mivel akkor a következő évi bázis lesz magasabb, amiről nehezebb lesz további, hasonló minőségű növekedést elérni, azonban magasabb bónusszal nem jár), se kevesebbet, mert akkor esetleg alacsonyabb kifizetés történhet

- Az előbbi pontból fakadóan, a várható bevételeknél vagy kiadáscsökkenésnél, tehát a befektetés pozitív fejleményeinél, a tervezésnél minden projektre optimális összegekkel számoltam (kivételt képeznek ez alól, azon tételek, ahol adatbázis szinten rendelkeztem az adatokkal, vö. Melléklet 1.), amiket legalább 80%-os valószínűséggel el lehet érni a kollégák piaci ismeretei szerint és természetesen a jelenlegi adóügyi szabályok szerint, tehát a javító amennyiben viszonteladóként tovább értékesít egy alkatrészt, az ÁFA-val egyik oldalon sem kalkuláltam, számoltam azonban a kárkifizetések esetében az ÁFA értékkel, mivel azt egy gépjármű megjavításakor a biztosítónak állnia kell a javító irányába, azonban nem számoltam az átlagkár meghatározásánál egyezségi kárrendezés esetén, mivel a jelenlegi jogi környezetben azt nem terheli ÁFA.
- A kiadási oldalon, minden esetben az elérhető adatokból dolgoztam, ahol pedig nem állt rendelkezésre saját forrás, a legrosszabb verzióval kalkuláltam, hogy a döntés során a vezetőknek mutatott mérleg csak abban az esetben legyen pozitív, ha a projekt nyereségességére valóban nagy esély mutatkozik. Ennek megfelelően a kiadási oldalon megjelenő értékek minden esetben tartalmazzák az esetlegesen felmerülő költségeket, legyen az ÁFA, vagy például a munkabérek esetén a járulékok, a munkavállaló számára biztosított kellékek elosztott költsége, stb.

2.8 Szoftverfejlesztés információs többletérték(ének) fogalma, jelensége

Korunk nagy egyik legnagyobb kérdése, hogy hogyan határozhatjuk meg azt az értéket, amelyet a különböző informatikai szolgáltatások a végfelhasználó számára jelentenek. Legyen az egy termék jövőbeni ára ami alapján a vásárlás időpontjáról döntünk, vagy csak egy pénzben nem értelmezhető kiegészítő adat, ami segíti a mindennapokban való tájékozódásunkat.

A fejezet célja a dolgozat elemzésének értelmezéséhez: Amennyiben a szoftver fejlesztése külső partner számára történik nem csak tisztán projekt alapú, tehát a kódon kívül más hozzáadott értékkel is rendelkezik a vállalat, mint például egy ún. DaaS (Data as a Service) amikor nem nyers adatbázisokat árul egy cég, hanem feldogozott

formában kereskedik vele, a megfelelő ár meghatározásához szükséges az információs többletérték meghatározása.

Ezen érték meghatározására a vállalatok általában saját módszert alkalmaznak, ezek közül az egyik, a speciálisan egy termék egy adott partnertípus számára kidolgozott értékének meghatározása az ún. Value Proposition. Ilyen esetben a vevő helyzetébe képzeljük magunkat és az ő folyamataiban illesztjük be a saját szolgáltatásunk, határozzuk meg azt, hogy számára mekkora költségcsökkenést, vagy bevételnövekedést fog jelenteni plusz információ, majd ennek bizonyos hányadát állapítjuk meg árként. Ennek nagy hátránya, hogy a teljes piacot és a vevőt is teljes mértékben szükséges ismernünk, azonban előnye, hogy amennyiben tényleg jól határozzuk meg az árat, akár értékesítési tevékenység nélkül is eladható a szolgáltatás. Nagyon jó példa erre a gépjármű biztosítás olyan beszállítói, akik akár a gépjárműről, akár korábbi káreseményekről, vagy a gépjármű tulajdonosáról bármilyen információval rendelkeznek, azt a biztosító rendelkezésére bocsájtják, majd a biztosító ez alapján reálisabban tudja meghatározni a rizikófaktorokat. Ha a beszállító tisztában van mögöttes adatokkal, mint, hogy mennyi várható plusz kárkifizetéstől menti így meg a biztosítót, hány esetben hasznos az általa átadott adat stb., egyértelmű és számokkal alátámasztható árat tud meghatározni, ami a partnere számára is elfogadható.

2.9 Szoftverergonómia és a felhasználói igényfelmérés kritikája

Bár dolgozatomnak nem célja a felhasználói ergonómia mérése és a felhasználói élmény mérésének rendszere, azonban külső fejlesztések esetén szükséges azok előzetes kialakítása, mivel az információ magában hiába rendelkezik hozzáadott értékkel, ha annak a rendszerből való kinyerése nehézkes, nem egyértelmű, vagy hiányosan megtervezett.

A fejezet célja a dolgozat elemzésének értelmezéséhez: Az egyes vállalatok folyamatainak elemzése során, egy-egy példával külön kitérek majd az előzetes felhasználói igényekre, valamint arra, hogy végül mi és milyen formában került abból leszállításra. Teszem ezt azért, hogy magyarázattal szolgáljak arra, miért fordul elő rengeteg esetben, hogy a Projekt Szponzor által kívánt elképzelést a felépített

szolgáltatás tökéletesen lefedi, azonban a felhasználói kényelmetlenségek miatt a célját sosem éri el, mivel vagy nem az előírásoknak megfelelően használják azt, vagy inkább kikerülik azt a kollégák. Még nagyobb problémát okozhat ez, ha magánszemélyek számára készítünk applikációt, ahol egy webshop esetén, a felhasználói élmény hiányában potenciális vásárlókat veszítünk.

3 A KUTATÁS ÉS EREDMÉNYEI

3.1 Kutatás, elemzési szempontok és módszertan

Dolgozatom céljaként határoztam meg, hogy egy-egy szabadon választott, ám általam jól ismert nagy, illetve középvállalat szoftverfejlesztési projektjét (vö. 3.2 és 3.3,) tekintem át a 2-es pontban leírt szempontok alapján. Feltételezésem szerint a vállalatok között jelentős különbségeket fogok találni, így az összehasonlító elemzésnek értelme nincs, azonban az alábbi hiányosságokra és problémákra külön fókuszt helyezek:

- Az egyes előkészítő pozíciók hiánya, mint akár a Projekt Menedzser, vagy az Üzleti Elemző, amelyek nélkül egy nagyobb szervezeten képtelenség megfelelő minőségben véghez vinni egy összetett feladatot vagy sok esetben olyan projektet végre hajtani, amely már eleve bukásra ítélt, vagy nem fog megtérülni.
- A végrehajtó és ellenőrző pozíciók problémái, mint a fejlesztői erőforrás folyamatos hiánya, vagy a nem megfelelő minőségű tesztelők a vállalatnál, akik csak manuálisan tesztelnek, de nem készek még az automata tesztelésre így lassítva a szállítás folyamatát.
- A munkakörök és a felelősségek nem megfelelő definiálása a folyamatok során, amik miatt egyes feladatok huzamosabb ideig egyhelyben állnak.
- Hiányos dokumentációk, illetve az azokból való felkészülés a projekt végrehajtására, majd mindezzel egyetemben az aktuális feladatok alul dokumentáltsága, amik már a tervezés közben félrevezetik a projekt résztvevőit.
- A használt fejlesztői módszertanok közötti különbség és az ebből fakadó hátrányok és előnyök, külön kiemelendő a Scrum rendszer üzlet kiszolgálásában fellelhető pozitív hatását.
- A kialakított folyamatok lépéseinek hiánya vagy nem megfelelő súllyal való kezelése, mivel ezek a legtöbb vállalatnál csak papíron léteznek, azonban a

jelentős üzleti nyomás miatt sokszor átugorják őket, amik a későbbiekben jelentősebb problémákat okoznak.

- Technológiai fejletlenségből eredeztethető hátrányok, például a felhő alapú szolgáltatásra való áttérés halogatása a felmerülő költségek és munkaigény miatt, amivel csak azt érik el, hogy amit jelenleg fejlesztenek, majd újra kell az áttérés után ezzel maguk előtt görgetve egyre több és több feladatot.
- Szintén technológiai és komplexitást vizsgálva az újra felhasználható kódok létrehozásra, mivel jól mutatja mennyire holisztikus szemléletű a vállalat.
- Használt segédeszközök és azok megfelelő használata, mivel sok esetben láttam, hogy vezették azokat, de sose készítették fel a produktív és hatékony használatra, csak kipipálták egy sorban.
- A projekt és a szolgáltatások mérőszámainak meghatározása és azok egységes követése, mert véleményem szerint nagyon sok esetben vagy a nem megfelelő értékek határozzák meg, vagy ami még rosszabb manipulálják azokat a vezetőség felé leadott riportokban.
- Az élesítés egyes hibáit, kifejezetten a felhasználók tájékoztatása az újdonságokra, ezzel is segítve munkájuk.
- A gazdaságossági számítások, különös tekintettel az azt megelőző piackutatásra, amit az üzlet nem megfelelően végez el, mivel annyira hisz a saját elképzelésében, hogy úgy gondolja, mindenképp el fogja érni vele az éves tervet.

Ennek egységes mérésére az alábbi „Táblázatot 5.”-t dolgoztam ki:

Folyamat	Szempont	Megfelelősség		
		Jó	Fejlesztendő	Nem megfelelő
Általános	Felelősségi körök	A felelősségi körök jól határoltak, a folyamat során mindenki tisztában van a feladatával és a hatáskörével is, az újonnan fejlesztendő szolgáltatásnak van tulajdonosa (Product Owner)	A hatáskörök nem egyértelműek, a kollégák sok esetben más helyett kénytelenek dolgozni, vagy több munkakört visznek	A hatáskörök nem léteznek, egy kollégának kell a folyamat több pontját is menedzselnie
Általános	Folyamat	Az innovációs folyamat minden lépése előre meghatározott és tartalmazza az összes résztvevő feladatát,	Az változási folyamat létezik, a kollégák tudják a feladatukat, azonban nem rendelkeznek átfogó	A változási folyamat nem létezik a vállalatnál, így ad-hoc jelleggel

Folyamat	Szempont	Megfelelősség		
		Jó	Fejlesztendő	Nem megfelelő
		határidejét és az elvárt eredményt, az egyes lépésekhez létezik ún. DoD – Definition of Done[34]	képpel arról, hogy mi a projekt aktuális állapota	történik a fejlesztés
Általános	Jog	Jogi oldalról áttekintésre került a projekt, vevői és beszállítói egyéb szerződéseket nem sért, az adatforgalom és a személyes adatok pontos tárhelye ismert és megfelel a GDPR előírásoknak	A vállalat csak a személyes adatokkal foglalkozik a kötelező GDPR [35] miatt, azonban nem figyel az egyéb szerződésekkel való együttműködésre	Jogi oldalról egyáltalán nincs megvizsgálva a fejlesztésre kerülő folyamat
Üzleti elemzés	Kiadások	Minden tervezhető kiadást tartalmaz, még a nem projekt oldalon megjelenő belső kollégák erőforrásainak értékét is, az egyszeri kiadásokat és a megújuló költségeket is, legalább három évre tervez, tartalmaz legalább 20% pluszt a felmerülő költségekre	A kiadások tervezése nem terjed ki minden tételre, csak a pénzbeli kiadásokra szűkül, a belső erőforrásokkal csak részben, vagy egyáltalán nem számol, legalább egy évre tervez	A kiadások tervezése csak a pénzbeli kiadásokra korlátozódik, vagy még azok se kerülnek előzetesen elhatárolásra, egy évre sem tervez
Üzleti elemzés	Bevételek	Minden potenciális költségmegtakarítást vagy realizálható bevételt tartalmaz, piaci szegmensekre bontva, havi szinten, a hozzá tartozó valószínűséggel	Csak a pénzbeli bevételekkel számol, sok esetben nem is indít költségmegtakarításra projektet	Rövid becslést végez a következő hónapokra várható bevételekről
Döntés	Vezetői kompetenciák	A döntéshozók átolvasták a döntéselőkészítő anyagokat, tisztában vannak az alternatívákkal és a döntés következményeivel, megbizonyosodtak a tervezés megfelelő minőségéről	A döntéseket nem készíti elő megfelelő munka és tervezés, a döntéshozók nem rendelkeznek piaci ismeretekkel esetleg az elemzéseket sem ismerték meg	A döntéshozatal nem csapatban és nem egyeztetéseken alapulva születik meg
Megrendelés	Általános	A Projekt Menedzser a megfelelő kollégákkal jó minőségben lefordította az üzleti elvárásokat a technológia nyelvére, a megvalósíthatósági terv megfelel a vállalat sztenderdjeinek, a megrendelés tartalmazza a pontosan a projektre szánt büdzsét és munkaórát, amely túllépés esetén újabb engedélyeztetési folyamatot generál	A Projekt Menedzser nem rendelkezik technológiai tudással, így csak a felületes elvárásokat ad át a megrendelés során, a folyamatra nincs hatással és nem is tudja projektet követni, megfelelően kézben tartani	A projekthez nincs Projekt Menedzser allokálva, a megrendelés az elkészült dokumentumokat átadását foglalja csak magában
Fejlesztés	Módszertan	A fejlesztők Scrum módszertan alapján	A fejlesztés Waterfall módszertan szerint	A fejlesztők nem alkalmaznak

[34] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: DoD – Definition of Done

[35] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Audit, Access Rights, GDPR, Személyes Adat

Folyamat	Szempont	Megfelelősség		
		Jó	Fejlesztendő	Nem megfelelő
		dolgoznak, együtt határozzák meg a feladatokat és a közös tudás eredményeképp születik meg a szolgáltatás, a felelősségi körök egyértelműek és megfelelően használják	zajlik, nem figyelik az egyes projektek prioritását, a fejlesztők egyedül dolgoznak egy-egy projekten	semmilyen egységes módszertant, egy kolléga egyszerre több projekten dolgozik
Fejlesztés	Prioritás	A fejlesztői csapat és vezetőjük tudja az egyes projektek közti prioritást, az adott feladathoz előkészítette a rendszereket (tárhely, erőforrás stb.) a tervezett időt elkülönítette a kollégák munkaidejéből, szabadságokkal és egyéni teljesítményszint-beli különbségekkel számol	A vezető fejlesztő rendelkezik a megfelelő tudással a csapat vezetésére, azonban nem tervez 2 hónapnál tovább, a tervezés is kimerül az egyes feladatok nagyságrendi erőforrás igényének meghatározásában és a feladatok kiosztásában	A fejlesztésben nincsenek prioritások ad-hoc készülnek az új funkciók, a feladatok kiosztása véletlenszerű
Fejlesztés	Dokumentáció	A fejlesztés minden pontja dokumentálásra kerül, a megírt kód átlátható és megfelel a belső sztenderdeknek, a korábbi fejlesztések is ilyenek így segítik a kollégák munkáját, az applikációk közötti adatáramlás egyértelmű és átlátható	A dokumentáció hiányos, a kód nem kerül kommentálásra, az applikációk közötti adatátadás részben dokumentált, a korábbi fejlesztések is viszonylagosan dokumentáltak	A szolgáltatáshoz nem készül dokumentáció, csak a megrendelés alapján elkészül a szükséges kód, a korábbi dokumentálatlan kódok miatt a fejlesztő jelentős időt veszít azok újra megismerésével
Fejlesztés	Technológia	A vállalat szolgáltatásai felhő alapú rendszereken futnak, saját kódbázissal rendelkeznek ahol a kollégák látják egymás munkáját és a projekten belül tudnak egyes modulokon dolgozni, a fejlesztői környezeten automatikusan frissül a kódbázis, valamint a tesztelésnek való átadás is automatizált, minden résztvevő azonnal értesül a változtatásokról, a vállalat rendelkezik Big Data rendszerrel ami elérhető a fejlesztők számára így szolgáltatás nem másik több szolgáltatás működésétől függ (alacsony dependencia)	A vállalat éppen költözés alatt van a felhő alapú szolgáltatásokba ezért az új funkciók már oda készülnek, közös kódbázissal rendelkeznek, de egyénileg használják azt, fejlesztői környezet rendelkezésre áll, de kézzel szükséges frissíteni, a szolgáltatás több egyéb szolgáltatásra épül így azok kártyavárként omolhatnak össze	A vállalat egyedi szervereket használ, elavult programnyelveket használ, a kód a kollégák saját gépén kerül tárolásra és azon is fejlesztenek belső környezet híján
Fejlesztés	Hozzáértés	A fejlesztők rendelkeznek a legújabb technológiákkal kapcsolatos tudással, folyamatos oktatásban	A fejlesztők maguk gondoskodnak arról, hogy a legújabb technológiákat	A fejlesztők a korábbi kódjaikat egészítik csak ki, nem is

Folyamat	Szempont	Megfelelősség		
		Jó	Fejlesztendő	Nem megfelelő
		részesülnek, a vállalat engedi a kreatív, szabad munkavégzést számukra, a fejlesztő kollégák képesek a nagyságrendi, ún. unit test elvégzésére, a tesztelés eredményét megértik és ismerik a kódot annyira, hogy azt időben javítsák	ismerjék, a vállalat nem fektet azokba, így élesben nem használhatják a tanultakat, saját maguk által fejlesztett kódot jellemzően ők tesztelik	érdeklődnek újabb technológiák után
Fejlesztés	Tesztelés	A tesztelő kollégák automatizált tesztelést végeznek akár fejlesztői kompetencia szinttel, létezik a szolgáltatásokról és a felhasználói folyamatokról teljesértékű dokumentáció, amely alapján az automata robotokat felépítik, a regressziós tesztek eredményeképp a fejlesztőknek való hibás részek automatikusak visszajelzésre kerülnek, Létezik a DoD[36]	A tesztelés manuálisan történik, mivel nincs automatizált tesztelőre kerete a vállalatnak, emiatt regressziós teszt is csak részben történik, az új funkciók és a korábbi nagyobb funkciók végig kattintgatásával	Tesztelés csak az adott funkcióra történik, a legtöbb esetben azt is a fejlesztő végzi
Fejlesztés	Élesítés	Az éles rendszerre való telepítést nem a fejlesztők, hanem az üzemeltetés végzi, a fejlesztés után a tesztelés automatikusan lefut, kisebb változtatások esetén egyáltalán nincs szolgáltatás kiesés, nagyobb változtatás esetén is csak minimális, de semmiképpen sem több, mint egy óra	Az élesítést sok esetben a fejlesztő végzi nem automatizáltan, minden alkalommal van szolgáltatáskiesés a frissítés alatt azonban az rövid ideig tart	Az élesítést a fejlesztő végzi, a tesztelés hiánya miatt rendszeresen hosszabb időre leáll a szolgáltatás, mivel éles környezetben kell javítani
Indítás	Átadás - Átvétel	A szolgáltatás változásainak dokumentálása megtörtént, a tesztelés eredményét megosztották a szolgáltatás későbbi üzemeltetőivel, a sűrűn felmerülő kérdésekre választ tud adni és az egyszerűbb, főleg felhasználói hibákat tudja kezelni, komplex képpel rendelkezik arról, hogy hol keresse a hibát és mi az, amivel a későbbiekben a fejlesztőkhöz kell fordulnia és mi az, amit meg kell tudnia oldani, a szolgáltatáshoz tartozó felhasználókat és jogosultságokat képes adminisztrálni, A Service	Az átadás-átvétel részben dokumentált, az üzemeltetők csak egyszerűbb változtatásokra vannak felkészítve, amik később is jelentős erőforrásokat foglalnak le a fejlesztői oldalon, a felhasználói hibákat megfelelően kezelik, a jogosultságokat és egyéb adminisztrációkat képesek elvégezni	Az átadás-átvétel csak szóban történik meg, sokszor az egyszerűbb hibák is fejlesztői erőforrást kötnek le, az adminisztráció csak minimális szintű vagy teljesen a fejlesztő által történik

[36] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: DoD – Definition of Done

Folyamat	Szempont	Megfelelősség		
		Jó	Fejlesztendő	Nem megfelelő
		Delivery Manager [37] megértette és átvette az üzemeltetés		
Visszajelzés	Mérőszámok	A Projekt Menedzser és a Product Owner együttesen kezelik a szolgáltatás utógondozását, rendszeres időközönként visszajelzést adnak a szolgáltatás minőségéről, további fejlesztéseket és funkciókat fejlesztetnek, figyelik a projekt sikerességét, legyen számokban mérhető bevétel, esetleg költségcsökkentés vagy vevői elégedettség felmérés	A projekt lezárása után még legalább fél évig követik annak eredményességét, azonban nincsenek egységes mérőszámok a sikerességre	Az után követes minimális, nem figyelik a fejlesztés eredményességét mérőszámokkal

Táblázat 5.: Vállalatok kiértékelési táblája (Forrás: Saját gyűjtés a fejlesztési folyamatok alapján)

3.2 Egy hazai vezető biztosító fejlesztési folyamata, eredményei, kritikája egy stratégiai jelentőségű projekten keresztül bemutatva

A klasszikus szoftverfejlesztés 20-30 éves múltra tekint vissza, azonban az utóbbi 10 év azonban gyökeres változást hozott a szegmensben az általános digitalizálódás miatt, mivel a felhasználók már mindent interneten kívánnak intézni (vö. https://en.wikipedia.org/wiki/Soft_systems_methodology). Ehhez azonban a globális vállalatok meglehetősen lassan tudtak alkalmazkodni, főleg méretükből fakadóan (vö. https://dandreamsofcoding.com/2014/12/16/why-big-companies-slow-down/).
Jellemzően ez vezetett a startup-ok kialakulásához, amelyet sokszor ezen vállalatok vásárolnak fel. Az alábbi elemzésen egy ilyen globális pénzügyi vállalat, egy biztosító IT rendszerét és folyamatait elemzem majd, szakdolgozatom címének függvényében. Az elemzés fókuszpontjai a 3.1-es fejezet „Táblázat 5.”-ban foglaltak alapján történnek.

Elemzésemben egy, a biztosító rendszereibe bevezetett változtatást fogok áttekinteni, ami a gépjármű biztosításon belül több belső folyamatra is hatással lesz. Ebben az esetben egy külső szolgáltató a biztosító partnere, aki egy szoftveralapú

[37] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: Service Delivery Manager

hívásra válaszolva, amely tartalmazza az adott gépjármű alvázszámát, megadja, hogy korábban milyen sérülései voltak az autónak, azok mikor történtek, milyen alkatrészek kerültek cserére, javításra, fényezésre és minden egyéb a káreseménnyel kapcsolatosan elérhető nem személyes adatot, amit korábban a javítási költségbecslés során felvittek a szolgáltatásukba. A szolgáltató által biztosított adatokat (vö. Melléklet 1.: szekunder adatok katalógusa) a biztosítói folyamat két pontján tervezik bekötni a rendszerbe, amelyek a projekt során a „Tervezett értéket" vagyis esetünkben többek között az „Információs Többletértéket" képviselik majd:

- A biztosítás megkötése előtt, ajánlatadáskor, hogy a vezető által okozott korábbi káresemények alapján alakítsa ki a biztosító a díjat, a várható káresemény valószínűségének és összegének számításba vételével.
- Esetlegesen a káresemény bekövetkezésekor, két okból:
    - Csökkentse a várható kárkifizetést a korábbi káresemények fényében, mivel, ha egy alkatrészt már javították, a biztosító jogosult csak részösszeg kifizetésére, hiszen nem eredeti gyári alkatrész cserél, hasonló a helyzet a fényezéssel.
    - Elkerülni az esetleges biztosítási csalásokat, mivel sokszor a csalók egy káreseményt több biztosító felé is megpróbálnak elszámolni úgy, hogy törött alkatrészeket egy hasonló színű és típusú autóra felszerelnek, így az eltérő rendszámmal, más biztosítónál CASCO szerződéssel rendelkező gépjárműre ott is bejelentenek egy káreseményt.

3.2.1 Háttér

A nagyvállalatok (példaként említve több biztosítót és bankot, amelyek belső rendszerét személyesen ismerem) a saját IT megoldásaikat a 80-as években kezdték el fejleszteni, először jellemzően külső szolgáltatókkal. Minden leányvállalat saját belső rendszert épített, amelyekben a vásárlók, a partnerek adatait tárolta, hogy segítse a számlázást, ügyfélkezelést, vagyis az általános napi munkát. Ez a 90-es években még megfelelő szinten tartotta a cégek működését, azonban magas élőmunka költséggel járt, hiszen a teljes adminisztráció kézzel zajlott, az informatikai üzemeltetést is belső kollégákkal kellett megoldani. A 2000-es évek végére egy tagolatlan, átláthatatlan,

sztenderdekkel nem rendelkező, országonként eltérő, nehezen kezelhető, ám mindösszesen robosztus rendszert hozott létre egy-egy cég, amelyben az egymás közötti kommunikáció és a központ felé történő beszámolók nehezen követhetők, egységesítés híján pedig teljesen eltérő mutatószámokat generálnak. Mindemellett, mivel a technológiai fejlődés megkívánta minden országban a folyamatos fejlesztéseket, azokat maguk a tagvállalatok végezték, sok esetben eltérő programnyelven, egyedi megoldásokkal, érdemi dokumentáció nélkül, bizonyos szolgáltatások esetében pedig olyan nem kívánt helyzetekkel, amikor egy rendszert csak egy informatikus kolléga ismert. Ezt felismerve a központok a 2010-es évek közepén elkezdték átszervezni az IT-t, létrehozva egy egységes, országokon átívelő szervezetet, közvetlen az anyavállalat által irányítva, ezzel elérve azt, hogy mindenhol egységes koncepció alapján dolgozzanak a kollégák, ezen keresztül kontrollálhatóvá tették a helyi eseményeket, mivel szinte minden informatika változtatást igényel manapság, jelentős költségeket tudtak megtakarítani a vállalati sztenderdek kialakításával és megelőzték azt, hogy az egyes országok néhány munkavállaló távozásával esetleg működésképtelenné válljanak bizonyos alrendszerek. Az általam bemutatott cég három éve indult el ezen az úton, azonban nagyon lassan halad, nem csak az üzleti oldal ellenállása miatt, hanem a jogszabályi környezetnek való megfelelősség miatt is, amely sok esetben teljes testreszabást igényel, ezzel jelentős többletmunkát okozva. Ehhez a hibrid rendszerhez alkalmazkodva alakították ki az innovációs folyamatukat, amelyet a következő pontokba részletes fogok kifejteni.

3.2.2 Ötlet

A hazai biztosítók már régóta szerettek volna létrehozni egy egységes adatbázist, annak ellenőrzésére, hogy megelőzzék a csalásokat, vagy csak, hogy teljeskörű információval rendelkezzen a gépjárművekről, amelyekre ajánlatot adnak, vagy amelyekkel kapcsolatban káreseményt rendeznek. Ennek azonban jogi és versenytilalmi akadályai is voltak, mivel nem oszthattak meg hivatalos úton ilyen információkat egymással, a gépjármű életút program is csak másfél éve indult el, és a vállalatok nagy része csak a kötelező gépjármű felelősségbiztosítás keretében kifizetett károk adatait viszik fel, hiszen nem szeretnék, ha a konkurencia akár belső adatokhoz férhetne hozzá ezen keresztül. Van azonban egy külső partner, amely a gépjármű

káreseményekkor felmerülő költségek kalkulálására fejleszt egyedi szolgáltatást, így rendelkezik egy olyan átfogó adattartalommal, amit megfelelően szűrve, a személyes és biztosítói adatok eltávolításával tökéletes háttere lehet egy ilyen szolgáltatásnak. Ennek részleteit azonban a beszállítóval közösen szükséges kidolgozni.

3.2.3 Globális Innovációs Fórum

Amennyiben egy tagvállalat nagyobb informatikai kiadást vagy fejlesztést tervez, egy egységes folyamaton szükséges azt végig vinni, amelynek első lépése a „Globális Innovációs Fórum" nemzetközi nyelven Global Change Forum[38], azaz GCP. A fórum legfontosabb feladata, hogy kiszűrje azon ötleteket, amelyek nem reális elvárásokon alapulnak, vagy amelyek üzleti szempontból nem állják meg a helyük. A folyamat első pontja, hogy Projekt Menedzsert és Architektet rendelnek a feladathoz, akik a várható üzleti tervtől függetlenül elkezdik kialakítani a várható költségek összegét és struktúráját, valamint kialakítják a megvalósíthatósági opciókat és előkészítik a projektet a vezetői döntésre, ahol egységes képet kaphat a vezetés az előnyökről és a hátrányokról, a várható bevételekről és kiadásokról. A folyamat megindításával kapcsolatban szigorú szabályokat fektetett le a vállalat, hogy mindenképp megelőzzék a meglehetősen drága informatikai erőforrások esetleges pazarlását. Ezek közé tartozik, hogy az egyes tagországok csak havonta egy üzleti igényt adhatnak le, vagy, hogy a prioritási lista élére csak jelentős bevételi potenciállal rendelkező igénnyel kerülhetnek. Ezzel kapcsolatban fontos megjegyezni, hogy figyelembe kell venni a vállalat számára rendelkezésre álló erőforrások mennyiségét és azonnal tájékoztatni a fontossági sorrend esetleg változásának eredményére a vezetőséget, például, ha egy újabb projekt elsődlegességet élvez másokkal szemben, akkor a további feladatokat több hetes vagy akár hónapos csúszást szenvedhetnek. Ennek átmeneti megoldásaként két fogalmat határoztak meg:

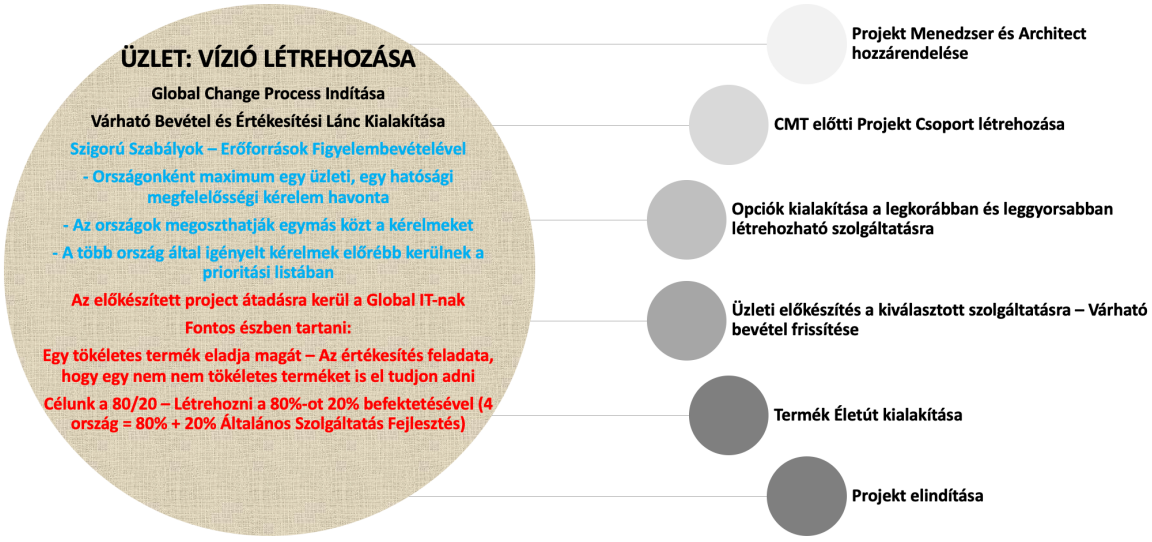
POC, azaz Proof of Concept: Ez tulajdonképpen azt jelenti, hogy a nagyobb projekteket egy úgynevezett koncepció érvényesítés indít el, azaz, hogy amíg nem biztos az üzlet abban, hogy valamilyen projekt jelentős bevételt hozhat, addig egy kis számú vevőn azt teszteli, ami csak minimális szolgáltatást igényel akár jelentős

[38] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Change Process

manuális munkával. Az a minimálisan elvárható szolgáltatás adja a következő fogalom alapját.:

MVP, azaz Minimum viable Product: Minimálisan leszállításra került szolgáltatás. A feladata az, hogy megvizsgálja élesben egy üzleti modell alkalmazhatóságát és felmérje a piaci igényt. Amennyiben sikeresen vizsgázik, lefejlesztésre kerül a végleges verzió, ami már tartalmazza a szükséges automatizációkat és alkalmas arra, hogy nagy számú partnert kiszolgáljon.

Az alábbi, vagyis a 6. Ábra az üzleti víziót és annak indulását foglalja össze, amely előzetesen már azt is tartalmazza, hogy a vállalat informatikai részlege milyen elvárásokat támaszt az erőforrások szűkössége miatt. Itt fontos kiemelni az ún. prioritási listát, amelyre, mint projekt a későbbiekben majd felkerül az ötlet, az alapján amilyen bevételi tervet a jobb oldalon található folyamat eredményeképp az üzlet felmutat.



Ábra 6.: Biztosító - Az üzleti vízió kidolgozása (Forrás: Saját munka)

3.2.4 Üzleti elemzés

A Projekt Menedzser feladata bevonni az Üzleti Elemzőt, hogy meghatározzák a várható nyereséget (vö. információs többletértéket), vagy a várható kár kifizetés csökkenést, esetleg azt a potenciált, amivel megelőzhetik azon gépjárművek CASCO szerződéseit, amelyek esetében magas a biztosítási csalás lehetősége. Az üzleti elemző

eredményei az alábbiakat mutatták abban az esetben, ha 60.000 darabot a vállalói, 36.000 darabot pedig kár oldalon használ fel a biztosító.

3.2.4.1 Vállalói oldal:

Kötelező Gépjárműfelelősség Biztosítás: A Kötelező Gépjárműfelelősség Biztosítás nem része a projektnek, mivel ott a biztosítónak nincs jogi lehetősége elutasítani egy szerződést, amennyiben az ügyfél őt választja.

CASCO Biztosítás: 60 napja van elállni a biztosítónak a szerződéstől.

- Éves CASCO ajánlatadások darabszáma jelenleg: 550.000 db
- Nyertes CASCO ajánlatok éves darabszáma (Ajánlat, amelyből szerződés lesz): 95.000 db (Tartalmazza mind az új gépjárműveket, mint az egyéb biztosítóktól átvett – megnyert kötvényeket)
- Az ebből reálisan lekérdezhető autók száma (Mivel minden autó megvizsgálására nem kapott elegendő büdzsét a projekt): 60.000 db/év (havi 5000 db/hónap).
- Az előzetes tesztek alapján hozott találati arány: Az éles indulás előtt a beszállító készített egy kimutatást, amelyben 100.000 db korábbi években történt CASCO szerződést, az azokhoz tartozó későbbi csalásokat, esetlegesen nem érzékelt nagyobb sérüléseket vizsgálta annak függvényében, hogy az a biztosító újra kifizette: 1/20-hoz – Tehát minden 20-ik lekérdezés okoz költségmegtakarítást hosszútávon.
- Átlagosan egy gépjárműre jutó megtakarítás összege (a gépjárművek átlagkára csalás esetén): 650.000 Ft/káresemény.
- A várható kieső CASCO díj az eldobott szerződéskötések okán (hiszen nem csak, hogy nem lesz kárkifizetés ezen gépjárművekre, de bevételt sem termelnek, mivel nem köt szerződést velük a biztosító): 150.000 Ft/káresemény.

Ennek eredményét az alábbi Táblázat 7. tartalmazza:

Forma	Érték	Képlet
Ajánlatok Száma	95.000 db	A
Felhasználható Darabszám	60.000 db	B
Találati Arány	1/200 (0,5%)	C
Találatok Várható Száma	300 db	D=B*C

Forma	Érték	Képlet
Átlagos Megtakarítás	650.000 Ft	E
Átlagos Bevételkiesés	150.000 Ft	F
Ezek Különbözete	500.000 Ft	G=E-F
Várható Megtakarítás	**150.000.000 Ft**	**H=D*G**

Táblázat 7.: Biztosító – Vállalói oldal várható megtakarítása (Forrás: Saját gyűjtés – Melléklet 1. Szekunder adatok katalógusa)

Tehát a várható megtakarítás nagyságrendi értéke 150.000.000,- Ft a biztosító vállalói oldalán (vö. Táblázat 7.).

3.2.4.2 Kár oldal:

Kár oldalon a biztosító azzal számol, hogy bizonyos esetekben levonhat a kártérítés összegéből, mivel korábban esetleg ugyanazon alkatrészek sérültek, vagy kerültek fényezésre, mint az aktuális balesetben is, így ún. avulásként az külön levonásra kerülhet a hazai jogszabályok miatt. Ezt az alábbi teszt és háttérszámítás foglalja magába:

Forma	Érték	Képlet
Éves CASCO károk száma	96.901 db	A
Lekérdezhető darabszám	36.000 db	B
Releváns használati arány (amelyben használható eredményt kap a szakértő)	1/20 (5%)	C
Használható darabszám	1.800 db	D=B*C
Átlagos CASCO kárkifizetés	420.160 Ft	E
Átlagosan elérhető kifizetéscsökkenés a szolgáltatás eredménye alapján	8%	F
Káreseményenként elérhető átlagos kifizetés csökkenés	33.612 Ft	G=E*F
Éves szinten realizálható kifizetés csökkenés	**60.501.600 Ft/év**	**H=D*G**

Táblázat 8.: Biztosító – Kár oldal várható megtakarítás (Forrás: Saját gyűjtés – Melléklet 1. Szekunder adatok katalógusa)

Tehát a várható kárkifizetés-csökkenés 60.501.600,- Ft. (vö. Táblázat 8.).

Ezek alapján a teljes megtakarítás éves várható összege 210.501.600,-Ft, ami három évre összesen (inflációs hatások nélkül) 631.504.800,-Ft.

3.2.5 Megvalósíthatósági vizsgálat

Az ötlet elindítása után, miután Projekt Menedzsert és Architektet rendeltek hozzá, elindul a megvalósíthatósági terv kialakítása. A két kolléga megvizsgálja a már elérhető szolgáltatásokat, megnézi, hogy esetleg azok kisebb változtatásaival megoldható-e az adott igény, ha erre nincs lehetőség kialakítja a belső fórumot a fejlesztő és egyéb informatikai kollegák bevonásával, hogy megtervezze a nagyobb változtatásokat, amennyiben pedig ez sem jelent megoldást, külső szolgáltatók után kutat, akik akár kész megoldással rendelkeznek az adott problémára, esetleg hasonló rendszert üzemeltetnek bármely konkurens vállalatnál. Ezek kombinációjaként kidolgoznak több megoldási javaslatot is (vö. lehet-e minden megoldás másként egyforma, annak lényege a projekt céljának elérése), amelyet a Projekt Menedzser előkészít a Stratégiai Menedzsment Triázsra, ahol az egyes országok vezetői a kész tervekből kiválasztják a számukra legmegfelelőbbet. Mivel a jelen szolgáltatás valójában csak adat átadás-átvétel, majd annak megjelenítése egy felhasználó számára (tehát az ún. MVP-nek nem része az, hogy az adatot feldolgozza bármilyen automatizmus), így csak egy megoldási mód lett felvázolva, valamint egy nagyságrendi becslés az elsődlegesen leszállítandó, valamint a teljes szolgáltatásról, amely az alábbiak szerint néz ki:

- Szoftverfejlesztői erőforrás a vállalói oldalon: 10 munkanap (Külső megrendelés)
- Szoftverfejlesztői erőforrás a kár oldalon: 10 munkanap (Külső megrendelés)
- Lekérdezés darabára: 300 Ft/db ÁFA-val
- Várható lekérdezés darabszám havonta: 8000 db/hónap
- Várható lekérdezés éves darabszáma: 96.000 db/év

Mindez költség oldalon

- Fejlesztés: 230.000,- Ft-os fejlesztői napidíjjal számolva: 4.600.000,- Ft
- Éves várható költség az adatbeszállítónak: 28.800.000,- Ft/év

Összes költség (inflációs hatások nélkül) az első 3 évre: 91.000.000,- Ft

3.2.6 Stratégiai Menedzsment Triázs

„A döntés lényegében azt jelenti, hogy a környezetből és a vállalat korábbi működéséből származó információk alapján olyan új információkat hoz létre, amelyek a vállalati működést biztosítják, az erőforrásokat mozgásba hozzák."[39]

A Stratégiai Menedzsment Triázs-t hetente egyszer rendezi meg a Global IT-t, amelynek feladata, hogy az egyes országok vezetői meghozzák a döntést a felajánlott megvalósíthatósági tervekből. Jelen esetben az elfogadott fejlesztési terv két informatika rendszer közötti kommunikációról szól, amelyek egyenként 15-15 napnyi fejlesztési erőforrást igényelnek, első esetben a CASCO szerződés vállalási folyamatában egy döntéshozatali pontként, míg a második esetben a kárrendezés során a gépjárműipar szakértőt informálva a korábbi káresemények részleteivel kapcsolatban. Mivel automatizmus nem épül a folyamatra, így visszajelző rendszer sem kerül kiépítésre, a később mérhetőség szempontjából csak az egyes esetek elemzése marad. Ezekre tekintettel a vezetők két korábban kialakított értéket hasonlítanak egymáshoz:

Teljes várható megtakarítás a következő 3 évre: vö. 631,5mFt

Összes költség a következő 3 évre: vö. 91mFt

Mivel a két érték egyértelmű megtérülést és ezáltal a vállalat számára nyereséget hordoz, a szolgáltatás és a fejlesztés is megrendelésre kerül. Fontos azonban megjegyezni, hogy a vállalat döntéselőkészítő folyamata nem tartalmazza, hogy alternatívákat is kidolgozzanak egy adott eredmény elérésére, ami egy irányba tereli a döntéshozókat.

3.2.7 Megrendelés

Ha a Stratégiai Menedzsment Triázs döntésének megfelelően a szolgáltatás belső fejlesztéséhez szükséges megrendelés megtörtént, akkor a Projekt Menedzser elindítja a fejlesztés leszállításához szükséges folyamatot. Ennek első lépése a részletes tervezés.

[39] Chikán Attila, 2006, 343

3.2.8 Részletes tervezés

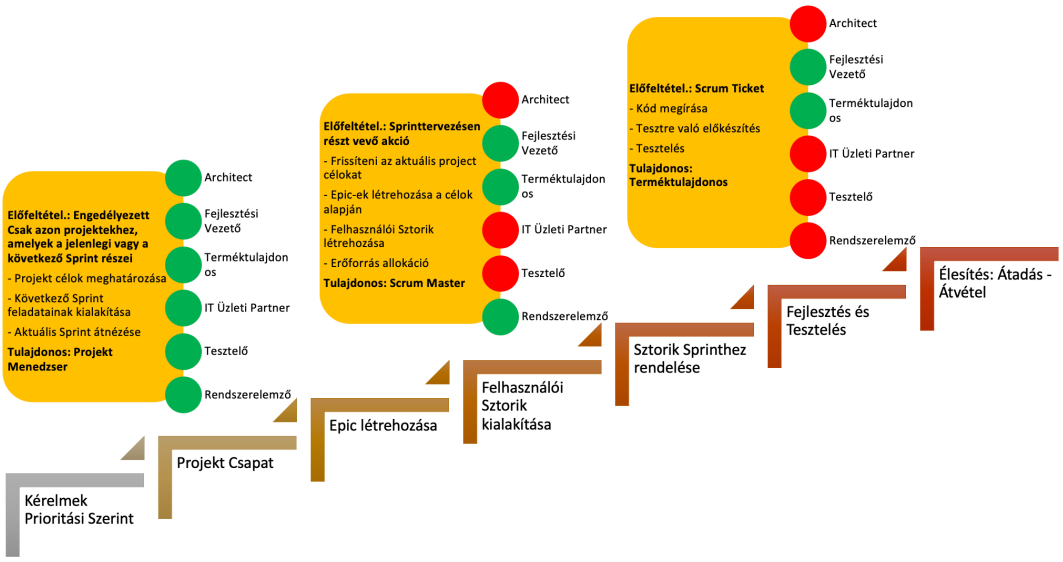
A részletes tervezés során a Projekt Menedzser felállítja a Projekt csapatot, amely informatikai fejlesztés esetén a következők munka- és feladatköröket tartalmazza:

- IT Architect:[40] A már korábban felvázolt megoldási javaslatot finomítja, kialakítja a szükséges technológiai folyamatokat, az egyes programok közötti adatáramlást. Feladata csak a tervezésre terjed ki.
- Fejlesztési Vezető: Feladata, hogy kód szinten megértse az Architekt által leírt folyamatot és megtalálja a megfelelő fejlesztő kollégákat azok létrehozására. A folyamat hátralevő részében irányító szerepet tölt be.
- Terméktulajdonos: Mivel a vállalatok az egyes szolgáltatásokat tekintetében hosszútávon gondolkodnak, ezért minden programhoz egy tulajdonost rendelnek, aki nem csak annak létrehozásában vesz részt, de minden a későbbiekben szükséges változtatást intéznek, valamint üzemeltetik és karbantartják az adott rendszert.
- IT Üzleti Partner: Arról szükséges biztosítani a fejlesztés folyamatát, hogy az végül minden üzleti igénynek megfeleljen.
- Tesztelő: A szolgáltatást fogja tesztelni az üzleti elvárásoknak megfelelően.
- Rendszerelemző: Felkészíti a technikai rendszereket a változtatásokra, létrehozza a megfelelő környezetet, allokálja a technikai erőforrásokat.

3.2.9 Fejlesztés – Szállítás

A szállítás folyamatát az alábbi, vagyis a 9. ábra szemlélteti, mely tartalmazza már a technológiai előkészítés lépéseit is:

[40] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: IT Architect



Ábra 9.: Biztosító – Fejlesztés és szállítás folyamat (Forrás: Saját munka)

3.2.9.1 Prioritás lista:

„Az innovációs ötletek hatékony szelektálása igen nagy jelentőségű folyamat: egyrészt azért, mert az innovációs láncban előre haladva egyre nagyobb költségbe kerül az esetleges tévedés; másrészt vigyázni kell arra, hogy minél inkább csökkentsük az ígéretes innovációk elutasításának veszélyét."[41]

Mint minden vállalatnak, a jelen helyzetben elemzésre kerülő biztosítónak is szűkös a fejlesztési kapacitása, még abban az esetben is, ha külső szolgáltatókat vesz igénybe. Ezért az egyes projektek versengenek egymással a végrehajtás tekintetében. Egy ekkora projekt jellemzően elég hamar a lista élére kerül, így a fejlesztés korán elindulhat.

3.2.9.2 Projekt csapat:

A szállítási és fejlesztési folyamat indításakor a Projekt Menedzser egy közös megbeszélésen jelzi az IT-val szembeni elvárásokat, felvázolja az Architekt által előkészített, fejlesztendő folyamatot és kialakítja az Projekt csoportot. A megbeszélés legfontosabb eredményeként a fejlesztői csapat kap egy egységes képet arról, mi a feladata.

[41] Chikán Attila, 2006, 275

3.2.9.3 Epic létrehozása:

A következő lépés a Scrum Master vagy a Terméktulajdonos által az ún. Epic-ek létrehozása. Ezek összefoglalják a nagyobb fejlesztendő szolgáltatásokat és folyamatokat, amelyekkel a kódolással foglalkozó kollégák tudnak dolgozni. A későbbiekben ezen funkciókról kell rendszeres státusz riportokat adnia a Projekt Menedzsernek, hogy megbizonyosodjon a megfelelő haladásról. Az Epic valójában azt tartalmazza, hogy az egyes szolgáltatásokban a végfelhasználó milyen akciókat kíván végrehajtani, hogy elérje a szükséges eredményt. Például kár oldalon a szakértő rákattint egy gombra, aminek eredményeképp a felhasználói felületén megjelenik egy információs tábla a korábbi káresemények adataival. Az egyes Epic-ek a JIRA nevű szolgáltatásban kerülnek dokumentálásra, egy-egy linkkel az Confluence projekt leírására, egy koherens rendszert alkotva.

3.2.9.4 Sztorik kialakítása:

Miután az egyes fejlesztők megkapták a saját Epic-jeit, létrehozzák a hozzájuk tartozó ún. Sztorikat. Ezek tulajdonképpen kisebb, néhány órás kódolások, amelyeket részletesen dokumentálnak és előkészítik a hozzájuk tartozó rendszerek a változtatásokra. A sztorik tartalmazzák az egyes szolgáltatások közötti kommunikációt, a felületek leírását, dizájn ötleteket stb. Esetünkben a káros kollégánál 4 sztori kerül kialakításra:

- A gomb elkészítése, amelyre a felhasználó rá tud kattintani.
- A háttérben a technológiák közötti kommunikáció, amelyben a biztosító rendszere meghívja a beszállító adatbázisát és lekéri alvázszám alapján az elérhető korábbi káresemények adatait.
- A visszatérő adatok egységes megjelenítése egy felületen a szakértő számára.
- Az eredmények elmentés PDF és Excel formátumban a későbbi felhasználhatóságra.
- Naplózás kialakítása, meghatározása, részletei, adattartalma stb.

3.2.9.5 Sztorik Sprinthez rendelése:

A Sztorikat ezután az ún. Sprintekhez rendelik. Egy Sprint 10, 15 vagy 20 munkanapból áll a vállalat belső fejlesztésének döntése szerint. Mivel két 10 napos

fejlesztésről beszélünk, amit külső partner szállít és két fejlesztő is rendelkezésre áll, így az első elérhető Sprint részeként elkészül a fejlesztés.

3.2.9.6 Fejlesztés és tesztelés:

A fejlesztő kollégák megkapják a számukra előkészített Sztorikat, amiket lekódolnak a belső és a külső teszt rendszereket. Maga a kódolás tartalmazza a nagyságrendi tesztelést, ahol a fejlesztő is átnézi, hogy az előírásoknak megfelelően működik-e a változtatás. Ha elkészült a változtatással azt átadja tesztelésre. A tesztelőknek manapság már rendelkezniük kell fejlesztői tudással is, hogy mind a regressziós, tehát visszamenőleges, ún. regresszív tesztet (megbizonyosodjanak arról, hogy a korábban a felhasználónak átadásra került funkciók is működnek) végeznek, illetve automatizált eredményeket is kiértékelnek, aminek során manipulálják a rendszert akár úgy, mintha több ezer felhasználó használná egyszerre. Ha ezen tesztek eredménye sikeres, az új kód átadásra kerül a belső szerverüzemeltetésre, aki élesíti azt.

3.2.9.7 Élesítés:

Élesítés során a vállalat belső üzemeltetése feltölti az új kódot az éles Rendszerre, majd kiadja az utolsó tesztelési lehetőséget is a megadott csapatnak. A frissítés során az esetek jelentős többségében számítani kell rövid leállásokra a szolgáltatásban, így arról előzetes tájékoztatni szükséges a felhasználókat.

3.2.10 Visszajelzés

„A tervezett és tényleges pénzáramlás összehasonlítása hasznos, de nem derül ki belőle, hogy a projekt a költségvetésen belül vagy azt túllépve készül el. Ahogy, hogy pontos képet kapjunk a költségteljesítményről, minden befejezett feladat tervezett és tényleges költségeit össze kell hasonlítani. Ez az ún. megtermelt értéken alapuló jelentés módszerével tehetjük meg.” [42]Az elkészült és élesített szolgáltatást a vállalat üzleti oldala rendszeresen szondázza, elemzi annak sikerességét, azonban ezt többféle módszerrel is szükséges megtennie. Ennek marketing és egyéb vetületei is léteznek,

[42] Verzuh Eric, 2006, 336.

bizonyos esetekben kérdőívként jelenik meg, vagy szúrópróba-szerűen kérdezik ki a szolgáltatást használókat. Mindemellett figyelik az elért eredményeket is, amelyek mérföldkövet jelentenek a szolgáltatás további alakításában. Jellemzően az ilyen nagyobb volumenű beszállítói szerződéseket évfordulóra kötik, hogy sikertelenség esetén a leghamarabb lezárhassák azt, így a legkisebb veszteséget okozva a vállalatnak.

3.2.11 Eredmények és kritika

A nagyvállalat elemzését a 3-as fejezetben kidolgozott metódus alapján készítettem el, amelyet az alábbi Táblázat 10. tartalmaz:

Folyamat	Szempont	Megfelelősség	
		Eredmény	Kritika
Általános	Felelősségi körök	Jó	A vállalatban megtalálhatók főbb funkciók és azok megfelelően működnek, tudják a feladatokat és átlátják a folyamatokat.
Általános	Folyamat	Fejlesztendő	Az innovációs folyamat nem egyértelmű, az egyes elágazások sok esetben problémát okoznak, a határidők sok esetben csúszhatnak emiatt, a projekthez allokált erőforrásokról nincs egységes kommunikáció.
Általános	Jog	Jó	A jogászok a nagyvállalatoknál nagyon fontos szerepet töltenek be, így minden projektben részt vesznek, a jogi megfelelősség a legfontosabb a büntetések elkerülése végett és a jó hírnév megőrzése érdekében.
Üzleti elemzés	Kiadások	Fejlesztendő	Kiadási oldalon a pénzbeli kiadásokkal számolnak, azonban a belső erőforrásoknál csak a legszűkebb, fejlesztői költséget veszik figyelembe, miközben egy projekten sok esetben 30-40 ember is dolgozik.
Üzleti elemzés	Bevételek	Jó	A nagyvállalatoktól a legtöbb esetben akár évekre előre elvárják, hogy pontos bevételi tervekkel rendelkezzenek, így erre külön figyelmet fordítanak.
Döntés	Vezetői kompetenciák	Fejlesztendő	A döntéshozók egységes projekt csoportot alkotnak a döntést közösen hozzák meg, a jó előkészítés miatt az esetek jelentős többségében megfelelő döntés születik. A fejlesztendő eredményt azért kaptam, mivel nem álltak rendelkezésre alternatívák és azok előnyei és hátrányai nem jelentek meg a döntéselőkészítésnél.
Megrendelés	Általános	Jó	Az üzleti elvásárok megfelelően átadásra kerülnek a fejlesztőknek, a dokumentáltság jó, a vállalat rendelkezik a megfelelő sztenderdekkel.
Fejlesztés	Módszertan	Jó	A fejlesztés Scrum módszertan alapján működik, a kollégák tudják a feladatuk.
Fejlesztés	Prioritás	Fejlesztendő	A fejlesztői csapat jelentős erőforráshiánnyal rendelkezik, bár a prioritásokkal tisztában vannak, sokszor FIFO alapon dolgoznak, hogy a leghamarabb átadhassák az elkészült szolgáltatásokat, ahelyett, hogy a legnagyobb bevétellel kecsegtetőn dolgoznának, a fejlesztési tervek nem mutatnak 2 hónapnál távolabbra.
Fejlesztés	Dokumentáció	Fejlesztendő	Bár belső sztenderdek léteznek, az erőforrás hiány miatt a fejlesztés nem kerül megfelelően dokumentálásra, ez jelentősen lassítja a munkát
Fejlesztés	Technológia	Fejlesztendő	A vállalat éppen a felhőbe költözésen dolgozik, így szolgáltatásainak jelentős része még az előző generációs

Folyamat	Szempont	Megfelelősség	
		Eredmény	Kritika
			rendszereken fut, ami folyamatos felügyeletet igényel, A kódbázis már létezik, de nem minden applikáció képezi részét. A tesztelésre való átadás és a kézzel történik.
Fejlesztés	Hozzáértés	Jó	A fejlesztők naprakész tudással rendelkeznek, azonban motivációjukat jelentősen rontja, hogy egyszerre több projekten is kell dolgozniuk és folyamatos erőforrás hiánnyal küzdenek.
Fejlesztés	Tesztelés	Fejlesztendő	Jelenleg nincs automata tesztelés és a regressziós tesztelés is csak kézzel történik a jelentősebb funkciók átnézésével.
Fejlesztés	Élesítés	Fejlesztendő	Az élesítést a nem felhőalapú rendszereken még a fejlesztők végzik, a frissítés minden esetben szolgáltatás kieséssel jár.
Indítás	Átadás - Átvétel	Fejlesztendő	Az átadás-átvétel jól dokumentált azonban a végfelhasználó nincs felkészítve a használatra, valamint az üzemeltetők jellemzően csak az egyszerűbb feladatok megoldására képesek.
Visszajelzés	Mérőszámok	Jó	A Projekt Menedzser havi visszajelzést ad az új szolgáltatáshoz kialakított mérőszámokról és a hozzájuk tartozó pénzügyi adatokról.

Táblázat 10.: Biztosító – Kiértékelés eredménye (Forrás: Saját elemzés és vállalati partnerek visszajelzései)

3.3 Egy középvállalat egy projektjének bemutatása, eredményei, kritikája

Az általam bemutatni kívánt cég egy 80 fővel rendelkező gépjármű márkaszervíz három telephellyel, ebből egy Budapesten található, egy Szegeden, egy pedig Székesfehérváron. A javító rendelkezik jogosultsággal 3 nagyobb gépjármű márka új gépjárműveinek eladására, szervizelésére, valamint két helyen karosszériajavító és fényezőüzemmel. Éves szinten 850-900 új és 1500 gépjárművet értékesít, magánszemélyeknek 1773 alkalommal végez általános szervízt, míg flottakezelőknek 4300 esetben, biztosítási káreseményekkel kapcsolatban pedig 1100 alkalommal foglalkozik, javítás és fényezés tekintetében is. Ez nagyságrendileg 350 millió forintos nettó nyereséget jelent 2.8 milliárd forint bevétel mellett a 2019-es adatok alapján.

3.3.1 Háttér

A hazai középvállalatoknak jellemzően nincs lehetőségük arra, hogy saját informatikai kollégákat, mint belső erőforrás tartsanak fenn, vagy amennyiben időben felismerik ennek szükségességét, ez kimerül egy Projekt Menedzser (jellemzően IT vezető titulusban) és egy rendszergazda foglalkoztatásában. Ez a szűkös erőforrás

minden innovációra rányomja a bélyegét, amelyek az esetek döntő többségében külső kényszer hatására jönnek létre, amit a piaci dinamizmus a konkurencia vagy a nemzetközi partnerek indítanak. Jelen esetben egy újonnan a piacra lépő flottakezelő által bevezetésre kerülő automatizált folyamatot fogunk átnézni, amely eredményeképp jelentős többletmunkához juthat a mindegyik telephely, ezzel növelve bevételét és partner hálózatát:

3.3.2 Előkészítés

A projekt előkészítése a flottakezelővel való tárgyalások sorozatának hatására alakul ki, amelyben mindkét résztvevő kialakítja az igényeit és felkészül a másik fél igényeinek kielégítésére. Ez a javító szempontjából jelentős beruházással jár, mivel a flottakezelő elvárja tőle a legmagasabb színvonalú ügyfélkiszolgálást és az alapvető automatizmusok beépítését. Emiatt át kell alakítania a belső informatikai rendszerét az ún. DMS[43]-t (Dealer Management System – Márkaszerviz Menedzsment Rendszer), valamint a hozzátartozó kiegészítő szolgáltatásokat. Az előzetes szerződés három márkára terjed ki, ezek az Opel, a Peugeot, Toyota, Volkswagen, amely szűkíti a várható darabszámot. Ezen kívül még további költségeket is vállalni kell, amelyeket az üzleti elemzésben részletezek.

3.3.3 Üzleti Elemzés

„A kisvállalatoknak a nagyokkal szembeni gyengeségei általában az alábbi okcsoportokra vezethetők vissza:

- a vezetői képességek, ismeretek strukturális gyengesége ("vállalkozói ismeretek", kooperációs képességek stb.)

- korlátozott hozzáférés egyéb erőforrásokhoz (pl.: koreszrútlen technológia, nem megfelelő üzleti információk, állami támogatások nehezen hozzáférhetők stb.)”[44]

[43] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: CMS
[44] Dr. Gyulai László - Dr. Hanyecz Lajos - Járos Péter - László Csaba, 2002, 49.

3.3.3.1 Várható bevétel növekedés meghatározása:

A flottakezelő a szerződésért cserébe éves szinten 1500 szervízeseményt ígért a javítónak, azonban nem kiskereskedelmi alkatrészáron és piaci óradíjon kívánja javíttatni a gépjárműveit, hanem volumenkedvezményt vár el a javítótól. Az ehhez tartozó értékeket a Táblázat 11. foglalja össze:

Téma	Érték	Képlet
A javító által a piacon alkalmazott átlagos óradíj szervízesemények során	9.331 Ft	A
A flottakezelő által elvárt óradíj kedvezmény	20 %	B
A vállalt óradíj a flottakezelő kedvezményével	7.464 Ft	C=A*(1-B)
Egy átlagos szervízesemény munkaideje	2,2 óra	D
Várható munkadíj bevétel egy szervizesemény esetén	16.421 Ft	E=C*D
A munkadíj és a munkavállalónak kifizetett díj átlagos különbség óradíjban (Az összeg, ami a munkavállalónak kifizetett munkabér és a bevétel különbsége a flottakezelői órabérrel)	5.827 Ft	F
A javító egy szervizeseményre jutó nettó nyeresége a flottakezelői óradíjjal	12.819 Ft	G=D*F
A javító által szervizelt gépjárművek átlagos alkatrész költsége kiskereskedelmi áron	70.278 Ft	H
A flottakezelő által elvárt alkatrész kedvezmény	10 %	I
A flottakezelő irányába várható átlagos alkatrészköltség	63.250 Ft	J=H*(1-I)
A javító átlagos nyereséghányada az alkatészeken (Becslés)	38 %	K
Az ebből fennmaradt átlagos nyereség az alkatrészeken a flottakezelőnek adott kedvezmény után	28 %	L
Átlagos alkatrésznyereség a flottakezelői szervizeseményeken	17.710 Ft	M=J* L
Egy flottakezelői eseményen várható teljes nyereség (munkadíj és alkatrész)	12.819 Ft + 17.710 Ft = 30.529 Ft	N=G+J
1500 szervízre tervezhető teljes javítói nyereség	**45.793.500 Ft**	**O=1500*N**

Táblázat 11.: Középvállalat – Várható bevételnövekedés (Forrás: Saját gyűjtés – Melléklet 1. Szekunder adatok katalógusa)

Tehát a flottakezelői szerződéssel járó potenciális nyereség vállalati szinten évente 45.793.500,- Ft (vö. Táblázat 11.). A javítók jellemzően nem gondolkodnak 3 éves megtérülésben, mivel rövidtávú célokkal rendelkeznek, méretükből fakadóan.

3.3.3.2 Kiadások:

A szerződés első évében várható kiadásokat a Táblázat 12. tartalmazza (a javítással és szervizeseménnyel kapcsolatos nettó nyereség esetében már számoltunk a kiegészítő javítói költségekkel, annak eredményeképp azonban arra jutottak, így az tartalmazza az esetleges javító kolléga felvételét):

Téma	Érték	Képlet
A külső szoftverfejlesztő partnertől kapott ajánlat a flottakezelővel való integrációra	1.800.000 Ft	A
Egy adminisztrátor munkatárs átlagos órabére	3.800 Ft/óra járulékokkal	B
Egy szervízeseményre jutó átlagos munkaidő (Tartalmazza az ügyfélfogadást, kezelést, számlázást és minden egyéb időt)	0,8 óra/esemény	C
Átlagos adminisztrációs költség szervíz eseményenként	3040 Ft/esemény	D=B*C
Az 1500 várható eseményre jutó teljes adminisztráció költség	4.560.000 Ft	E=D*1500
Egyszeri plusz kiadás a raktárkészlet felkészítésére, hogy azonnal kiszolgálhassak az érkező javítandó gépjárműveket mind a három telephelyen	8.000.000 Ft	F
Az első évben várható teljes költség	**14.360.000 Ft**	**G=F+E+A**

Táblázat 12.: Középvállalat – Várható kiadások (Forrás: Saját gyűjtés – Melléklet 1. Szekunder adatok katalógusa)

Tehát a várható első éves költség 1500 szervízesemény esetén 14.360.000,- Ft (vö. Táblázat 12.).

„A megtérülési idő az egyik legelterjedtebb, nem diszkontáláson alapuló döntési kritérium. Arra a kérdésre ad választ, hogy a kezdő tőkebefektetés a beruházás révén

képződő pénzáramlásokból hány év alatt térül meg” [45] ennek eredménye tulajdonképpen azt jelenti, hogy amennyiben egy éven belül kívánja kitermelni a költséget vállalat, csak el kell osztania a kezdő befektetést, a várható évi nettó pénzárammal, amennyiben ez 12-nél alacsonyabb, a projektet mindenképp érdemes elindítani.

3.3.4 Megrendelés

A vállalat vezetőinek döntése a várható nyereség és kiadás eredményeképp a megrendelés irányába dőlt el, mivel a várható 45.793.500,- Ft-os bevétel többszörösen meghaladja a 14.360.000,- Ft-os várható költséget. Azonban fontos számolnunk az időtényezővel, ami szerint a költséges és a várható bevételek eltérő időben realizálódnak. A kiadási oldalon keletkező elsődleges befektetés, tehát az 1.800.000,- Ft-os fejlesztés, valamint a 8.000.000,- Ft-os alkatrész raktárkészlet beruházás azonnal megjelenik, míg a munkadíjak, valamint a bevételek a várható bevezetéshez szükséges 3 hónap után kezdenek csak megjelenni. Ezért valójában az első évben a teljes kiadási oldal megjelenik, azonban a bevételnek csak ¾-ed része. Azonban a következő években már nem lesz szükséges újabb, hasonló kiadásra. Így a számok az alábbiak szerint alakulnak, amely alapján a cég tulajdonosai egyértelműen a projekt elindítása mellett döntenek.

3.3.5 Eredmények és kritika

A középvállalatnál a már a 3-as fejezet elején bemutatott értékelő táblázatot használtam alapul, melynek eredménye az alábbi Táblázat 13.:

Folyamat	Szempont	Megfelelősség	
		Eredmény	Kritika
Általános	Felelősségi körök	Fejlesztendő	A hatáskörök nem egyértelműek, sok szerepet egy kolléga kell, hogy ellásson, ez részben a vállalat méretére vezethető vissza.
Általános	Folyamat	Fejlesztendő	Az innovációs folyamat létezik, azonban ritkán használják, nem egyértelműek a folyamat egyes pontjai, nem rendelkeznek folyamat leírással.
Általános	Jog	Fejlesztendő	Jogi oldalról a kisebb vállalatok általában csak arra fektetnek hangsúlyt, hogy a jó hírnevüket vagy egy esetleges büntetést elkerüljenek, így itt csak a megfelelő eredményt kaphatja a vállalat.

[45] Illés Ivánné, 2007. 49.

Folyamat	Szempont	Megfelelősség	
		Eredmény	Kritika
Üzleti elemzés	Kiadások	Nem megfelelő	A kiadásokat csak pénzbeli összegként mérik, nem foglalkoznak a belső erőforrások vagy egyéb várható felmerülő költségek részleteivel.
Üzleti elemzés	Bevételek	Nem megfelelő	Részletesebb piaci vizsgálat nélkül, a flottakezelő által ígért darabszámokkal számolnak, amelyre a szerződés nem nyújt garanciát.
Döntés	Vezetői kompetenciák	Jó	A döntéshozatal a hiányos előkészítés miatt több sebből vérzik, azonban a vezetők a számukra elérhető adatokat megfelelően használják fel.
Megrendelés	Általános	Nem megfelelő	Projekt Menedzser hiányában csak az üzleti dokumentumok kerülnek átadásra külső fejlesztőnek.
Fejlesztés	Módszertan	Fejlesztendő	A külső fejlesztő válogat az egyes projektek közt, így a megrendelő nem számolhat azzal, hogy mikorra készül az általa megrendelt fejlesztés.
Fejlesztés	Prioritás	Nem megfelelő	Mivel külső fejlesztővel dolgozik a vállalat, erre egyáltalán nincs hatása.
Fejlesztés	Dokumentáció	Nem megfelelő	A szoftverfejlesztéssel foglalkozó beszállítók általában nem készítenek részletes dokumentációt, ezzel is biztosítva maguknak azt, hogy a megrendelő náluk maradjon.
Fejlesztés	Technológia	Nem megfelelő	Technológiailag elavult rendszereket használ a vállalat, amelyeken a fejlesztés jóval hosszabb idő igényel
Fejlesztés	Hozzáértés	Nem megfelelő	Dokumentáció hiányában először meg kell ismerni a szolgáltatás aktuális állapotát, majd utána lehet folytatni a kódok írását, azonban mivel külső fejlesztőt vesz igénybe a projekt, erre szintén nincs hatása, nem versenyeztet.
Fejlesztés	Tesztelés	Nem megfelelő	Nincs klasszikus tesztelést, azt a megrendelő végzi.
Fejlesztés	Élesítés	Nem megfelelő	Az élesítést a fejlesztő végzi, a tesztelés maga a megrendelő.
Indítás	Átadás - Átvétel	Nem megfelelő	Mivel a vállalatnak nincs belső IT üzemeltetése, minden hibával a beszállítóhoz fordulnak.
Visszajelzés	Mérőszámok	Fejlesztendő	A mérőszámok kidolgozásra kerültek, azokat bizonyos időközönként megvizsgálják.

Táblázat 13.: Középvállalat – Kiértékelés eredménye (Forrás: Saját elemzés és vállalati partnerek visszajelzései)

4 KÖVETKEZTETÉSEK, JAVASLATOK

Javaslataimat a Táblázat 5. folyamatának lépései alapján állítottam össze.

4.1 Általános innovációs és változási folyamatok

Ahogy a Táblázat 5. folyamat oszlopának általános sorai alatt jeleztem, majd a Táblázat 10. és a Táblázat 13. táblázatban részleteiben elemeztem is a nagy, illetve középvállalatnál, az általános innovációs folyamatok több helyen is hiányosak. Ezek megelőzésére szükséges általános sztenderdek bevezetése minden vállalat számára. Az alábbiakban foglaltam össze azon sztenderdeket, amelyek egy nagyvállalattól minimális szinten elvárhatók:

- Feladat és munkakörök: Bár a biztosítói elemzésben látható, hogy minden munkakör megtalálható a vállalaton belül, a feladat- és felelősségi körök nem egyértelmű meghatározása miatt jelentősen lassul a végrehajtás (vö. 3.2.11.).
- Jogi háttér: A biztosító nagyon jól szerepelt a jogi kérdésekben, azonban ennek elsődleges oka, hogy az MNB (Magyar Nemzeti Bank) mint felügyeleti szerv, évente akár többször is ellenőrzi őket. Azonban a hazai középvállalatoknál, legalábbis az általam megismerteknél, sok esetben még személyes, GDPR hatálya alá tartozó adatok tárolásával vagy kezelésével sem foglalkoznak kellő gondossággal (vö. Táblázat 13.).
- Folyamat: Ahogy az angol mondás is tartja „One size fits all” alapon szükséges találni egy egységes folyamatot, amely során egy kiválasztott személy, jellemzően a Projekt Menedzser, minden releváns, vállalaton belüli szakterület véleményét kikéri és elemzi, majd az alapján tervezi meg a végrehajtást. Ellenkező esetben hiányos lesz az elkészült szolgáltatás (vö. Táblázat 10. és Táblázat 13. nem megfelelő eredményei).
- Ki, mit, mikor és miért hozzáállás: A szoftverfejlesztés alapja az úgynevezett felhasználói sztori, ami segít abban, hogy a felhasználó helyébe képzeljük magunkat, aki leírja nekünk, hogy igazából mit is szeretne elérni/látni/csinálni, tehát milyen helyzetben, mikor és milyen ok miatt szeretne egy bizonyos eredményhez jutni. Ennek hiánya megmutatkozik mind a két vállalat esetében,

pedig jelentősen felgyorsíthatná a folyamatokat, valamint minőségibb szolgáltatást hozhatna a felhasználóknak (vö. 3.2.11. és 3.3.5), ha a user-story-alapú[46] fejlesztést meg lehetne erősíteni.

- Kommunikáció: Ahogy a két vállalat kritikájában (vö. 3.2.11. és 3.3.5) is kifejtem, a kommunikációs csatornák nem megfelelően kerülnek kihasználásra. Vagyis pl. azt előre nem határozták meg, kik az értesítendő felek a projekt csapaton belül és azon kívül, a megrendelői és felhasználói oldalon. Ezek előkészítése és a velük való folyamatos munka a Projekt Menedzser feladata.

4.2 Üzleti elemzés

Az üzleti elemzés alapvetően a várható kiadási és bevételi oldal elemzéséből áll, amelyekhez az ajánlásaim az alábbiakban foglaltam össze:

- Kiadás: A legnagyobb hiba minden vállalat esetében, a következő, Verzuh Projektmenedzsment könyvében található idézetben is megtalálható elemzés hiánya, „Az összes munkavégzéssel kapcsolatos becslés részletes forrása az egyes feladatok becslése. A sorrendi kötöttségek és az erőforrásgok kiegyensúlyozása után ez reális képet ad arról, hány emberre van szükség. Az erőforrások becslése az összes munkavégzésre vonatkozik.” [47] Ahogy a két céges elemzésben is jeleztem, (vö. 3.2.11. és 3.3.5) a vállalatok az esetek jelentős többségében nem számolnak olyan kiadásokkal (pl. minden résztvevő munkatárs költsége, licence költségek, amelyeket már korábban beszerzése kerültek stb.), amelyek nem közvetlen a projekt során merülnek fel, mivel úgy gondolják azon költségek állandók, így a használatuk nem jár többletkiadással a vállalat szempontjából. Ez a megközelítés azonban véleményem szerint a lehető legrosszabb, hiszen munkáltatóként egy versenypiacon az a feladatunk, hogy az adott erőforrásokból a lehető legnagyobb bevételt termelő rendszert, illetve szolgáltatásokat állítsuk elő. Tehát, ha ezen költségek nem kerülnek elszámolásra, úgy tűnhet, mintha végtelen mennyiségű kolléga dolgozna a vállalatnál és nem lenne szükség a priorizálásra az egyes feladatok közt. Ebben

[46] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: User Story
[47] Verzuh Eric, 2006, 199.

az esetben egyértelmű javaslatom az, hogy minden, a projekten dolgozó kolléga bekerülési értékét vegyék figyelembe, akár napidíjban meghatározva, mivel így elég hamar kiderülne, hogy valóban rentábilis-e az adott innováció, hiszen így a kiadás oldalon jelentős összegek jelennének meg, amelyek tényleg felmerülnek a végrehajtás során.

- Bevétel: Bevétel oldalon három problémás részletet találtam, amelyekre az alábbi pontokban teszek ajánlásokat:
    - Piaci vizsgálat: A vizsgálat eredményeképp látható, hogy a biztosítói elemzés során (vö. 3.2.11.) készül STEEP elemzés, azonban a javítónál erre talán még nagyobb szükség lenne, mivel ez alapján akár magasabb bevételt is elérhetne a vállalat a flottában található gépjárművel szervizelésekor: ha például lenne általános képe lenne a döntéshozónak arról, hogy a piac egyéb szereplői milyen óradíjon vagy kedvezménnyel dolgoznak. Ezért ezek elemzése véleményem szerint minden esetben szükséges a projekt elindítása előtte.
    - Információs többletérték megértése és használata: A javítói projekt esetében a valódi hozzáadott értéket a piaci adatok utáni kutatás hozta volna létre, mivel azzal sokkal jobb tárgyalási pozíciót alakíthatott volna ki a flottakezelő ellenében, aminek eredményeképp, ha ilyennel a kezében csak 1.000,- Ft-tal magasabb óradíjat tudott volna kialkudni, az éves szinten az átlagos 2.2 órás munkaidővel és 1500 eseménnyel számolva 3.300.000,- Ft-tal magasabb bevételt érhetett volna el évente. A biztosítói projektnél mindenképp szükséges lenne az információs többletérték meghatározása, amelyet a 2.8.-as pontban részletezek. A világ legnagyobb multinacionális vállalatai élnek az információs többletértékből, és itt a biztosító hasonlóan működik, elég csak belegondolnunk abba, hogy milyen alternatív kár kifizetés csökkentési lehetőségekkel nem számolt a megtérülésnél, hiszen egy káresemény során azt tudni, mi történt korábban a gépjárművel, rengeteg érvet ad a biztosító nevében eljáró szakértő kezébe, hogy azzal akár pl. pszichológiailag is befolyásolja a károsultat. Az ilyen típusú információk értékének meghatározása nagyon körültekintő munkát

igényel, mivel több élethelyzettel is számolni kell, azonban elengedhetetlen az egyértelmű projekt érték meghatározáshoz.

- Projekt érték meghatározása: Ahogy a 2.7-es pontban mélyrehatóan jellemeztem, manapság már nagyon jó és részletes projekt értékelési rendszerek állnak rendelkezésre, amelyek használatát egyik elemzett vállalatnál sem lehetett megtalálni. Ezek ismerete és használata azért is fontos lenne, mert a későbbiekben havi vagy negyedéves szinten lehet ellenőrizni a projekt megtérülését, pontos képet adni a projekt szponzornak arról, milyen hatással van a várható eredményekre, ha csúszik a projekt, vagy ha bármilyen változás áll be.

4.3 Döntés

Az elemzés (vö. 3.2.11. és 3.3.5 fejezet) során is külön rámutattam, főleg a biztosítói folyamat esetében, hogy a megfelelő döntésekhez szükséges különböző alternatívák felállítása azok részletes előnyeinek és hátrányainak megfogalmazásával. Sajnos ez még a biztosító esetében is jelentős problémát jelent, mivel a döntéshozatalkor a vezetők egy irányba indíthatják csak meg a változási folyamatot, ahelyett, hogy teljes képük lenne arról, egy problémát milyen módokon lehet megoldani és esetleg azok milyen későbbi megoldási lehetőségeket takarnak. Több alternatíva felvázolása nélkül lehet-e minden megoldási javaslat másként egyformán ideális kérdést fel sem lehet tenni, ami pedig a téves döntések automatizált feltárásának egyik alapvetése.

4.4 Megrendelés

A megrendeléssel kapcsolatban csak egy megjegyzést tettem, azonban az az egész folyamatra és a később átadandó szolgáltatás minőségére hatással van, ez pedig a részletes technikai és üzleti dokumentáció, amely a biztosító esetében megtörténik, azonban a javítónál (vö. 3.3.5) hiányos. Abban az esetben, ha a vállalaton belül nincs olyan kolléga, aki legalább minimális szinten értene az informatikához, valamint a belső rendszerek működéséhez, az üzleti folyamat szinte biztosan csorbát fog

szenvedni, mivel a beszállítóknak az üzleti igények alapján dolgoznak, azonban nem feltétlen ismerik a megrendelő üzleti érdekeit és piaci helyzetét. Ahogy jelen esetben is, az egyik, ha nem a legfontosabb feladat minden informatikai fejlesztés során, hogy az üzleti igények technológiai lépésekre való lefordítása egy olyan kolléga által történjenek, aki mind a két területre megfelelő rálátással bír és meg van hozzá a szaktudása. Véleményem szerint, bár jelentős a hiány az informatikus kollégákból, az ilyen tudással rendelkező szakemberekből még nagyobb.

4.5 Fejlesztés

A szolgáltatás szoftverfejlesztéséhez tartozó ajánlásokat a kritikához kialakított táblázat (Táblázat 5.) folyamata alapján készítettem el:

- Fejlesztési módszertan és prioritás: A fejlesztési módszertan határozza meg azt, hogy egy vállalat belső fejlesztési osztálya milyen gyorsan képes alkalmazkodni a felmerülő üzleti igényekhez. Az elemzés során (vö. 3.2.11. és 3.3.5) mind a biztosító, mind a javító esetében jeleztem, hogy az általuk használt formula nem alkalmas az azonnali és minőségi reakcióra, ennek elsődleges oka pedig a használt módszertan hiányossága, így az első és legfontosabb ajánlás, hogy térjenek át Agile/Scrum alapú szoftverfejlesztésre és ezt várják el a beszállítóktól is. Ennek előnyeit a 2.1.2-es fejezetben részletes is kifejtettem.
- Dokumentáció: a dokumentáció mind a két vállalat esetében hiányos (pl. = Architektúrális dokumentumok, szolgáltatás dizájn dokumentum, felhasználói felülettervezéshez szükséges dokumentumok stb.), azonban fontos megkülönböztetni, hogy egy javítónál a beszállító nem feltétlen felel a megfelelő technikai dokumentációért, csak az átadott szolgáltatás megfelelő működéséért. Azonban a biztosító és így jellemzően a nagyvállalatok esetében, ahol belső fejlesztés zajlik, a dokumentáció az egyik alappillére a megfelelő és minőségi, azonban gyors szolgáltatás kialakításának, elkészítésének, az alábbiak miatt:
    - Jelentős belső erőforrásokat emészt fel az, ha egy kollégának akár napokat kell csak azzal foglalkoznia, hogy egy másik munkatárs által

létrehozott szolgáltatást megismerjen és folytatni tudja annak fejlesztését, mert az nem volt dokumentálva (kellőképpen), vagy mert egy több ezer soros kód csak a véletlen műveként működik megfelelően.

- A lehetőség egy globális vállalat esetében, hogy akár egy kolléga szaktudásán és ismeretein múljanak jelentős bevételt nyújtó szolgáltatások, manapság már megengedhetetlen, ehhez pedig mindenképpen szükséges a jó minőségű dokumentáció és az, hogy egy csapaton belül ismerjék a munkatársak az egyes projektek részleteit és azt, hogy pontosan mivel foglalkoznak egy szolgáltatás fejlesztése során.

Ezek miatt az alábbi ajánlásokat fogalmaztam meg:

- Érdemes bevezetni egy egységes rendszert, ami mind a folyamatot, mind a hozzátartozó dokumentációt egységesen irányítja. A folyamatok kikerülésére nem ad lehetőséget, a dokumentációról pedig egyértelmű utasításokat ad. A 2.4-es pontban a JIRA és a Confluence kettőséről adtam rövid összefoglalást, azonban a piacon több hasonló szolgáltató is található (pl. Proofhub, Assembla, Crocagile, stb.).
- Léteznek a piacon egységes útmutatások (pl. GNU rendszer https://www.gnu.org/prep/standards/html_node/Comments.html) az ún. átlátható és a kommentelt kód kialakítására, amit a vállalatok érdeke bevezetni.
- Az architekt kollégák feladata szolgáltatásokon belüli és a szolgáltatások közti adatáramlás diagrammok (pl.: https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm) kialakítása, amik egy képen jelzik a teljes működést.

- Technológia

Az informatika jelenlegi fejlődése szinte követhetetlen, sokszor naponta jelennek meg forradalmi technológiai újítások, azonban az alábbi négy trend követése minden vállalat számára ajánlott, ezek bevezetésének előnyeit az alábbiakban foglaltam össze, a hozzájuk tartozó részletes leírást pedig a 2.3-as fejezet tartalmazza:

- Felhő: „A programozók saját idejükből és a gépidőből sokat vesztegettek el ilyen szimulátorok megírására, ill. azok felhasználására. A szimulátorok alkalmazására általában egyszerű ok készteti őket: egy számítógépközpont új gépeket vásárol, de továbbra is használni szeretné a régi gépre írt programokat (inkább, mint hogy újra kelljen írni őket). Általában azonban ez többe kerül és kevésbé eredményes, mint ha egy erre a célra szervezett munkacsoport egy ideig kifejezetten az újra programozás feladatával foglalkozik.” [48] A felhő alapú szolgáltatás építésnek két nagy előnye van és az előbbi idézet szerint csak kifejezetten erre fókuszálva érdemes átköltözni rá. Az egyik az, hogy jelentős költségcsökkentés érhető el vele beszállítói oldalon, mivel, ha klasszikus szervert vásárolunk a maximális terhelésre készítjük azt fel, tehát az egyszeri kiadásunk azt fedezi, míg felhőalapú szolgáltatásokat automatikusan skálázódnak a használat függvényében, így csak akkor fizetünk magasabb díjat, ha azt valóban kihasználták az ügyfeleink, tehát együtt mozog a bevételünk és a kiadásunk. Míg a másik, szintén alacsonyabb kiadást eredményező tétel, hogy a felhőalapú rendszerek komplex szolgáltatási csomagot tartalmaznak, így a vállalatnak nem szükséges belsős állandó munkaerőt fenntartani az üzemeltetésre, a rendszerbiztonságra és egyéb informatikai területekre.
- Kódbázis és Microservices: Itt is szintén két előny azonosítható be, amely miatt quasi minden vállalat számára erősen ajánlott a bevezetése. Elsőként a kollaborációs lehetőség említeném, hiszen a kódbázis egy helyen tárolja az összes szolgáltatás adatát, míg a Microservices alapú fejlesztés megengedi, hogy a csapatok egyes tagjai egyedi, azonban ugyanahhoz a szolgáltatáshoz tartozó funkciókon dolgozzanak, ami a fejlesztés sebességét jelentősen felgyorsítja. Az előzőből fakad a második előny, ami szintén a kollaboratív munka eredménye, ami pedig a kód és ezáltal a szolgáltatás minőségének folyamatos javulása, mivel

[48] Knuth Ervin Knuth, 1981, 158.

az egyes kollégáknak a rendszer csak akkor enged élesíteni, ha azt egy másik munkatárs már tüzetesen átnézte és elfogadta.

  - Big Data: A Big Data alapjaival a legtöbb vállalat még csak most ismerkedik, most tudja meg, milyen gazdasági és piaci előnyöket szerezhet számára, legyen az akár célzott hirdetés vagy profilozás, hiszen ez az, ami valójában megalapozta az ún. hozzáadott érték fogalmát és anyagiakban kifejezhető értékét. Itt azonban szoftverfejlesztési szempontból tekintjük fontos tényezőnek a Big Data rendszert, amivel növelhető egy szolgáltatás rendelkezésre állása és leszűkíthető az külső rendszerektől való függősége. Ennek oka, hogy az informatikai rendszerek minden esetben adatokkal dolgoznak, ezen adatokat pedig valamilyen arányban külső forrásokból szerzik, dolgozzák fel, majd tárolják. Minél több ilyen adatforrással foglalkozik egy szolgáltatás, annál nagyobb az esélye annak, hogy ha akár csak egy forrás is elérhetetlenné válik a teljes rendszer összeomlik. Azonban, ha a vállalat rendelkezi Big Data rendszerrel, ezt egy helyen összefoghatjuk, ami sokkal egyszerűbbé és átláthatóbbá teszi a működtetést.

- Hozzáértés és szakmai tudás: Az informatika a világ egyik leggyorsabban fejlődő területe és szinte már nem is tudunk elképzelni olyan üzletágat, amit ne ez működtetne/támogatna. Ez azonban nagy felelősséget és költséget ró a vállalatokra, mivel a munkaerő folyamatos oktatása elengedhetetlen. Azonban minél jobb egy informatikus szakértelme, annál többet ér a piacon, így az oktatás mellett a juttatásai is ezzel együtt kell, hogy nőjenek. Fontos megjegyezni, hogy sok vállalat azért nem fektet az alkalmozottjaik tudásába, mivel úgy gondolja, hogy így nehezebben fognak tudni váltani. Szerencsére azonban ez az elemzett cégek esetében nem fordult elő, de az ilyen hozzáállással véleményem szerint a saját pozícióit is kockáztatja a mindenkori cégvezető. Mindemellett fontos a szakmai megbecsülés, az önállóság és a kreatív munkavégzésre való motiválás is, mivel az IT a folyamatos erőforráshiány miatt a klasszikus munkaerőpiaccal ellentétesen működik, azaz

sok esetben a munkavállaló diktálhat és szabhatja meg a kereteket, a munkáltató pedig csak örül, ha megfelelő kollégát talál.

- Tesztelés: A 2.1.2-es fejezet 5.-ös pontjában kifejtem, hogy egy megfelelően működő Scrum csapatban hogyan zajlik a tesztelés és nem is teszek különbséget a tesztelés és a fejlesztés között, mivel ezek a folyamatos együttműködés eredményei. A témával kapcsolatban az alábbi ajánlásokat fogalmaztam meg, amelyek a korábbi 2.1.2-es fejezetben is szerepeltek:
    - Unit Test: A tesztelés első és legfontosabb lépése, aminek bevezetése elengedhetetlen, az ún. Unit Test, amit még maga a fejlesztő végez, az adott, éppen fejlesztés alatt lévő funkció megfelelő működésével kapcsolatban.
    - Átadás tesztelésre: Ennek két fontos eleme van, amit mindenképp érdemes beépíteni: az egyik az, hogy a fejlesztő kolléga a közös dokumentációs térben és a kód kommentjeként jelzi, mit szükséges tesztelni, valamint, ha a tesztelőnek való frissített kód automatikusan kerül fel egy olyan környezetre, ahol azt végre lehet hajtani.
    - Regressziós és automata tesztelés: A korábbi funkciók tesztelése (regressziós) a legfontosabb egy vállalati informatikai rendszer működtetésében, mivel ennek segítségével győződhetünk meg arról, hogy nem csak az újonnan fejlesztett részei működnek a szolgáltatásnak, hanem minden korábbi is, ami a vállalat megfelelő/stabil működésének az alapja. Ennek kötelező része az automata tesztelés, ami tulajdonképpen egy robot, ami a felhasználói, valamint a szolgáltatások közötti interakciókat figyeli, majd a korábbiaktól eltérő működést jelzi a tesztelő kollégának.
- Élesítés: Az új kód élesítésével kapcsolatban két ajánlást fogalmaztam meg, amelyeket a 2.1.2-es fejezet 7. pontjában jelzek is és sajnos, mind a két elemzett vállalatnál hiányzik. Ez az élesítés automatizálása, mivel előfordulhat, hogy több szolgáltatás is érint egy verzió váltás így minimalizálható a kiesés, valamint az, hogy ezt az üzemeltetés végezze, aki átlátja az aktuálisan működő rendszert, nem pedig a fejlesztés, aki csak az adott kiegészítő funkciók átadására koncentrál.

- Fejlesztés mérése: A 2.5-ös fejezetben leírt mérőszámok bevezetése, minden vállalat számára ajánlott, természetesen ezek mellett továbbiakat lehet találni a Scrum és Agile leírásokban, azonban, mivel egyik elemzett vállalat sem ezt a módszertan használja, előzetes szükséges annak bevezetése. A mérőszámok magukban foglalják a következőket: az átláthatóságot, a produktivitást, hatékonyságot, a minőséget és a bevételtermelést.

4.6 Indítás

A szolgáltatás átadása a felhasználóknak és a Projekt Menedzser és a Product Owner feladata. Ennek legfontosabb része a végfelhasználók tájékoztatása a bevezetett változásokról, amely mind a két elemzésben résztvevő vállalatnál hiányos volt. Ezen kívül ajánlott még bevezetni sztenderdeket pl.: a hotline és az üzemeltetés felkészítésére az esetlegesen beérkező felhasználói hibák megoldására és kezelésére.

4.7 Visszajelzés

Minden sikeres projektben szerepelnie kell az átadott szolgáltatás későbbi visszajelzésekre alapuló folyamatos fejlesztésének. Ahhoz, hogy teljes képet kapjunk arról, mennyire elégedettek a felhasználók a programunkkal, beépített és előre meghatározott kérdéseket (pl. felhasználói élményekkel kapcsolatos kérdések, szoftver ergonómiához kapcsolódó kérdések) kell alkotnunk már a tervezéskor. Ezzel kapcsolatban tehát csak egyetlen, azonban nagyon fontos ajánlást kívánok megfogalmazni, az pedig nem más, mint, hogy már az ötlet kidolgozásakor szükséges bevonnunk minden lehetséges értintettet.

5 ÖSSZEFOGLALÁS

Dolgozatom megírásának legfőbb motivációja 15, informatikai menedzsment pozíciókban töltött év tapasztalatának összefoglalásaként két, a szoftverfejlesztés során elengedhetetlen fogalom (gazdaságossági számítások és az információs többletérték) jelentőségének bemutatása és használatának ismertetése.

Az első számomra nagyon fontos kifejezés a gazdaságossági számítások kialakítása egy innovációs (vö. szoftverfejlesztési) projekt során, aminek részleteit több helyen is (vö. 3.2.4 fejezet és 3.3.3 fejezet, mint várható költségek) kifejtettem, és aminek hiánya vagy esetleges hiányosságai már a tervezéskor eltéríthetik az innovációs folyamatokat az előre meghatározott céltól, legyen az a költségek egységes meghatározása, vagy a piaci elemzések, várható bevételek (vö. információs többletérték a szoftverek, az informatikai projekt esetében) ismeretével kapcsolatos probléma.

Mindezekből következik a második fogalom, ami az információs többletérték, amely meghatározása minden vállalat és projekt esetében eltérő metódust kíván, de lényege, hogy egy informatikai szolgáltatás, nem határozható meg csak a befektetett munkadíj és anyagköltség, valamint a ráépülő profit eredőjeként, hanem a felhasználó számára hozzáadott értéket is szükséges meghatározni, amely számára az információhoz való hozzájutással létre jön. Dolgozatomban erre is több verziót mutattam be (vö. 3.2.4 fejezet, mint várható költségcsökkentés és 3.3.3 várható bevételnövekedés), amelyek közül talán a biztosító esetében a gépjármű kártörténetét emelném ki, mint követendő példát.

A dolgozat a knuth-i elvárások teljesítésének első lépcsőfoka, ahol olyan esettanulmányokat tudunk kialakítani, melyek tesztesetként szolgálnak a későbbi automatizálások számára.

6 FELHASZNÁLT FORRÁSOK JEGYZÉKE

6.1 Kötetek

- ANDOR György: *Vállalati Pénzügyek II.* Budapest, BME, 2005.
- BARTHÉLEMY Jerome: *The Hidden Costs of IT Outsourcing.* MIR, Solan Management Review, 2001.
- BÉLYÁCZ Iván: *Befektetés-elmélet.* Pécs, PTE, 2001.
- BÉLYÁCZ Iván: *A vállalati pénzügyek alapjai.* Budapest, Aula, 2007.
- BENCSIK Andrea: *Menedzsment és projekttechnikák.* Veszprém, Veszprémi Egyetemi Kiadó, 2005.
- BORSI Balázs: *A technológia és tudásáramlás szerepe a magyar vállalati versenyképesség szolgálatában.* http://www.pm.gov.hu/Dokumentumok/Seo/fuzetek 2004.
- BŐGEL György - FORGÁCS András: *Informatikai beruházás - üzleti megtérülés.* Budapest, Műszaki könyvkiadó, 2003.
- CHIKÁN Attila – DEMETER Krisztina: *Az értékteremtő folyamatok menedzsmentje.* Budapest, AULA Kiadó, 2003.
- CHIKÁN Attila: *Bevezetés a vállalatgazdaságtanba.* Budapest, AULA Kiadó, 2006.
- DEVARAJ Sarv - KOHLI Rajiv: *The IT Payoff - Measuring the Business Value of Information Technology Investements* Amazon, 2002.
- ERDŐS Ferenc: *IT-outsourcing a kis- és közepes vállalkozásoknál.* Budapest, Számokt, 2007.
- FARKAS János: *Az ötlettől a megvalósulásig.* Budapest, Akadémiai Kiadó, 1974.
- GYÖKÉR Irén: *Emberi erőforrás menedzsment.* Budapest, Műszaki Könyvkiadó, 2000.
- Dr. GYULAI László - Dr. HANYECZ Lajos - JÁROS Péter - LÁSZLÓ Csaba: *Bevezetés a kis- és középvállalkozások pénzügyi modellezésébe (kézirat)* Finance Oktatási és Kutatási Alapítvány, 2002.

- ILLÉS Ivánné: *Vállalkozások pénzügyi alapjai.* Saldo Pénzügyi Tanácsadó és Informatikai Zrt. Budapest, 2007.
- KERÉKGYÁRTÓ Györgyné - MUNDRÚCZÓ György: *Statisztikai módszerek a gazdasági elemzésben.* Budapest, Aula Kiadó, 2001.
- KNUTH Ervin Donald: *A számítógép-programozás művészete 1, 2, 3.* Addison-Wesley Publishing Company Inc. 1981.
- RAFFAI Mária: *Az információ - Szerep, hatás, menedzsment.* Győr, Palatia, 2006.
- SCHWABER Ken és SUTHERLAND Jeff: *The Scrum Guide.* 2020 November.
- Dr. SZERB László: *Kisvállalkozások finanszírozása* (Kézirat – 1. fejezet) 2007.
- VERZUH Eric: *Projektmenedzsment.* Budapest, HVG Kiadó, 2006.
- ZELLER Gyula - KOLTAI Zoltán: *Pénzügyi alapismeretek.* Pécs, Metamédia Bt., 2017.
- ZOLTAYNÉ PAPRIKA Zita: *Döntéselmélet.* Budapest, Alinea Kiadó, 2005.

6.2 Internetes források

- LAWRENCE David B.: *The quantification of the value of information in decision making.* Iowa State University 1979. Letöltve: 2021.03.27. *https://core.ac.uk/download/pdf/38918398.pdf*
- FEHÉR Péter: *Informatikai beruházások pénzügyi értékelése.* 2006. Letöltve: 2021.04.02. *https://tudman.files.wordpress.com/2009/01/feher-peter-informatikai-beruhazasok-penzugyi-megterulese-v11.pdf*
- ERDŐS Ferenc: *Az informatikai beruházások értékelése és megtérülése.* Széchenyi István Egyetem Informatikai Tanszék 2008. Letöltve: 2021.04.11. *http://www.sze.hu/~gaul/tszhonlap_public/vallinfo/erteke2.pdf*
- ERDŐS Ferenc: *A kis- és közepes vállalkozások informatikai beruházásai és azok megtérülési lehetőségei Magyarországon - Doktori értekezés. 2009.* Letöltve: 2021.04.15. *https://rgdi.sze.hu/files/Ertekezesek,%20tezisek/Erdos%20Ferenc%20ertekezes_vegleges.pdf*

- SNAPP Shaun: *How to Understand the Basic Rules of Software TCO Calculation.* 2018. Letöltve: 2021.04.19. *https://www.brightworkresearch.com/how-to-understand-the-basics-of-software-tco/*

7 MELLÉKLET

7.1 Szekunder adatok katalógusa, adatvagyon kialakítása és feldolgozása

A szekunder adatok körét IT vezetőként én alakítottam ki és készítettem elő a munkáltatóm engedélyével. Annak érdekében, hogy a jelen dolgozat alapját adó adatvagyon méretéről, komplexitásáról átfogó képet kaphasson az Olvasó, annak folyamatát az alábbiakban foglalom össze:

1. A munkahelyem a haza biztosítói és flottakezelői gépjármű költség kalkulációk vezető szolgáltatója, mind káresemények, mind szervízesemények tekintetében.
2. Az online programot igénybe vevő 3400 felhasználó kitölti a gépjármű tulajdonosának és a biztosítás adminisztratív adatait, majd az megadja az alvázszámot, amelyre a program egy gyári adatbázisból meghatározza a márkát, modellt és a felszereltséget, (ahogy az 1.2-es fejezetben jeleztem, ez 54 márka, 3800 modell és több, mint 240.000 féle felszereltségi változat).
3. Ezután egy ún. „robbantott" ábrán kiválasztja a sérült alkatrészeket (a rendszer jelenleg több, mint 9 millió alkatrészt tartalmaz, cikkszámmal, nettó kiskereskedelmi árral, gyári ajánlással a munka normaidőkre), amihez megadja a szükséges pozíciót, mint a csere, fényezés, javítás, majd a Számítás gombra kattint, ami létrehozza a teljes értékű kalkuláció a megfelelő adattartalommal és PDF-ben a felhasználó rendelkezésére bocsájtja.
4. A háttérben ennek eredményeképp elkészül egy XML[49]-fájl (XML = Extensible Markup Language – Kiterjeszthető Jelölőnyelv), ami strukturáltan tartalmazza az összes megadott és generált értéket, amelyet egy API[50]-n keresztül letöltve jellemzően a partner belső szolgáltatásaival való adatáramlásra használnak fel, egyszerűsítve saját munkájuk.

[49] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: XML
[50] 8. Fejezet: Rövidítések És „Idegen" Szakkifejezések Jegyzéke: Webservice/API

5. Mivel ezen adatokhoz hozzáfér a szolgálató is, lehetőségünk van letölteni az XML-fájlt és azt feldolgozni. Példaként alább található egy rövid részlet egy XML-fájlból az Ábra 14.-ben, amely a gépjármű alapadatait tartalmazza:

```
<Vehicle>
 <VehicleIdentification>
  <ItemId>1A3967A3-3BA6-E393-C586-3D6664DF95A9</ItemId>
  <VIN>NMTKZ3BXX0R214547</VIN>
  <ManufacturerCodeAX>70</ManufacturerCodeAX>
  <ManufacturerName>TOYOTA * [70]</ManufacturerName>
  <ModelCodeAX>CH</ModelCodeAX>
  <ModelName>C-HR   2016.10. -től                (ZYX/NGX/ZGX) [CH]</ModelName>
  <SubModelCodeAX>01</SubModelCodeAX>
  <SubModelName>C-HR [01]</SubModelName>
  <SelectedAXMOCodes>B2F4F5G9H1H3H7I8J2J3J6J7K8L4L7M7N3P7Q1Q7Q9R3S5V3V4V5W1Y4U8</SelectedAXMOCodes>
  <VINApplied>Yes</VINApplied>
  <VINQueryTime>2020-12-09T08:33:42.268</VINQueryTime>
  <SellingTypeCode>ZYX10L-AHXEBW</SellingTypeCode>
  <EngineIdentification>2ZRFXE - 1800CC 16-VALVE DOHC EFI</EngineIdentification>
  <EngineVariant>1798 CM3 90 KW HIBR.</EngineVariant>
  <VehicleInterior>BOR ULESEK</VehicleInterior>
  <PaintVariant>2-RÉTEGŰ-METÁL - 1K0 - METAL STREAM</PaintVariant>
  <VINQueryExecuted>true</VINQueryExecuted>
  <VINQueryGaveResult>true</VINQueryGaveResult>
  <ModelYear>2018</ModelYear>
```

Ábra 14.: Gépjármű kalkuláció eredményének XML tartalma (Forrás: Saját munka)

6. Az XML-fájlokat egy ütemezetten futó parancs tölti le öt percenként, majd minden éjszaka az újabb kalkulációkat az XML-ben található bekezdések (ún. tag) alapján hozzáfűz egy mysql adatbázishoz. Az egész program egy belső, csak VPN[51]-en elérhető szerveren fut (VPN = Virtual Private Network – Virtuális Belső Hálózat), így illetéktelenek nem férnek hozzá, az XML-ből pedig minden személyes adatot (rendszám, név, cím, biztosítási adatok stb.) eldobunk az SQL[52]-táblákba való beolvasás előtt (SQL = Structured Query Language – Struktúrált Lekérdező Nyelv).
7. A mysql adatbázisból hetente készítünk egy dump-fájlt, amit minden kolléga a saját gépén egy ún. Mysql Workbench [53]nevű programban tud szerkeszteni, illetve lekérdezéseket futtatni rajta. Az adatbázis 45 fő táblát és 18 segédtáblát tartalmaz, 49 millió rekorddal, 2015-től 2020 év végéig 1,45 millió káreseményről (2,3 millió kalkulációt, mivel egy káreseményre sok esetben a

[51] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: VPN
[52] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: SQL, MySQL, Workbench
[53] 8. Fejezet: Rövidítések És „Idegen” Szakkifejezések Jegyzéke: SQL, MySQL, Workbench

javító és a biztosító, vagy szakértő is készít kalkulációt), amelyből végeredményben csak a 2019-es adatokat használat fel. Alábbi Ábra 15.-n egy rövid leírás található néhány tábláról:

Táblakezelő	XML Tag	Adat Típus	Adat Leírása
AX_VehicleIdentification			
		INT(11)	
	VIN	VARCHAR(255)	
	ManufacturerCodeAX	CHAR(2)	Gyártókód
	ModelCodeAX	CHAR(2)	Modellkód
	SubModelCodeAX	CHAR(2)	Altípuskód
	SelectedAXMOCodes	VARCHAR(255)	kiviteli változatok kódjai egyetlen hosszú karaktersorozatban
	VehicleSelection	VARCHAR(255)	értéke VIN, hogyha alvázszám alapján történt a beazonosítás
	VINApplied	BOOLEAN	Alvázszám lett szolgáltatva
	VINQueryTime	DATETIME	VIN lekérdezés ideje
	SellingTypeCode	VARCHAR(255)	Kereskedelmi kód
	EngineIdentification	VARCHAR(255)	Motor id
	EngineVariant	VARCHAR(255)	Motor adatok (köbcenti, teljesítmény), VIN lekérdezésből származik
	VehicleInterior	VARCHAR(255)	Gépjármű belső adatai, ugyancsak VIN lekérdezésből
	PaintVariant	VARCHAR(255)	Fényezés típusa, adatai, ugyancsak VIN lekérdezésből
	VINQueryExecuted	BOOLEAN	VIN lekérdezés történt
	VINQueryGaveResult	BOOLEAN	VIN lekérdezés adott vissza eredményt
	ManufactureDateRangeStart	DATE	Mikortól gyártják adott típust, VIN lekérdezésből származó info
	ManufactureDateRangeEnd	DATE	Meddig gyártották az adott típust, VIN lekérdezésből származó info
	ModelYear	VARCHAR(255)	Gyártási év
	VINQueryReturnCode	VARCHAR(255)	Alvázszámlekérdezés válaszüzenet pl. 00
AX_VehicleAdmin			
		INT(11)	
	PlateNumber	VARCHAR(255)	Rendszám
	RegistrationDate	DATE	Hazai forgalomba helyezés
	FirstRegistrationDate	DATE	Első üzembe helyezés
	NextTechCheck	DATE	Műszaki vizsga érvényessége
	PreviousDamage	BOOLEAN	Korábbi károk
	PreviousDamageDesc	VARCHAR(255)	Előzménykár leírása
	PreviousOwnerCount	INT(11)	Hányadik tulajdonos
	ReplacementValueAmount	INT(11)	Forgalmi érték
	VehicleCondition*	INT(11)	Jármű kárkori állapota

Ábra 15.: Adatbázis táblák tartalma – Példa (Forrás: Saját munka)

8. Ebből készül el a végleges tábla, amelyet Excel-be exportáltam, majd átlagokat és egyebeket (Pl. eloszlás, darabszám, medián stb.) számoltam belőle. Alább az Ábra 16.-on egy példa az SQL lekérdezésre:

```
USE xxxx;
DROP PROCEDURE IF EXISTS create_damage_stats_for_process;

DELIMITER $$
CREATE PROCEDURE create_damage_stats_for_process(IN process_name VARCHAR(255))
BEGIN

  SET default_storage_engine=MYISAM;

  SET global sql_mode='';

  DROP TABLE IF EXISTS process_damage_stats;

    CREATE TABLE
    process_damage_stats
  SELECT * FROM damage_stats_with_previous_month WHERE `process` = process_name;

    DROP TABLE IF EXISTS damage_stats_without_chosen_process;

    CREATE TABLE
    damage_stats_without_chosen_process
  AS
    (SELECT
      manufacturer_code,
      manufacturer_name,
      model_code,
      model_name,
      calculation_year,
      calculation_month,
      claim_type_id,
      claim_type,
      vehicle_age_in_range,
      owner_type,
```

Ábra 16.: Adatbázis lekérdezés – Példa (Forrás: Saját munka)

9. Az Excelbe exportált adatokon már csak átlagokat és alap kimutatásokat készítek. Az elkészült 2019-es biztosítói táblában 328.192 káresemény a javítói elemzéshez használt táblában pedig elmúlt 4 év 84.000 szervizeseménye volt megtalálható.

7.2 A gazdaságossági elemzésekben található táblázatok tartalmának háttere, amelyben a szekunder adatok felhasználásra kerültek

Az egyes elemzések során létrehozott táblázatokban található adatok értelmezéséhez szükséges adatokat az alábbiakban részletezem

- Táblázat 7. (3.2.4.1 fejezet):

- Forma	Definíció	Forrás
Ajánlatok száma	A biztosító által kiadott CASCO biztosítási ajánlat online vagy offline rendszeren keresztül	Biztosítós kolléga visszajelzése

- Forma	Definíció	Forrás
Felhasználható darabszám	A biztosító által maximálisan megvásárolható szolgáltatás mennyisége a vezetők által biztosított büdzsé alapján, vállalói oldalon	Biztosítós kolléga visszajelzése
Találati arány	Belső biztosítói elemzés eredménye az elmúlt 5 év felderített csalási eseményei alapján, amelyekhez az adatokat a vállalatom segítségével alakította ki	Biztosító által átadott korábbi károkban érintett gépjárművek alvázszámai alapján a vállalatom által készített kimutatás
Találatok várható száma	A megadott csalási *Találati arány* és a *Felhasználható darabszám* eredménye	Származtatott adat
Átlagos megtakarítás	Belső biztosítói elemzés eredménye az elmúlt 5 év felderített csalási eseményeinek átlagkár összege, amelyekhez az adatokat a vállalatom segítségével alakította ki és amelyek végül nem került kifizetésre	Biztosító által átadott korábbi károkban érintett gépjárművek alvázszámai alapján a vállalatom által készített kimutatás
Átlagos bevételkiesés	A 2019-es átlagos CASCO díj a biztosítónál, amely az elutasított szerződés miatt nem kerül a biztosítóhoz, mint bevétel	Biztosító által átadott korábbi károkban érintett gépjárművek alvázszámai alapján a vállalatom által készített kimutatás
Ezek különbözete	*Átlagos megtakarítás* és *Átlagos bevételkiesés* különbözete	Származtatott adat
Várható megtakarítás	*Találatok várható számának* és az *Átlagos megtakarítás* és *Átlagos bevételkiesés* különbözetének szorzata	Származtatott adat

- Táblázat 8. (3.2.4.2 fejezet):

Forma	Definíció	Forrás
Éves CASCO károk száma	Azon káreseményekre készült kalkulációk száma, amelyek esetében a biztosító CASCO-t jelölt meg biztosítási módozatnak	Saját vállalati adatbázis, az 1.-es melléklet 1.-es pontja alapján
Lekérdezhető darabszám	A biztosító által maximálisan megvásárolható szolgáltatás mennyisége a vezetők által biztosított büdzsé alapján, kár oldalon	Biztosítós kolléga visszajelzése
Releváns használati arány (Amelyben használható eredményt kap a szakértő)	Belső biztosítói elemzés eredménye, amelyekhez az adatokat a vállalatom segítségével alakította ki, korábbi	Biztosító által átadott korábbi károkban érintett gépjárművek

Forma	Definíció	Forrás
	káresemények alapján, ahol utólag derült ki, hogy valamekkora összeget levonhatott volna a kárkifizetés során a biztosító	alvázszámai alapján a vállalatom által készített kimutatás
Használható darabszám	A *Lekérdezhető darabszám* és *Releváns használati arány* hányadosa	Származtatott adat
Átlagos CASCO kárkifizetés	Átlagos CASCO kárkifizetés összege 2019-ben az adott biztosítónál	Saját vállalati adatbázis, az 1.-es melléklet 1.-es pontja alapján
Átlagosan elérhető kifizetéscsökkenés a szolgáltatás eredménye alapján	Belső biztosítói elemzés eredménye, amelyekhez az adatokat a vállalatom segítségével alakította ki, korábbi káresemények alapján, ahol utólag derült ki, hogy átlagos mekkora összeget vonhatott volna le a kárkifizetés során a biztosító	Biztosító által átadott korábbi károkban érintett gépjárművek alvázszámai alapján a vállalatom által készített kimutatás
Káreseményenként elérhető átlagos kifizetés csökkenés	Az *Átlagos CASCO kárkifizetés* és az *Átlagosan elérhető kifizetéscsökkenés a szolgáltatás eredménye alapján* értékek hányadosa	Származtatott adat
Éves szinten realizálható kifizetés csökkenés	A *Használható darabszám* és *A káreseményenként elérhető átlagos kifizetés csökkenés* szorzata	Származtatott adat

- Táblázat 11. (3.3.3.1 fejezet):

Téma	Definíció	Forrás
A javító által a piacon alkalmazott átlagos óradíj szervízesemények során	A vállalatom saját adatbázisában található, a javító által készített szervizeseményekhez tartozó kalkuláció átlagos óradíja	Saját vállalati adatbázis, az 1.-es melléklet 1.-es pontja alapján
A flottakezelő által elvárt óradíj kedvezmény	A flottakezelő által igényelt átlagos óradíj kedvezmény a piaci árakhoz képest, amelyet a hozott volumenért cserébe vár el	A flottakezelő közlése, amely jelezte, hogy a készülő kalkulációkhoz ezt az egységes beállítás igényli
A vállalt óradíj a flottakezelő kedvezményével	*A javító által a piacon alkalmazott átlagos óradíj szervizesemények során,* valamint *A flottakezelő által elvár óradíj kedvezmény* eredménye	Származtatott adat
Egy átlagos szervízesemény munkaideje	A kalkulációk eredményeiből készített adatbázis eredményeiből kinyert átlagos	Saját vállalati adatbázis, az 1.-es

Téma	Definíció	Forrás
	munkaidő, amelyet egy szervizesemény során kiszámláz a javító	melléklet 1.-es pontja alapján
Várható munkadíj bevétel egy szervizesemény esetén	*A vállalt óradíj a flottakezelő kedvezményével* és *Egy átlagos szervizesemény munkaideje* szorzata	Származtatott adat
A munkadíj és a munkavállalónak kifizetett díj átlagos különbség óradíjban (Az összeg, ami a munkavállalónak kifizetett munkabér és a bevétel különbsége a flottakezelői órabérrel)	*A vállalat óradíj a flottakezelő kedvezményével*, mínusz szerelőnek óránként kifizetett, tehát kiadás oldalon megjelenő összeg, tulajdonképpen a szerelő munkadíján való nyeresége a javítónak	A javító által megadott adat
A javító egy szervizeseményre jutó nettó nyeresége a flottakezelői óradíjjal	Az *Egy átlagos szervizesemény munkaideje* és a *munkadíj és a munkavállalónak kifizetett díj átlagos különbség óradíjban* szorzata	Származtatott adat
A javító által szervizelt gépjárművek átlagos alkatrész költsége kiskereskedelmi áron	A vállalatom saját adatbázisában található, a javító által készített szervizeseményekhez tartozó kalkuláció átlagos alkatrész tartalmaz	Saját vállalati adatbázis, az 1.-es melléklet 1.-es pontja alapján
A flottakezelő által elvárt alkatrész kedvezmény	A flottakezelő által igényelt átlagos alkatrészkedvezmény a piaci árakhoz képest, amelyet a hozott volumenért cserébe vár el	A flottakezelő közlése, amely jelezte, hogy a készülő kalkulációkhoz ezt az egységes beállítás igényli
A flottakezelő irányába várható átlagos alkatrészköltség	*A javító által szervizelt gépjárművek átlagos alkatrész költsége kiskereskedelmi áron* és *A flottakezelő által elvárt alkatrész kedvezmény* hányadosa	Származtatott adat
A javító átlagos nyereséghányada az alkatrészeken (Becslés)	A gyári alkatrész importőrtől való megvásárlása és a végfelhasználónak való eladása (jellemzően beszerelés) során keletkezett nyereség a javító oldalán	Gépjármű importőri becslés, szigorúan bizalmas
Az ebből fennmaradt átlagos nyereség az alkatrészeken a flottakezelőnek adott kedvezmény után	A gyári alkatrész importőrtől való megvásárlása és a végfelhasználónak való eladása (jellemzően beszerelés) során keletkezett nyereség a javító oldalán a flottakezelő kedvezményével számolva (nem egyenes következménye	Javítói becslés

Téma	Definíció	Forrás
	a korábban megadott két adatnak, mivel alkatrésztípustól függ a nyereséghányad)	
Átlagos alkatrésznyereség a flottakezelői szervizeseményeken	*A flottakezelő irányába várható átlagos alkatrészköltség* és *Az ebből fennmaradt átlagos nyereség az alkatrészeken a flottakezelőnek adott kedvezmény után* szorzata	Származtatott adat
Egy flottakezelői eseményen várható teljes nyereség (munkadíj és alkatrész)	*A javító egy szervizeseményre jutó nettó nyeresége a flottakezelői óradíjjal* és az *Átlagos alkatrésznyereség a flottakezelői szervizeseményeken* összege	Származtatott adat
1500 szervízre tervezhető teljes javítói nyereség	Az éves szinten 1500 eseménnyel felszorzott *Egy flottakezelői eseményen várható teljes nyereség*	Származtatott adat

- Táblázat 12. (3.3.3.2 fejezet):

Téma	Definíció	Forrás
A külső szoftverfejlesztő partnertől kapott ajánlat a flottakezelővel való integrációra	A javítónak adott ajánlat a DMS-t fejlesztő szoftvercégtől	A javító által megadott adat
Egy adminisztrátor munkatárs átlagos órabére	Az adminisztrációt végző munkatárs átlagos költsége, amely a szervizesemények során felmerül	Javítói becslés
Egy szervízeseményre jutó átlagos munkaidő (Tartalmazza az ügyfélfogadást, kezelést, számlázást és minden egyéb időt)	A flottakezelői és a belső rendszerekben való adminisztráció átlagos ideje a flottakezelő becslése szerint	Flottakezelői becslés
Átlagos adminisztrációs költség szervíz eseményenként	Az *Egy adminisztrátor munkatárs átlagos órabére* és az *Egy szervízeseményre jutó átlagos munkaidő* szorzata	Származtatott adat
Az 1500 várható eseményre jutó teljes adminisztráció költség	*Az Átlagos adminisztrációs költség szervíz eseményenként* szorzata 1500-zal, azaz a várható események számával	Származtatott adat
Egyszeri plusz kiadás a raktárkészlet felkészítésére, hogy azonnal kiszolgálhassak az érkező javítandó gépjárműveket mind a három telephelyen	Annak a költsége, hogy a javító felkészüljön a beérkező gépjárművek szervizelésére, mivel azt azonnal ki kell tudnia	Javítói becslés

Téma	Definíció	Forrás
	szolgálni, így raktárkészletet kell növelnie, azonban nem biztos, hogy azt záros határidőn belül el is tudja adni, vagy fel tudja használni, így teljes egészében költségként számol vele	
Az első évben várható teljes költség	*A külső szoftverfejlesztő partnertől kapott ajánlat a flottakezelővel való integrációra*, és *Az 1500 várható eseményre jutó teljes adminisztráció költség*, valamint az *Egyszeri plusz kiadás a raktárkészlet felkészítésére, hogy azonnal kiszolgálhassak az érkező javítandó gépjárműveket mind a három telephelyen* összege	Származtatott adat

8 RÖVIDÍTÉSEK ÉS „IDEGEN” SZAKKIFEJEZÉSEK JEGYZÉKE

Mivel a szoftverfejlesztés szaknyelve az angol és folyamatos technológiai újdonságok megjelenése, ezért bizonyos kifejezések csak idegen nyelven érhetők el:

Access Rights: Hozzáférési metódusok és jogosultságok összessége, amelyek meghatározzák, hogy az egyes felhasználók a fejlesztés során létrehozott szolgáltatásokhoz milyen formában, mely kollégák engedélyével és milyen funkciók elérésével rendelkezhetnek annak használata során.
Link: https://wiki.en.it-processmaps.com/index.php/Access_Management

Agile: Avagy agilis fejlesztés, egy 2001-ben bevezetett szoftverfejlesztési metodológia, melyet James Martin és James Kerr alapoztak meg a DuPont vállalatnál. Elsődleges célja, hogy segítse az alkalmazkodó tervezést, az evolúciós fejlesztést, korai szállítást, folytonos tovább fejlesztést (lásd CI/CD) és bátorít a változásokra adható gyors és rugalmas válaszokra. Alapjai az ismétlődő, növekményes és evolúciós fejlesztési ciklus (lásd sprint), a szemtől-szembe kommunikáció, a gyors visszajelzés és alkalmazkodási ciklus kialakítása, valamint a legjobb minőségre való törekvés.
Link: http://moodle.autolab.uni-pannon.hu/Mecha_tananyag/szoftverfejlesztesi_folyamatok_magyar/ch04.html

Audit: Auditnak, azaz ellenőrzésnek hívjuk azokat a belső, vagy külső vállalat által végzett teljeskörű átvilágításokat. Célja, hogy felszínre hozza azokat hibákat, hiányosságokat, amelyek problémákat okozhatnak a vállalat működésében.
Link: http://www.szervez.uni-miskolc.hu/blaci/minmen/audit_fogalma.html

AWS: Amazon Web Services rövidítése, mely az Amazon vállalat által szolgáltatott felhőalapú rendszer általános megnevezése. Előnye, hogy nem igényel külön üzemeltetési részleget a vállalatnál, mivel az egyes technológiai feltételeket a fejlesztők maguknak képesek megteremteni. Hátránya azonban, hogy bizonyos vállalatok a jogi háttér miatt csak részfunkcióikat használhatják, a GDPR és egyéb szabályozások miatt.
Link: https://aws.amazon.com/about-aws/

Bug: A program használata során, a felhasználó által a megfelelő működés közben észlelt hibákat bug-oknak hívják, ezek az úgynevezett programhibák. A bug kifejezést gyakran tévesen Grace Hoppernek tulajdonítják, aki egy korai, elektromechanikus számítógép üzemzavaráról írt. A történet egyik elterjedt változata szerint a Harvard egyetem meghibásodott Mark II számítógépében egy molylepke okozott mechanikai hibát, azt azonban már Charles Babbage használta az 1800-as évek során
Link: https://www.computerhope.com/jargon/b/bug.htm

Business Analyst: Az üzleti elemzők általános megnevezése, akiknek feladata, a megrendelő részére készítendő szoftverrel szemben támasztott elvárások precíz, szisztematikus dokumentálása. Munkájával segít meghatározni, hogy egy rendszer változtatás gazdasági szempontból milyen anyag és egyéb pozitív hatással járhat a vállalat szempontjából.
Link: https://www.guru99.com/introduction-business-analysis.html

Change Process: Korunk nagyvállalatai sokszor nehezen és lassan tudnak reagálni a piac és egyéb körülmények változására, ezért jellemzően egységes változtatási folyamat rendszert vezetnek be, amelyben meghatározzák az egyes szereplők feladatai, határidejét, így képessé válnak arra, hogy a folyamataikat a lehető leghamarabb a megfelelően átalakítsák a lehet leggyorsabban.
Link: https://www.prosci.com/resources/articles/what-is-change-management-and-how-does-it-work

CI/CD: A CI/CD a Continous Integration (folyamatos integráció/összefésülés) és a Continues Delivery (folyamatos szállítás) egységes rövidítése, egy fejlesztési gyakorlat, amelyben a fejlesztők a szoftver forráskódját naponta többször, egy közös platformon összefésülik, egyeztetik. Így a kisebb változtatások is gyorsan elérhetővé válnak a csapat többi tagja számára. Az összefésülést követően minden új kódrészlet ellenőrzésre kerül, amely lehetővé teszi a fejlesztők számára, hogy korán felismerjék a problémákat és még az elején javítsák azokat, így időt nyerve és minimalizálva a javítandó kódok mennyiségét.
Link: https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

Commitment: A szó jelentése elköteleződés, amely azt az időpontot jelenti, amikorra a fejlesztői csapat vállalja egy új funkció, vagy hiba javításának leszállítását a tervezés során létrehozott specifikáció alapján, annak megfelelő minőségben.
Link: https://cedw.medium.com/commitments-in-agile-software-development-24a2ae6c8de4

Confluence: A vállalatok által nagy valószínűséggel legsűrűbben használt dokumentációs felület, amely összekötve egyéb folyamatszervező szolgáltatásokkal, egységes, átlátható rendszert képez.
Link: https://www.atlassian.com/software/confluence/features

CRM (Customer Relationship Manager): Egy belső szoftver, melynek feladata a vállalat vevőinek menedzselése. Ezek közé tartozik többek között az ügyfelek és kapcsolattartók adatainak kezelése, marketingkampányok szervezése, értékesítési és üzleti lehetőségek kezelése, ügyfélszolgálati folyamatok ellátása, üzleti folyamatok menedzselése, illetve ezen funkciók összefoglalása, elemzése.
Link: https://vallalatiranyitasi-rendszer.hu/crm-rendszer-jelentese/

Daily Standup: A modern szoftverfejlesztő csapatokban (lásd SCRUM és Squad) dolgozó fejlesztők naponta tartanak egy rövid megbeszélést, amelyen egyeztetik az aktuális állapotot, esetlegesen segítséget kérnek egymástól. A megbeszélés lényege, hogy a fejlesztés a lehető leggyorsabban és a legjobb minőségben működjön.

Link: https://www.scrum.org/resources/what-is-a-daily-scrum

DB: A DB a database, azaz adatbázis angol megfelelője. Az adatbázisok strukturált formában tárolják azon adatokat, amelyekből a szoftverek dolgoznak. Ezek lehetnek akár a munkavállalók adatai, vevői adatok, vásárolt adatok, ad-hoc elérhető adatok, amelyekkel a vállalat nem rendelkezik stb. Az adatbázisokban található adatokhoz többféle módon hozzáférhetnek a programok, közvetlen kiolvasással, ún. webservicen (lásd Webservice), valamint manuális, esetleg automata lekérdezéssel (lásd Query).
Link: http://info.berzsenyi.hu/adatbazisok/az-adatbazis-fogalma

Defect: Defect hívjuk a szoftverfejlesztésben, amikor a program futása nem az előre, a specifikációban meghatározott eredményt hozza. Fontos különbség a bug és a defect között, hogy bug esetében a program működése csorbát szenved, esetleg bizonyos funkciói teljes egészében hiányoznak, míg defect esetén az nem okoz egyértelmű hibát, csak nem a felhasználó által elvárt eredményt produkálja.
Link: https://softwaretestingfundamentals.com/defect/

Deployment: Deploymentnek hívják a fejlesztők, amikor egy programon változtatást hajtanak végre, azaz élesítik a szoftver egyik újabb verzióját. A deployment egy összetett folyamat, amelynek eredményeképp az éles szoftver bizonyos komponenseit megváltoztatjuk a megrendelő által előre meghatározott igények szerint.
Link: https://www.educative.io/edpresso/what-is-software-deployment

Developer: A Developer a szoftverfejlesztő kolléga angol megnevezése. A legtöbb szakirodalom, így hivatkozik rá. Néhány esetben Engineerként jelenik meg.
Link: https://www.ksh.hu/docs/szolgaltatasok/hun/feor08/2/2142.html

Development Architect: Development Architectnek hívják azon fejlesztő munkatársakat, akik a fejlesztési szoftver szakmai döntéseit meghozzák és kidolgozzák. Feladatuk a lehető legáttekinthetőbb, mások által is értelmezhető kód megírása, egységes sztenderdek kidolgozása, a legújabb technológiák bevezetése.
Link: https://www.red-gate.com/simple-talk/opinion/opinion-pieces/the-role-of-the-technical-architect-in-development/

DevOps: A DevOps a Developer (fejlesztő) és az Operation (szolgáltatás) szavak összetétele, amely metódus feladata a szolgáltatás lefejlesztése után a lehető legegyszerűbb, legkevesebb hibával járó folytonos átadása a szolgáltatást később üzemeltetők számára.
Link: https://hu.wikipedia.org/wiki/DevOps

DoD – Definition of Done: A szoftverfejlesztés eddig legfontosabb pontja előre meghatározni, hogy mit tekint a megrendelő kész, megfelelően üzemelő szolgáltatásnak, amely létrehozását elvárja a fejlesztői csapattól.
Link: https://ancaonuta.medium.com/user-story-definition-of-done-dod-in-agile-software-development-and-the-technical-debt-a3abf6821ef2

Environment: Magyarul környezet szónak fordítjuk, amely az egyes felületeket jelzi, amelyen a szoftverfejlesztő kollégák dolgoznak. Célja, hogy a fejlesztés során előforduló hibák még azelőtt kiderüljenek, hogy az ez éles rendszerbe állítják, amelyet a megrendelő használ.
Link: https://medium.com/swlh/environments-in-software-development-cf84adbbf197

Feature: Feature-nek nevezik a szoftvert többletfunkcióval kiegészítő, a megrendelőtől érkező igényt, amely valamilyen előnyt nyújt a felhasználónak.
Link: https://www.productplan.com/glossary/feature-driven-development/

GDPR: General Data Protection Regulation az Európai Unió által kidolgozott általános adatvédelmi irányelv, melyet 2018. májusában lépett életbe és feladata a magánszemélyek adatainak védelme a jogosulatlan felhasználóktól.
Link: https://ec.europa.eu/info/law/law-topic/data-protection_en

GDPR személyes adat (PII): A GDPR személyes adatnak tekint minden olyan adatot vagy adathalmazt, amely alapján egy természetesen személy egyértelműen beazonosítható. Általános és egységes meghatározása nem létezik, mivel például a név magában, Kovács Gyula, nem tekinthető személyes adatnak, hiszen több ezer ember rendelkezik vele, azonban, ha ezt már címmel együtt találjuk meg, már személyes adatnak nyilvánul.
Link: https://gdpr.eu/eu-gdpr-personal-data/
GDPR különösen érzékeny személy adat: Különösen érzékeny adatnak tekinti a GDPR azon adatokat melyek alkalmasak lehetnek hátrányos megkülönböztetésre, mint például a vallás, politikai hovatartozás, szexuális orientáció stb.
Link: https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en

IT Architect: IT Architectnek nevezzük azon munkatársakat, akiknek a feladata a szolgáltatás kidolgozása során a szoftverek és azok kapcsolatainak kialakítása, a rendszerek nagyságrendi technológiai kidolgozása.
Link: https://www.altexsoft.com/blog/engineering/solution-architect-role/

JIRA: A JIRA az Atlassian szolgáltató által szoftverfejlesztési folyamatokat támogató rendszer, melyet vállalatok százai használnak.
Link: https://www.atlassian.com/software/jira/features

KPI: Key Performance Indicator rövidítése, mely egy munkavállaló vagy csapat egységes rendszerbe foglalt teljesítményét értékeli, mutatja az elvárt, a várható, valamint az elért teljesítményt
Link: https://kpi.org/KPI-Basics

MD – Mandays: A szoftver fejlesztés erőforrás igényeit jellemzően ún. Mandays-ban, azaz a fejlesztői munkanapokban határozzák meg.
Link: https://medium.com/welld-tech/stop-rolling-the-dice-how-to-estimate-effort-in-software-development-ee8267898ce9

On Premise: Az On Premise rendszer azt jelenti, hogy egy szolgáltatás egyedi szerveren fut, azaz nem felhő alapú.
Link: https://en.wikipedia.org/wiki/On-premises_software

PMO: Project Management Office-nak nevezik a vállalatok azon csapatokat, amelyek feladata a vezetőség által meghatározott célok végrehajtásának menedzselése a vállalaton belüli és külső partnerekkel egyeztetve. Feladatuk minden esetben egy cél végrehajtásának érdekében zajlik, meghatározott költségvetés mellett.
Link: https://www.pmi.org/learning/library/project-management-office-strategy-execution-1449

Product Owner: A Scrum rendszer (Lásd még Scrum) terminológiájában Product Ownernek nevezik az egyes szolgáltatások vagy szolgáltatás csoportok tulajdonosait, akik a termék teljes életciklusában figyelik, fejlesztik azt.
Link: https://www.scrum.org/resources/what-is-a-product-owner

Regression Test: Mivel a vállalatok a szolgáltatásaikat folyamatosan fejlesztik, ezért egy-egy frissítés esetében, amikor új funkció kerül bevezetésre, az hatással lehet a korábbi működésre, vagy más rendszer működésére. Ennek automatikus tesztelésére szolgált az ún. regression test.
Link: https://en.wikipedia.org/wiki/Regression_testing

Release: A release valójában szabadon engedést jelent. Ez a szoftver fejlesztés esetében azt jelenti, hogy készen áll a legújabb szoftver verzió az élesítésre.
Link: https://en.wikipedia.org/wiki/Software_release_life_cycle

Scrum: A Scrum egy összetett, bonyolult, többszereplős jellemzően folyamatos fejlesztésre kialakított keretrendszer, amelyet többek között a szoftverfejlesztésben is használnak. Feladata a hatékony, egységes fejlesztési terv és szállítási rendszer kialakítása.
Link: https://www.scrum.org/resources/what-is-scrum

Service Delivery Manager: A Services Delivery Manager az elsődleges kontakt a vállalat gazdasági/megrendelői oldala és az IT között. Feladata a folyamatok menedzselése és a gazdasági/jogi/egyéb problémák informatikai szaknyelvre fordítása.
Link: https://www.freelancermap.com/blog/what-does-service-delivery-manager-do/

Sprint: A fejlett szoftverfejlesztési metodológiák a fejlesztési ütemterveket ún. sprintekre osztják. Ezek jellemzően 2-4 hetes időszakot vesznek igénybe, amelyek előre szigorúan eltervezett feladatokat tartalmaznak.
Link: https://www.scrum.org/resources/what-is-a-sprint-in-scrum

SQL, MySQL, SQL Workbench: Az SQL a Structured Query Language azaz a Struktúrált Lekérdező Nyelv rövidítése. Feladata, hogy adatbázisokon, egy egyedi programnyelv használatával különböző utasításokat hajtson végre, legyen az új táblák létrehozása, matematikai műveletek végrehajtása, törlés stb. Ennek egyik piacon

elérhető fajtája a MySQL, a Windows PC-n telepíthető egyénileg használható keretrendszere pedig az SQL Workbench.
Link: http://www.agt.bme.hu/szakm/adatb/db5.htm

System Analyst: A System Analystok feladata a fejlesztői csapatok adatokkal való ellátása, a projektek fejlesztésre való felkészítése.
Link: https://en.wikipedia.org/wiki/Systems_analyst

Test Scenario: A legújabb automata tesztrendszerek képesek arra, hogy megadjuk gépi kódban azt, ahogy a felhasználó viselkedik és azt, hogy milyen választ vár el a szolgáltatástól. Ennek segítségével másodpercek alatt kiderülnek a rejtett hibák az applikációkban, mielőtt azok a felhasználóhoz jutnának.
Link: https://www.softwaretestinghelp.com/what-is-test-scenario/

User Story: User Storynak nevezzük a fejlesztő számára átadott teljes felhasználói élményt tartalmazó dokumentumot, amely a végső elvárásokat tartalmazza.
Link: https://www.atlassian.com/agile/project-management/user-stories

VPN: Virtual Private Network, azaz virtuális belső hálózat, amely megvédi az illetéktelenektől egy vállalat adatait, legyen az üzleti vagy személyes. Emellett segíti a vállalat működését az adatok egyszerűbb elérhetőségével.
Link:
https://hu.wikipedia.org/wiki/Virtu%C3%A1lis_mag%C3%A1nh%C3%A1l%C3%B3zat

Waterfall: Egy korábbi szoftverfejlesztési metodológia, amely alapvetően a FIFO-ra épül. Azaz az elsőnek beérkező feladatokat végzi el elsőnek. Nagy hátránya, hogy nem vizsgálja az egyes fejlesztési kérelmek gazdasági hasznosságát.
Link: https://www.projectmanager.com/waterfall-methodology

Webservice/API: Application Programme Interface-nek nevezzük az egyes szolgáltatások, programok általi kommunikációt, melyek ember számára jellemzően értelmezhetetlenek, azonban a fejlesztés során az egyik legkomolyabb odafigyelés igénylik a megfelelő eredmény elérése érdekében.
Link: https://en.wikipedia.org/wiki/API

XML: Az Extensible Markup Language rövidítése, ami Kiterjeszthető Jelölőnyelv-et jelent. Ez egy olyan speciális fájl formátum, amely lehetőséget ad arra, hogy egy külső program is feldolgozza és megértse a tartalmát, így segítve a szolgáltatások közötti kommunikációt.
Link: http://nyelvek.inf.elte.hu/leirasok/XML/index.php?chapter=1