Magyar Agrár- és Élettudományi Egyetem

Szent István Campus

Műszaki Menedzser Szak

MINŐSÉGBIZTOSÍTÁSI FOLYAMATOK HATÉKONYSÁGÁNAK NÖVELÉSE A SZOFTVERFEJLESZTÉSBEN

Belső konzulens:	Dr. Medina Viktor Ferenc
	Egyetemi docens
Külső konzulens:	Takács Katalin
	QA Lead
Készítette:	Sztaracsek Lilla
	IS7UWZ
	Levelező tagozat
Intézet/Tanszák:	Műszaki Intézet/Műszaki Menedzsment Tanszék

Gödöllő

2023

MAGYAR AGRÁR- ÉS ÉLETTUDOMÁNYI EGYETEM

Szent István Campus, Gödöllő
Cím: 2100 Gödöllő, Páter Károly utca 1.
Tel.: +36-28/522-000
Honlap: https://godollo.uni-mate.hu

MŰSZAKI INTÉZET MŰSZAKI MENEDZSER MESTERSZAK
Projektmenedzsment specializáció

DIPLOMADOLGOZAT

feladatlap

Sztaracsek Lilla (IS7UWZ)

részére

A diplomadolgozat címe:

Minőségbiztosítási folyamatok hatékonyságának növelése a szoftverfejlesztésben

Feladatkiírás:

Elemezze a GSGroup Handyman részlegének agilis szoftverfejlesztési és szoftvertesztelési folyamatait. Hasonlítson össze tesztelési folyamat fejlesztési modelleket, ezek közül válaszon ki egyet, amely a legjobban alkalmazható az adott terület fejlesztésére. A kiválasztott modell felhasználásával azonosítsa a vizsgált folyamat fejlesztendő területeit. Elemezze a Handyman terméket szoftversérülékenységek és szoftvergyengeségek szempontjából és tegyen javaslatot az ebből adódó kockázatok csökkentésére.

Közreműködő tanszék: Műszaki Menedzsment

Külső konzulens: *Takács Katalin QA Lead, GSGroup Innovation Center Zrt.*

Belső konzulens: *Dr. Medina Viktor Ferenc egyetemi docens,* MATE, Műszaki Intézet

Beadási határidő: 2023. 05 hó 02 nap

Gödöllő, 2023. jan. hó 20. nap

Jóváhagyom		**Átvettem**
Dr. Kiss Miklós	Dr.	Sztaracsek Lilla
(tanszékvezető)	(szakfelelős)	(hallgató)

A dolgozat készítőjének külső konzulense nyilatkozom arról, hogy a hallgató az előre egyeztetett konzultációkon megjelent.

Gödöllő, 2023. 04. hó 24. nap

Takács Katalin

(külső konzulens)

1. Tartalomjegyzék

1. Bevezetés

A diplomamunkám téma választása során fontosnak tartottam, hogy egy valós gyakorlati problémával foglalkozzak, ahol az azonosított problémák megoldásával később valós eredményeket érhetek el. A GSGroup szoftveres és hardveres megoldásokkal foglalkozó vállalatnál dolgozom minőségbiztosítási mérnök pozícióban és jelenlegi tapasztalataim alapján úgy látom, hogy a cég Handyman részlegén lévő minőségbiztosítás nem megfelelően szűri ki a termékhibákat, a folyamatok szabályozatlanok és sok esetben a folyamatok megváltoztatásával hatékonyabban lehetne megvalósítani a termékek minőségbiztosítását. A nem megfelelő minőségbiztosítási folyamatra véleményem szerint jó példa, hogy az általunk fejlesztett Handyman alkalmazás termékcsomagra csak az elmúlt évben több mint 200 vevői reklamáció érkezett az ügyfél támogatással foglalkozó részlegre és az ügyfelektől visszaérkező hibák javítására a Handyman részleg alkalmazottai 2022-ben több mint 2200 munkaórát fordítottak. A minőségbiztosítási csapat jelenleg csak a termék minőségének ellenőrzésére fókuszál és nem foglalkozik olyan intézkedések kialakításával és bevezetésével, melyek hozzájárulnának a szoftvertesztelési folyamat hatékonyságának javításához.

A diplomadolgozatom egyik célja a GSGroup Handyman részlegén található agilis szoftvertesztelési és fejlesztési folyamatok elemzése, a folyamatok és az igények felmérését követően egy olyan szoftvertesztelési folyamat fejlesztési modell kiválasztása, mely jól alkalmazható a részlegen belül, végül egy felmérés elkészítése a jelenlegi folyamat érettségi szintről és ez alapján az első körben fejlesztendő folyamatterületek meghatározása. Mivel véleményem szerint a vállalton belül nem fordítanak kellő figyelmet a szoftversérülékenységekből adódó minőségi kockázatokra, ezért a diplomadolgozatom másik célja, hogy egy felmérésen keresztül meghatározzam a termék sérülékenységeknek való kitettségét, majd olyan javaslatokat fogalmazzak meg, melyek alkalmazásával csökkenthetőek az ilyen típusú kockázatok.

2. Témához kapcsolódó hazai és nemzetközi szakirodalom áttekintése

2.1.Szoftveripar sajátosságai

Napjainkra a szoftverek a mindennapi életünk részei lettek, megtalálhatóak a háztatásokban, a kórházakban, a bank szektorban és még sorolhatnám. A szoftveripar és például a gépipar is termékek előállításával, felépítésével foglalkozik. Ugyanakkor egy fizikai termékhez viszonyítva (például egy alkatrészhez képest) sokkal nehezebb megfogalmazni, hogy milyen a jó szoftver, mivel a fizikai termékeknél sokkal erőteljesebb elképzelés van a vásárlók fejében, hiszen a kritériumokat nehéz pontosan meghatározni a szoftverekre vonatkozóan, ebből adódóan pedig az adekvát mérésre alkalmas módszerek kidolgozása sem egyszerű. [1] A nehezen megfogalmazható elvárások következményeként a követelmények a fejlesztési folyamat során sokszor változnak, a fejlesztés megkezdésekor általában nincs jól meghatározott teljeskörű specifikáció. A megrendelők sok esetben nem reális elvárásokat támasztanak a szoftverekkel szemben. Ugyanakkor a minőség-ellenőrzés a legtöbb modern szoftverfejlesztési modellben már a termékkövetelmények meghatározásánál megjelenik, pont úgy, ahogy napjainkban a gépiparban is megjelennek olyan megoldások, melyek például képesek a gép hangja alapján előre jelezni a meghibásodást és ezzel megakadályozza a nem megfelelő alkatrészek gyártását. [2]

Emellett a szoftvereknek sokkal gyorsabban kell változniuk ahhoz, hogy működőképesek és használhatóak maradjanak - például egy mobil alkalmazás esetében a szoftvernek működnie kell a legújabb Android operációs rendszeren is, vagy képesnek kell lenniük a kommunikációra olyan eszközökkel, melyek a termék kiadásnál még nem léteztek. Emellett a szoftvernek le kell követnie a világban történő változásokat – a koronavírus időszak alatt jelentősen megváltoztak az elvárások és a használati szokások a videókonferenciás megoldásokkal szemben, a Microsoft Corporation ennek megfelelően pedig fél-egy éven belül

már olyan megoldásokkal egészítette ki a Microsoft Teams alkalmazását, melyben tantermekben található táblát lehet szimulálni és támogatja a „közös táblára rajzolást”. [1]

Egy alkatrész gyártás során a fejlesztés főként a már kiépített gyártási folyamatok optimalizálására, hatékonyabbá tételére irányul esetleg a már meglévő tervek javítására, melynek a célja a gyártási sebesség növelése, a termelési költségek optimalizálása és igény esetén a lehető legnagyobb mennyisség előállítása, ezzel szemben a szoftverfejlesztés fókuszában az új funkciók, új szolgáltatások tervezése és megvalósítása áll [3] – bár ez a kijelentés egy ideális szoftver esetében állja csak meg helyét, hiszen a gyakorlatban szoftvergyártó cégek jelentős fejlesztői és tesztelői erőforrásokat használnak fel a régi rendszerekből örökölt hibák javítására.

2.2.Az informatikai rendszerek minőségének állapota

Statisztikai adatok alapján megállapítható, hogy a világban a teljes informatikai kiadások 75%-a régóta létező, örökölt rendszerek karbantartására, javítgatására és fejlesztésére irányul, a fejlesztési kiadások 25%-a pedig a rossz minőség korrigálásával foglalkozik. [4] Mindezt annak ellenére, hogy már 1972-ben megjelent Magyarországon az automatizált programtesztelés (a szoftvertesztelés egy technikája), majd 1992-ben elkezdődött az ISO 9000 szabvány alkalmazása a szoftverfejlesztésben. [5] Esetünkben a Handyman egyik szoftverénél, a Handyman Office alkalmazásnál is megjelennek ezek a problémák, ráadásul a többi újonnan fejlesztett rendszer is ehhez a termékhez kapcsolódik, erre a termékre épül, tehát a Handyman Office-ban megjelenő hibák az új fejlesztések költségeire is hatással vannak.

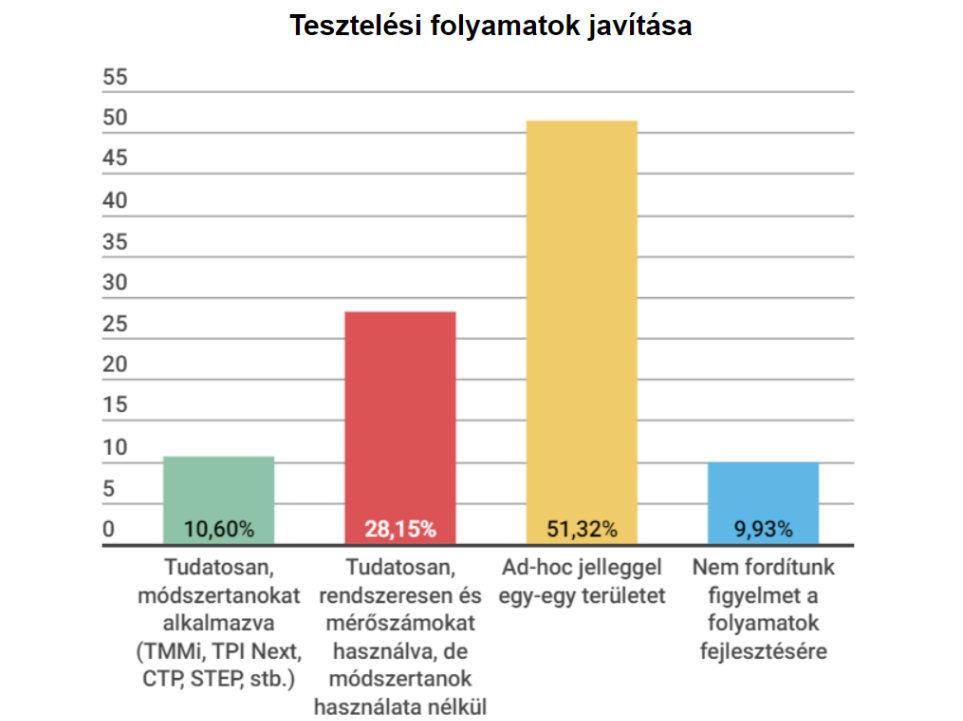
2018-as Egyesült Államokban készült statisztikák alapján a szoftverhibákból származó veszteségek, a rossz minőségű rendszerekből fakadó veszteségek majdnem 50%-át teszik ki. A szoftverfejlesztők és szoftvertesztelők összesített munkaidejének 60%-át a hibák megtalálására és javítására összpontosítják. A hibák megtalálásának és javításának a költsége az ¼-e a rossz

minőségű szoftverek által okozott veszteségeknek. Emellett a sok hiba csökkenti az alkalmazottak munkatermelékenységét, az ütemtervek betartásának esélyét és növelheti az alkalmazottak fluktuációját a munkavállalók munkakedvének csökkenésén keresztül. A becslések szerint 2017-ben átlagosan az IT projektek 52%-a az előre meghatározott költségek 189%-ába kerültek. [4] Ezekben a statisztikákban főként pénzügyi/tőzsdei iparágban alkalmazott informatikai rendszerekben megjelent hibákat vizsgáltak, melyek jelentősen nagyobb veszteséget okozhatnak, mint például a GSGproup AS által forgalmazott szoftverek. Ugyanakkor a Handyman rendszerben is megjelenhetnek kritikus, magas veszteséget okozó hibák (például abban az esetben, ha egy nagy projekt számlázásánál nem jól működik a költség összesítő funkció, vagy ha a munkavállalók munkaóráit rosszul összesítik és ezáltal nem a megfelelő fizetést kapják).

Egy 2018-as Magyarországon a szoftvertesztelők körében végzett kutatás szerint, a minőség-ellenőrzés alkalmazása a szoftvereknél a projekt költségeitől függ és 35,5%-ban még mindig kényszerpályán mozog [6], vagyis még mindig magas azon cégek száma, ahol a top menedzsment és a vezetőség nem látja a jó minőségű termék pozitív hatásait és a szoftvertesztelést csak kényszerből csinálja – ez hasonlóságot mutat a autóiparban sokszor tapasztalható „a beszállítás alapfeltétele, hogy a vevők által meghatározott tanúsításokat a termeléssel foglalkozó vállalatok megszerezzék, ezt a cégek kényszernek tekintik és nem gondolják, hogy ez a saját javukat szolgálná, ezért gyakran kiépítik a minőségügyi rendszer elemeket az auditokra, de azokat a későbbiekben nem működtetik hatékonyan” hozzáállással.

A Masterfield Oktatóközpont kutatásában a megkérdezett szoftverteszteléssel foglalkozó magyarországi munkavállalók körülbelül 30%-a elégedetlen az ügyfélnek átadott szoftverek minőségével, ezen belül 2,65% gondolja úgy, hogy a szoftver minősége jelentősen az elvárható minőségi szint alatt van és csupán 15% gondolja úgy, hogy átlag feletti minőségűek a kiadott termékek. Ennek egyik kiváltó oka lehet, ahogy azt az 1. ábra is mutatja, hogy a megkérdezettek kevesebb mint 10%-a javítja tudatosan (módszertanokat alkalmazva) a folyamatait, illetve a már korábban említett vezetői hozzáállás a minőséghez – 2018-ban a „menedzsment javítása” ennek ellenére csak az 5. helyet foglalta el a legkritikusabb fejlesztési

lépések körében. [6] Ugyanakkor ezt a felmérést szoftvertesztelők bevonásával készítették, így a felmérés adatai nem reprezentálják azokat a vállalatokat, ahol még nem jelent meg a szoftvertesztelés fontosságának felismerése – tapasztalataim alapján Magyarországon sok ilyen típusú vállalat található.

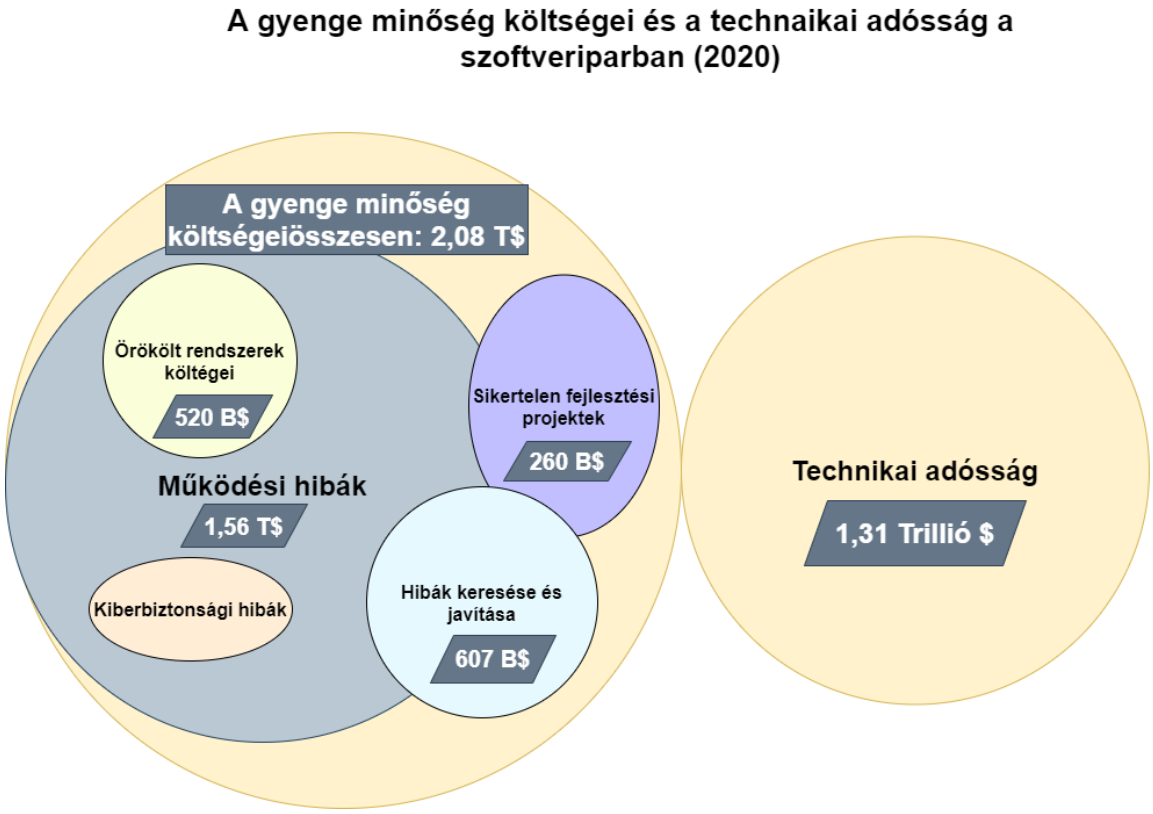


1. ábra: "Hogyan javítjátok a tesztelési folyamataitokat?" kérdésre adott tesztelői válaszok 2018-ban. Külső forrásból származó ábra [6]

Ezek alapján megállapíthatjuk, hogy a szoftveriparban sok esetben nem jelenik meg az ISO 9001:2015-ös szabványban megfogalmazott vevőközpontúság, a vevői követelmények teljesítése és törekvés a vevői elvárások meghaladására, ennek következtében pedig ezek informatikai vállalatok nehezen érhetnek el tartós sikert, hiszen rossz minőségű termékekkel nem érhetik el a vevők és más érdekelt felek bizalmát. [7]

2.3. Napjaink legfontosabb és legtöbb kárt okozó szoftvergyengeségei

Napjainkban egy adatközpontú, erősen digitalizált világban élünk, ahol egyre nagyobb hangsúlyt kap a magánélet és a személyes információk biztonsága. A digitális megoldások sérülékenységei miatt mostanra személyes adatok millió váltak nyilvánossá az adatkiszivárgások és kibertámadások miatt. [8] Ezzel párhuzamosan, ahogy a 2. ábra is mutatja egyre nagyobb és nagyobb költségei vannak a gyenge szoftverminőségnek és az egyre növekvő technikai adósságnak. [9]



2. ábra: A gyenge minőség és a technikai adósság nagysága (dollárban kifejezve) 2020-ban az Egyesült Államokban. Saját készítésű ábra, Saját készítésű ábra, külső forrás alapján [9]

A 2020 -2022 között végzett kutatások szerint a szoftverminőség állapota nem javult, sőt a statisztikák alapján romló tendenciát mutat:

- A **szoftversérülékenységek és gyengeségek** miatt a kiberbűnözéssel összefüggő veszteségek jelentősen nőttek, számos kritikus infrastruktúrát ért támadás, melyek jelentős károkat okoztak. [9]

- Jelentősen megszaporodtak a **szoftverellátási lánc**ból adódó problémák, 2021-ben 650%-kal nőtt a nyílt forráskódú szoftverek felhasználásából adódó rendszer meghibásodások száma – egy közepes méretű alkalmazás körülbelül 200-300 fajta harmadik féltől származó kódot tartalmaz. [9] A Handyman alkalmazás is rengeteg külső féltől származó kódot használ, és tapasztalataim alapján valóban sokszor előfordul, hogy a termékben talált hiba harmadik féltől származik.

- A szoftverekhez kapcsolódó **technikai adósság** mérete is egyre növekvő tendenciát mutat. A szoftverfejlesztési folyamatok során gyakran előfordul, hogy a tesztelő csapat visszajelzései alapján az adott verzió kiadása előtt elkezdik javítani a tesztelő csapat által talált hibákat, ugyanakkor a hibajavítások sok esetben új hibák megjelenéséhez vezetnek. A „zéró nyitott hiba” állapotot általában a cégek úgy érik el, hogy a kritikus, magas prioritású hibákat javítják, majd a megmaradó hibákat a következő verzióra halasztják, vagyis a hiba „Backlog” -ba kerül. A javításra szoruló hibák száma így drasztikusan elkezd növekedni, majd a probléma megoldása az idő előrehaladásával egyre nehezebbé válik. [10] Ez a probléma a Handyman termékcsomagnál is megjelenik, a „Backlog” -ban található hibák száma olyan méretet öltött az évek alatt, hogy az összes hiba megoldása több hónapra lekötné a teljes fejlesztő és tesztelő csapat erőforrásait.

A szoftverek gyengeségei a hibákból, sebezhetőségekből és egyéb kódban, architektúrában, megvalósításban és tervezésben megtalálható hiányosságokból adódnak. [11]Az előzőekben bemutatott három probléma megoldása mindenképpen figyelmet igényel a Handyman termékcsomag szemszögéből is, viszont a diplomadolgozatom terjedelmi korlátai miatt, csak a „szoftver sérülékenységek és gyengeségek” témával fogok mélyebben foglalkozni.

2.3.1. Szoftverek sérülékenységei

A szoftver sérülékenység alatt olyan gyenge pontokat, hibákat, hibás tervezési döntéseket, helytelen implementációkat, programozási és konfigurációs hibákat értünk, melyeket ki lehet használni a kibertámadások során. A sérülékenységek lehetővé teszik a támadók számára, hogy belépjenek a rendszerbe és onnan érzékeny adatokat lopjanak el, módosítsák/töröljék ezeket az adatokat, károsítsák a rendszer működését, vagy akár kiterjesszék a támadást más rendszerekre is. [9]

Az elmúlt éveket az informatika világában a bűnelkövetési célú szoftverek terjedése, a mobileszközökön futtatható kártékony szoftverek számának robbanásszerű növekedése és a botnet hálózatok[1] egyre növekvő fenyegetése jellemezte. Ezen támadások legfőbb célpontja a vállalati szféra, a kritikus infrastruktúra és a kormányzati szervek. Az ilyen típusú szoftverrendszerek elleni sikeres támadás katasztrofális lehet, bizonyos esetekben (például autókban elhelyezett szoftver esetében) személyi sérülésekkel, vagy akár halállal is járhat. [12] A Handyman termékcsomag forgalmazásának a célcsoportja a vállalati szféra, tehát a tapasztalatok alapján a kibertámadások egyik kiemelkedő célcsoportjába tartozik.

Ha egy szoftverfejlesztő cég nem végez megfelelő IT biztonsági ellenőrzést az általuk fejlesztett terméken, az jelentős kockázatot jelenthet, hiszen enélkül a szoftverben benne maradhatnak olyan hibák és sérülékenységek, melyeket a támadók kihasználhatnak, melynek következménye lehet a rendszer instabilitása vagy hibás működése – vagyis a rendszer nem lesz képes a felhasználók termékkel kapcsolatos alapkövetelményeinek kielégítésére. [13] Vagyis kijelenthetjük, hogy a rendszer biztonságával kapcsolatos hibákat minőségbiztosítási hibának kell tekinteni. Ugyan az informatikai iparágban az IT biztonság és a minőségbiztosítás két elkülönített szakma, ennek ellenére még sem szabad őket külön kezelni, hiszen mindkettő

[1] **Botnet hálózat**: Fertőzött informatikai eszközökből álló hálózat, amelyet a botnet gazdája többféle károkozásra is alkalmazhat. A fertőzött munkaállomások felhasználásának célja főképp kéretlen levelek kiküldése, szolgáltatás megtagadást okozó támadások (Denial-of-Service ─ DoS) indítása, vagy éppen szenzitív (például banki) adatok eltulajdonítása. Felhasznált forrás: [32]

célja a jó minőségű, megbízható szoftverek létrehozása. [14] Tapasztalataim alapján, abban az esetben, ha egy szoftverfejlesztő cég nem rendelkezik IT biztonsági csapattal és valamilyen esemény hatására külső céget bíznak meg a biztonsági rések felfedésére, a külső ellenőrzés eredményeként majdnem minden esetben találnak több sérülékenységet is, amelyre a fejlesztés során nem derült fény. Az IT biztonság egyre növekvő fontosságának fényében érdekes információkat szolgáltat az, hogy a szoftverfejlesztéssel foglalkozó cégek mennyire veszik komolyan, hogy erőforrásokat szabadítsanak fel az ilyen típusú sérülékenységekből származó károk megelőzésére. A Coleman Park kutató cég 2022 áprilisában világszintű kutatást végzett a Dynatrace megbízásából az IT biztonság aktuális helyzetéről. Ennek az Európai kontinensre vonatkozó fontosabb adatai az 1. táblázatban láthatóak (mivel a GSGroup Európában tevékenykedik, azon belül is inkább az északi régiókban, ezért ezeket az adatokat tartom relevánsnak) [15]:

1. táblázat: IT biztonság helyzete Európában

Vizsgált paraméter/Ország	**Német**	**Svéd**	**Finn**	**Norvég**
Szervezeten belül többrétegű kiber biztonsági álláspontot képviselnek	64%	73%	59%	70%
Szervezetek képesek kezelni a sebezhetőségeket éles környezetben, futtás közben	40%	39%	53%	36%
Szervezet rendelkezik olyan IT biztonsági csapattal mely teljes rálátással rendelkezik valós időben a fejlesztés során felhasznált kódokra és alkalmazásokra	24%	25%	17%	19%
Kárkezelési jegyek (sérülékenység kihasználása után) számának növekedése	65%	70%	67%	47%
Szervezeten belül megfelelő érettségi szinttel rendelkező DevSecOps[2] módszertan található	31%	36%	24%	42%
Olyan IT biztonsággal kapcsolatos riasztások átlagos napi száma melyek beavatkozást igényeltek 2022-ben	656 db	498 db	625 db	870 db

Saját készítésű táblázat, külső forrás [15]

[2] **DevSecOps**: Olyan szoftverfejlesztési módszertan, melyben a szoftverfejlesztéssel, IT biztonsággal és üzemeltetéssel foglalkozó csapatok együttműködnek, annak érdekében hogy megbízhatóbb és biztonságosabb szoftvert hozzanak létre. A módszertan célja, hogy az IT biztonságot már az alkalmazásfejlesztési folyamat kezdeti szakaszában alkalmazzák, és az végigkísérje a fejlsztés teljes folyamatát.

Az 1. táblázatban található adatok alapján szembetűnő, hogy a legtöbb valós IT biztonsági riasztás Norvégiában történt, ami a dolgozatom szemszögéből azért fontos mivel, a Handyman termékcsomagot Norvégiában használják a legtöbben.

A MITER kutatásai alapján összeállításra került 25 leggyakoribb és legveszélyesebb szoftver gyengeség és sérülékenység 2020 és 2022 között. A lista létrehozása során a MITER minden ismert gyengeséget értékelt a CWE (Common Weaknesses Enumeration – Gyakori gyengeségek listája) adatbázisából, a gyengeségek elterjedtsége és súlyossága alapján, miután elemezték az NVD (National Vulnerability Database – Nemzeti Sérülékenységi adatbázis) és a CISA által készített (Known Exploited Vulnerabilities – Ismert kihasznált sebezhetőségek) katalógust. A lista elkészítése során 37 899 CWE került átvizsgálásra. [9]

A 25 leggyakoribb és legveszélyesebb szoftver gyengeség és sérülékenység súlyosság és gyakoriság szerinti csökkenő sorrendben: [11]

01. **CWE-787**: **„Out-of-bounds Write”** A termék a tervezett puffer vége/kezdete előtt ír adatokat.
02. **CWE-79**: **„CSS”** A termék nem ellenőrzi a felhasználó által vezérelhető bemenetet, mielőtt azt kimenetként jelenítené meg más felhasználónak.
03. **CWE-89: „SQL Injection”** A szoftver az SQL-parancsot külsőleg módosítható bemenet felhasználásával hozza létre, de nem semlegesíti azokat a speciális részeket, amelyek módosíthatják az SQL-parancsot, amikor azt beküldik.
04. **CWE-20: „Improper Input Validation”** A termék fogad bemeneti adatokat, de nem vagy hibásan ellenőrzi, hogy a bemenet rendelkezik-e az adatok biztonságos és helyes feldolgozásához szükséges tulajdonságokkal.
05. **CWE-125: „Out-of-bounds Read”** A termék a tervezett puffer vége után vagy a kezdete előtt olvassa be az adatokat.
06. **CWE-78: „OS Command Injection”** A szoftver az OS parancsokat külsőleg befolyásolható bemenet felhasználásával hozza létre, de nem semlegesíti azokat a speciális elemeket, amelyek módosíthatják az OS parancsot, amikor azt tovább küldik.

07. **CWE-416: „Use After Free”** A memória felszabadítása utáni hivatkozás a program összeomlásához, váratlan értékek használatához vagy kód futtatásához vezethet.
08. **CWE-22: „Path Traversal”** A szoftver külső bemenetet használ az elérési út létrehozása, de nem semlegesíti megfelelően azokat a speciális elemeket melyek a kívül a korlátozott mappán kívülre mutatnak.
09. **CWE-352: „Cross-Site Request Forgery”** A webalkalmazás nem képes ellenőrizni, hogy a kérelmet benyújtó felhasználó szándékosan nyújtott-e be a kérést.
10. **CWE-434: „Unrestricted Upload of File with Dangerous Type”** A szoftver lehetővé teszi veszélyes típusú fájlok feltöltését, melyek feldolgozásra kerülnek a termékben.
11. **CWE-476: „NULL Pointer Dereference”** NULL mutató hivatkozás törlése.
12. **CWE-502: „Deserialization of Untrusted Data”** A termék deszerializálja a nem megbízható adatokat anélkül, hogy megfelelően ellenőrizné a kapott adatok érvényességét.
13. **CWE-190: „Integer Overflow or Wraparound”** A program olyan számítást hajt végre, amely egész szám túlcsordulást eredményezhet.
14. **CWE-287: „Improper Authentication”** A szoftver nem megfelelően ellenőrzi a felhasználók személyazonosságát.
15. **CWE-798: „Use of Hard-coded Credentials”**
    A szoftverben be vannak égetve a hitelesítő, vagy kriptográfiai adatok.
16. **CWE-862: „Missing Authorization”**
    A termék nem hajt végre jogosultság-ellenőrzést, amikor egy felhasználó hozzáférést kér egy erőforráshoz vagy műveletet hajt végre.
17. **CWE-77: „Command Injection”**
    A termék a parancsot egy külsőleg befolyásolható bemenet felhasználásával hozza létre, de nem semlegesíti azokat a speciális elemeket, amelyek módosíthatják a parancsot, amikor azt beküldik.
18. **CWE-306: „Missing Authentication for Critical Function”**
    A termék nem végez hitelesítést olyan funkciók esetében, amelyek felhasználói azonosítást igényelnek, vagy jelentős mennyiségű erőforrást fogyasztanak.

19. **CWE-119: „Improper Restriction of Operations within the Bounds of Memory Buffer”** A szoftver tud olvasni vagy írni olyan memóriahelyről, amely kívül esik a puffer tervezett határán.
20. **CWE-276: „Incorrect Default Permissions”** A telepítés során a telepítő fájlok engedélyei úgy vannak beállítva, hogy bárki módosíthassa ezeket a fájlokat.
21. **CWE-918: „SSRF”** A webszerver lekéri egy URL-címnek a tartalmát, de nem biztosítja kellőképpen a kérés elküldését.
22. **CWE-362: „Race Condition”** A program olyan kódsorozatot tartalmaz, amely más kóddal párhuzamosan futhat, és a kódsorozat ideiglenes, kizárólagos hozzáférést igényel egy megosztott erőforráshoz, de eközben a megosztott erőforrás egy másik, párhuzamosan futó kódsorozattal módosítható.
23. **CWE-400: „Uncontrolled Resource Consumption”** A termék nem szabályozza megfelelően egy korlátozott erőforrás kiosztását és karbantartását, ami végül a rendelkezésre álló erőforrások kimerüléséhez vezet.
24. **CWE-611: „Improper Restriction of XML External Entity Reference”** A szoftver olyan XML-fájlt olvas, ami URI-vel rendelkező XML-entitásokat tartalmazhat, amelyek a külső dokumentumokra mutatnak, így a szoftver nem megfelelő dokumentumokat ágyaz be a kimenetébe.
25. **CWE-94: „Improper Control of Generation of Code”** A termék egy kódszegmens részeit generáltatja, de nem semlegesíti azokat a speciális elemeket, amelyek módosíthatják a kívánt kódszegmens szintaxisát vagy viselkedését.

2.4. Az agilis szoftverfejlesztés és a Scrum[3] módszertan

A számítógépes technológiák folyamatos fejlődésével, kialakult az igény a hatékonyabb fejlesztési folyamatok kialakítására, melyek képesek alkalmazkodni az emberek folyamatosan változó igényeihez. Az agilis szoftverfejlesztés koncepciója 2001-ben jelent meg, ekkor került publikálásra az agilis kiáltvány, napjainkra pedig jelentősen elterjedt, mivel egyre több vállalat

[3] **Scrum**: Összetett problémák megoldására alkalmas inkrementális és iteratív keretrendszer, melyet gyakran használnak az agilis szoftverfejlesztés eszközeként.

kezdi felismerni a módszer előnyeit. [16] Manapság az agilis fejlesztésnek már több fajtája van, a GSGroup Handyman részlegén az agilis szoftverfejlesztésen belül a Scrum keretrendszert alkalmazzák.

Az agilis kiáltvány tartalma:

„A szoftverfejlesztés hatékonyabb módját tárjuk fel saját tevékenységünk és a másoknak nyújtott segítség útján. E munka eredményeképpen megtanultuk értékelni:

Az egyéneket és a személyes kommunikációt a módszertanokkal és eszközökkel szemben

A működő szoftvert az átfogó dokumentációval szemben

A megrendelővel történő együttműködést a szerződéses egyeztetéssel szemben

A változás iránti készséget a tervek szolgai követésével szemben

Azaz, annak ellenére, hogy a jobb oldalon szereplő tételek is értékkel bírnak, mi többre tartjuk a bal oldalon feltüntetetteket." [17]

A Scrum módszertan alapötlete, hogy egy új, komplex termék fejlesztése esetén nem lehet előre meghatározni a végső termék specifikációt. A Scrum tapasztalatokból és adott tudásból felépülő folyamatellenőrzési elméleten alapul. A folyamatellenőrzés alapja a transzparencia (a felelősösök lássák a szükséges nézőpontokat), az ellenőrzés (haladás, munkatermékek ellenőrzése az eltérések észleléséért) és a korrekció. [18]

A Scrum csapatok önszerveződőek, vagyis a csapatok döntik el hogy hogyan lehet a leghatékonyabban elvégezni az adott feladatot, illetve kereszt funkcionálisak, ami ezt jelenti, hogy a csapat tagjai rendelkeznek a feladat megfelelő minőségű elvégzéséhez szükséges tudással és nincsenek függőségei olyan személyekkel, akik nem részei az adott csapatnak. [18] A szoftveriparban a kereszt-funkcionális csapat gyakorlatilag azt jelenti, hogy a fejlesztő csapatban a fejlesztők mellett dolgoznak szoftvertesztelők, terméktervezők és üzemeltetési mérnökök is, vagy legalábbis van olyan személy, aki rendelkezik a szükséges szaktudással az adott területeken. [19]

A Scrum csapatok általában maximum 10 főből állnak, a csapat résztvevői:

1. **Terméktulajdonos**: Feladata az üzleti, vevői, piaci követelmények alapján a mérnöki csapat által elvégzendő munka priorizálása. Arra fókuszál, hogy a fejlesztőcsapat a lehető legtöbb értéket nyújtsa az üzlet számára. [19]

2. **Scrum mester**: Feladata a Scrum keretrendszer megértése és szabályainak betartatása a csapaton belül, emellett a Scrum csapaton kívüli érintettekkel segít megértetni azt, hogy milyen interakciókkal lehet maximális létrehozott értéket elérni A Scrum mester támogatja a terméktulajdonost, a fejlesztőcsapatot és a szervezetet is. [18]

3. **Fejlesztő csapat:** Szakemberek, akiknek a feladata, hogy a Sprintek[4] végén elkészüljön a termék egy potenciálisan kiadható inkrementuma. A fejlesztőcsapat tagjai felelősek a munka szervezéséért és menedzsmentjéért. A csapat tagjai speciális tudással és tapasztalattal rendelkeznek, de a Scrum keretrendszerben minden fejlesztő csapattag „fejlesztő" -ként jelenik meg. A fejlesztő csapat mérete ideális esetben 3-9 főből áll. [18]

A Scrum keretrendszer meghatároz bizonyos tevékenységeket, úgynevezett „Scrum eseményeket", melyeket a Scrum csapatoknak folyamatosan végre kell hajtaniuk:

1. **Teendőlista (Backlog) szervezése**: Ez a tevékenység a terméktulajdonos felelőssége, a priorizálás az ügyfelektől és a fejlesztő csapattól kapott visszajelzések alapján történik. [19]

2. **Sprint tervezése**: A tervezés során meghatározásra kerül az adott sprint során elvégzendő feladatok köre és az, hogy hogyan lehet elvégezni az adott inkrementum elkészítését. A tervezésben a teljes Scrum csapat részt vesz. [18]

[4] **Sprint**: A Scrum módszertan azonos hosszúságú időegységekre bontja a szoftverfejlesztést. Egy-egy ilyen időegység neve a Sprint.

3. **Sprint**: Általában a sprint hosszúsága két hét, de gyakorlatban előfordulnak egy hetes, illetve egy hónapos sprintek is. Minden fejlesztési tevékenység a sprint során történik. [19]

4. **Napi Scrum – Stand up**: A sprint során mindennap tart egy megbeszélést a fejlesztőcsapat, maximum 15 perc időtartamban, ahol a csapat tagjai kifejtik, hogy mivel foglalkoztak előző nap, mivel fognak foglalkozni ma, illetve vannak-e olyan tényezők, amik gátolják őket a munka elvégzésében. [18]

5. **Sprint áttekintés**: A Sprint végén történik, a Scrum csapat tagjainak és az érintettek részvételével. A fejlesztő csapat bemutatja az elkészült inkrementumot, az érintettek pedig visszajelzést adnak. A Sprint áttekintés eredménye a módosított termék Backlog, ami meghatározza a következő Sprint során megvalósítandó tételeket. [18]

6. **Sprint visszatekintés (retrospektív):** A Scrum csapat megvitatja, hogy mik voltak azok a tényezők, amik jól működtek az előző Sprintben, illetve mik voltak azok, amik rosszak voltak és hogy ezeken a dolgokon hogyan lehetne javítani. [19]

2.5. Szoftvertesztelési folyamatok fejlesztését támogató modellek

Minden folyamat egyedi és ezért a folyamatfejlesztések megkezdése előtt fontos a folyamat megértése, ebben segít a folyamatábra, hiszen segít a komplex feladatokat egyszerűen és átláthatóan leírni. A folyamatábra a folyamat tevékenységeinek, eseményeinek és ezek logikai kapcsolódásának vizuális szemléltetése. [20]

Napjainkra számos modell készült a tesztelési folyamatok javítása érdekében, ezek a modellek kifejezetten szoftvertesztelésre lettek kialakítva, ezért sokkal jobban alkalmazhatóak, mint az általános folyamatfejlesztési modellek. A szoftver tesztelési folyamatok folyamatos fejlesztése ugyanolyan fontosságú egy szoftverfejlesztő vállalat életében, mint maga a szoftvertesztelési tevékenység. [21] A szoftvertesztelési referencia modellek kiindulási pontot

adnak a tesztelési folyamatok javításának elindításához, mivel viszonyítási pontot adnak arról, hogy milyen egy jól működő, hatékony folyamat. [22] .

Ebben a fejezetben a manapság leggyakrabban alkalmazott modellek a TMMi [5]és a TPI Next[6] modellek kerülnek bemutatásra. Ugyan a diplomadolgozatom során nem célom vállalatok összehasonlítása, de az alábbiakban bemutatott modellek a folyamat fejlesztés mellett alkalmasak a szoftveriparban tevékenykedő vállalatok összehasonlítására és értékelésére is. [22]

A TPI Next és a TMMi különböző architektúrával rendelkezik. A TMMi egy szakaszos modell, ami azt jelenti, hogy az adott érettségi szintek folyamat területekből épülnek fel és az adott folyamat területek csak az adott érettségi szinten vannak jelen. Ahhoz, hogy a szervezet magasabb érettségi szintre léphessen, teljesítenie kell az adott érettségi szint folyamatterületeinek minden követelményét. A TPI Next egy folytonos modell, melyben a fő hangsúly a folyamatterületeken van – ezekhez kerül hozzárendelésre az érettségi szint, így a modell lehetővé teszi, hogy az egyes folyamatterületeket érettségi szintje külön-külön értékelhető legyen. [22]

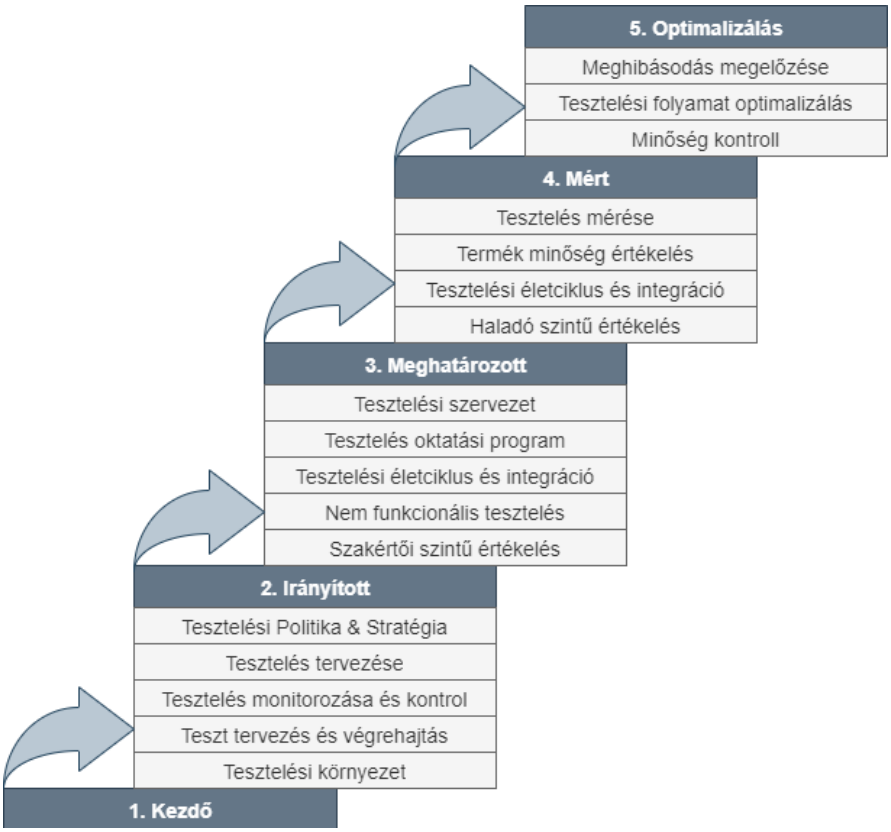
2.5.1. A TMMi (Test Maturity Model integration) modell

A TMMi szoftvertesztelésben adaptálható, folyamatokkal foglalkozó érettségi modell a CMMI[7] modell kiegészítéseként jött létre. [21] A CMMI egy olyan módszertan melyet a szoftverfejlesztési folyamat fejlesztésére és finomítására használnak és a kezdeti módszerektől indulva az optimalizált folyamatig meghatározza a folyamatok érettségi szintjét (5 szintre bontva), ugyanakkor a szoftvertesztelés csak érintőlegesen jelenik meg benne az ellenőrzés részeként. [22] A TMMi részben a szoftver iparágban megjelenő gyakorlatokon alapul,

[5] **TMMi**: Test Maturity Model Integration – Tesztelés érettségi modell integráció
[6] **TPI Next**: Test Process Improvement Next - Tesztelési folyamat fejlesztés
[7] **CMMI**: Capability Maturity Model Integration – Képesség érettségi modell integráció

kifejlesztése során számos iparági tesztelési és szoftverminőségi modellt vettek figyelembe. Manapság a TMMi-t számos iparágban használják, különböző életciklus-modellekkel kombinálva, és elmondható, hogy a tesztelési folyamatok fejlesztésének vezető modelljeként tartják számon világszerte. [23]

A világszintű elterjedés miatt a TMMi Foundation, a TMMi modell „gondozója” szoros kapcsolatban van számos ISO/IEC szervezettel annak érdekében, hogy a lehető legoptimálisabban tudja karbantartani és fejleszteni a modellt. A TMMi Foundation 2014 óta tagja és támogatója az ISO/IEC 26 munkacsoportnak melynek egyik célja, hogy a TMMi referencia modell összhangban legyen az ISO 29119 szabványban található fogalmakkal és definíciókkal. [23] A modell a folyamat referencia modellek csoportjába tartozik, vagyis megadja, hogy milyen folyamat fejlesztési tevékenységeket kell elvégezni a következő érettségi szint eléréséhez. A TMMi modell teljes mértékben a CMMI modell szerkezetét használja, hiszen ugyanúgy 5 folyamat érettségi szintet határoz meg, és ezeket rendeli hozzá a tesztelési folyamatokhoz. A szervezet csak akkor léphet a következő érettségi szintre, ha az alatta lévő szinten meghatározott célok legalább 85%-ban teljesülnek. Az 5 érettségi szint a 3. ábra alapján épül fel. [24]



3. ábra: TMMi által meghatározott folyamat érettségi szintek. Saját készítésű ábra. Saját készítésű ábra, külső forrás alapján: [25]

A TMMi modell érettségi szintjeinek bemutatása: [25]

- **1. szint – Kezdő**

  A kezdő/kiinduló szinten elhelyezkedő szervezetek jellemzője, hogy a szoftverfejlesztési projektek túllépnek az előre meghatározott idő és költségkereteken, illetve a szoftverminőség szintje alacsony. Ezen a szinten a szoftver minőségbiztosítása csupán a hibakeresésről szól, nincsenek formalizált folyamatok és maga a szoftvertesztelés nem szervezetten/irányítottan történik. [26]

- **2. szint - Irányított:**

  Fókuszba kerül a vállalati szintű tesztelési politika és a tesztelési stratégia. A szoftvertesztelés szervezett folyamattá válik és szervezeti szinten kialakulnak az ezzel kapcsolatos alapelvek. Ezen a szinten már megjelenik a tesztelés megtervezése és a tesztelési eredmények nyomon követése, illetve minden projekt teszteléséhez rendelkezésre állnak a megfelelő tesztkörnyezetek[8]. [21]

- **3. szint - Meghatározott:**

  Ezen a szinten biztosított, hogy minden projekt ugyanazokat a szabványokat és folyamatokat alkalmazza a szervezetben. A csapatok szervezetten működnek – a működés szabályai dokumentáltak, illetve kialakításra kerültek az oktatási programok, a munkavállalók előre meneteli útvonalai, illetve az új kollégák betanításának lépései is definiáltak. A szoftvertesztelés megfelelően integrálva van a fejlesztési életciklusban, illetve a tesztelés jelen van már a projektek korai életciklusában is. A projektek során megjelenik a nem-funkcionális[9] tesztelés tervezése és végrehajtása. Ezen a szinten jelenik meg először a felülvizsgálat, bár itt még csak a teszteléssel foglalkozó munkavállalók végzik azt el. [26]

[8] **Tesztkörnyezet**: "Az a környezet, ami hardvert, beműszerezést, szimulátorokat, szoftvereszközöket és egyéb tesztelést segítő elemeket tartalmaz annak érdekében, hogy a tesztelést le lehessen folytatni". Forrás: [33]

[9] **Nem-funkcionális tesztelés**: Célja nem a funkcionalitások tesztelése, hanem egyéb termék jellemzők, például akadálymentesség, kompatibilitás, terhelhetőség, teljesítmény stb. vizsgálata.

- **4. szint - Mért:**

  Ezen a szinten a fókusz a tevékenységek és eredmények mérésén van. A mérések már a projektek korai szakaszában megjelennek és a lehető legjobban próbálják biztosítani, hogy a termék hibamentes (bár a tesztelés alapelvei alapján, hibamentes termék nem létezik) azáltal, hogy a mért értékek alapot nyújtanak a tevékenységek, a folyamatok és eredmények értékeléséhez. A felülvizsgálat az előző szinten bevezetett gyakorlatokra építve, már haladó szinten jelenik meg. [27]

- **5. szint - Optimalizálás:**

  Minden tevékenység és eredmény nyomon követés és irányítás alatt van, a tevékenységek és folyamatok optimalizálásán van a hangsúly, annak érdekében hogy biztosítva legyen a folyamatos fejlesztés a hibamegelőzéssel, a minőség optimalizálással, az automatizált tesztelés, illetve a refaktorálás[10] minél szélesebb körű használatával. [27]

Az érettségi szinteken belül úgynevezett „Folyamat területek” vannak, melyek egymáshoz kapcsolódó folyamatokat foglalnak magukban – ezek a területek egyébként megtalálhatóak a CMMI modellben is, csak ebben az esetben a folyamat területek a szoftvertesztelési folyamatra specializálódnak. [24]

Az érettségi szinteken található folyamatterületek további három komponensből állnak: [24]

- **Kötelező komponensek:**

  Ezek a komponensek olyan specifikus célokat[11] és általános célokat[12] tartalmaznak, melyeket teljesíteni kell az adott folyamatterület és a hozzá tartozó érettségi szint teljesítéséhez.

[10] **Refaktorálás**: Informatikában a programkód átalakítást jelenti úgy, hogy a szoftver viselkedése nem változik meg. Ezzel egy könyebben érthető, jönyebben karbantartható és job minőségű program kód jön létre.

[11] **Specifikus cél a TMMi-ban**: Egy konkrét folyamatterületre jellemző (gyakorlatilag nem ismétlődő) cél.

- **Elvárt komponensek:**

  A speciális és az általános célokat a speciális és az általános gyakorlatok írják le, melyek az elvárt komponensek csoportjába tartoznak. Ezek a gyakorlatok és alternatívák a célok elérését segítik.

- **Tájékoztató komponensek:**

  A tájékoztató komponensek a gyakorlatokhoz tartozó „algyakorlatok”, például jegyzetek, munkatermékek. Céljuk, hogy az általános és a speciális gyakorlatokról több információt szolgáltassanak.

A modell feltételezi, hogy minden szervezet legalább az 1. (Kezdő) szinten helyezkedik el. Az egyes szintek teljesítése biztosítja, hogy a megfelelő tevékenységeket elvégezték annak érdekében, hogy a következő szint biztos alapokra épüljön, mivel a modell a fenti ábrán látható módon szakaszos architektúrával rendelkezik. A különböző érettségi szinteken való áthaladás növeli a tesztelés és termékminőség menedzsment azon képességét, hogy minél jobban alkalmazkodjon a szervezet igényeihez. Ezzel egy időben javul a vállalat által fejlesztett szoftverek minősége, csökken a termékben található hibák száma és javul a szoftvertesztelés hatékonysága. [23]

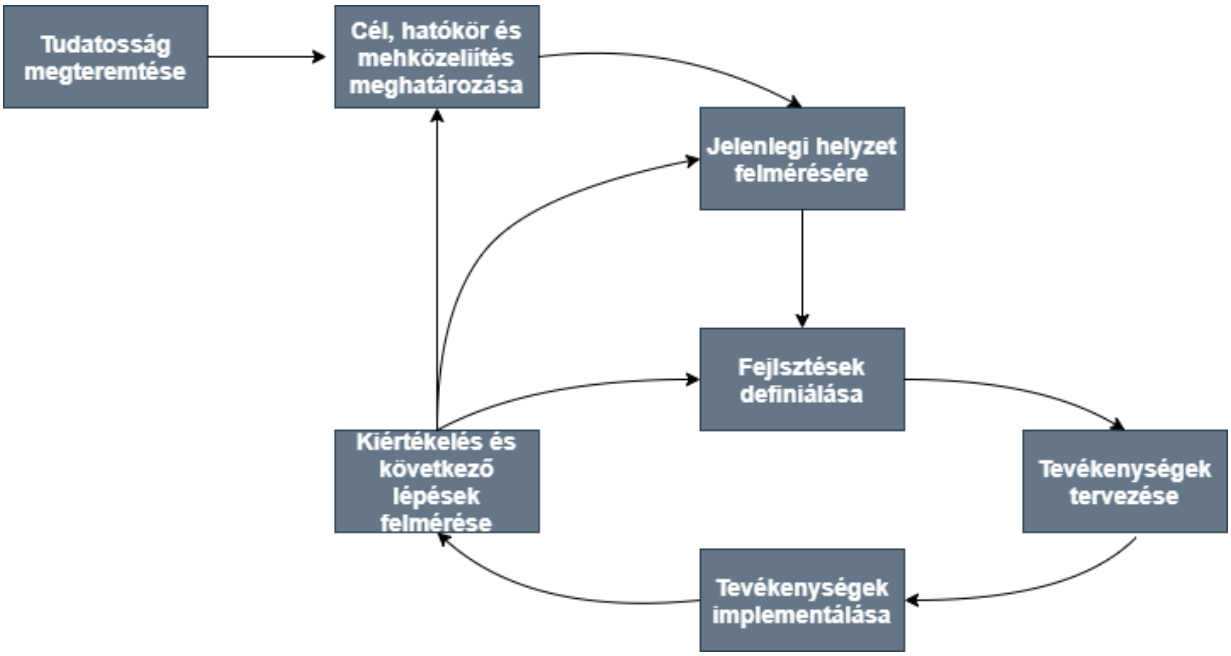
2.5.2. A TPI Next (Test Process Improvement Next) modell

A TPI Next a TPI modell utódja, melyet a holland Sogeti vállalat fejlesztett ki. Az eredeti TPI megközelítéshez képest a TPI Next kulcsterületeinek számát 16-ra csökkentették és további elemeket vezettek be, annak érdekében, hogy a modell hatékonyabban alkalmazkodjon az iparági igényekhez a tesztelési folyamatok javítása terén. [21] A TPI Next modell a széleskörben elterjedt PDCA (Tervezés – Cselekvés – Ellenőrzés - Beavatkozás) ciklus alapján készült. [24] A TPI Next modell a TMMi modellel szemben független a szoftverfejlesztési folyamatfejlesztési modellektől. A modell célja, hogy a szoftvertesztelés minél jobban

[12] **Általános cél a TMMi-ban**: Több folyamatterületre, vagy általánosan minden területre jellemző (gyakorlatilag ismétlődő) cél

kielégítse az üzleti célokat, a folyamatok optimalizálásával növelje a szoftvertesztelés hatékonyságát, csökkentse a költségeket és a tesztelésre fordított időt. [28]

A TPI Next gyakorlatilag egy folyamatos fejlesztési folyamat. A folyamat tevékenységeit a 4. ábra mutatja be. A folyamatban megjelenő PDCA ciklus lehetővé teszi, hogy a folyamatok fejlesztése folytonos legyen addig a pontig, ameddig szükség van rá. [26]



4. ábra: TPI Next folytonos fejlesztési folyamat lépései. Saját készítésű ábra. Saját készítésű ábra külső forrás alapján [29]

A modell 16 kulcs területet, 4 érettségi szintet és 157 ellenőrző pontot tartalmaz, emellett a modellben megjelennek a csoportok, jóváhagyók és a fejlesztési javaslatok. [28]

A TPI Next modell az alábbiakban felsorolt elemekből épül fel. [29]

- **Kulcs területek:**

  A TPI kulcsterületek gyakorlatilag a 2.5.1 fejezetben bemutatott TMMi folyamat terültek leképezése, a kulcs területek gyakorlatilag egymáshoz kapcsolódó tevékenységek halmazát

jelölik. A kulcsterületek lefedik a tesztelési folyamat speciális aspektusait, mint például a tesztelési stratégiát, metrikákat, tesztelési eszközöket és a tesztkörnyezetet.

- **Érettségi szintek:**
  A kulcs területek elemzését a területekhez meghatározott érettségi szintek támogatják. Az érettségi szintek definiálva vannak az ellenőrző pontokhoz, kulcsterületekhez és tesztelési folyamatokhoz is. A 4 szint: Kezdő, Kontrollált, Hatékony, Optimalizálás.
- **Ellenőrző pontok:**
  Olyan állítások melyekre a TPI Next keretrendszer használata során igennel, illetve nemmel kell válaszolni. Az ellenőrző pontok adják a modell alapját. Minden kulcsterület saját egyedi ellenőrző pontokkal rendelkezik.
- **Csoportok:**
  Az érettségi szinteknél pontosabb, kisebb felosztású egységekkel méri az ellenőrző pontok érettségét.
- **Engedélyező tevékenységek:**
  Olyan tevékenységeket jelöl, melyeket a minőségbiztosítási csapaton kívülálló személyeknek kell elvégeznie az adott ellenőrző pontok teljesítéséhez.
- **Fejlesztési javaslatok:**
  A fejlesztési javaslatok az ellenőrző pontokhoz tartoznak és ahhoz adnak fejlesztési javaslatokat, hogy az adott ellenőrzési pont teljesíthető legyen.

2.5.3. A TPI Next és TMMi modell összehasonlítása

Az első és egyben legfontosabb jellemzője mindkét modellnek, hogy abban az esetben, ha a szervezet érintett területei (főként a minőségbiztosítási csoport és a menedzsment) nem elkötelezettek a modellek alkalmazásán keresztül a folyamatok fejlesztésére, akkor a modellek bevezetését nem fogják jelentős eredmények és javulás kísérni. [26] Tehát az első lépésként minden esetben fel kell mérni a szervezet projekt iránti elkötelezettségét, és abban az esetben, ha nincs meg a motiváció, akkor ki kell alakítani az igényt a szervezetben. Emellett a menedzsment elköteleződése kiemelten fontos, hiszen a modellek kialakítása a szervezetben humán erőforrásokat igényel, illetve, ha a vállalat nem rendelkezik megfelelő szaktudással a

modellekről, szükség lehet külső tanácsadók bevonására, illetve oktatások megszervezésére is (jelenleg Magyarországon egy TMMi oktatás 300 ezer forint körüli áron érhető el egy fő részére). Tehát a modell bevezetésének fő költség elemei a projektben résztvevő kollégák ideje és az oktatási programokra fordított kiadások. Emellett a modellek bevezetése során megjelenhet a szervezeten belül a változtatásokkal szembeni ellenállás, ezért fontos, hogy a modell előnyei, és a hozzájárulása a szervezet fejlődéséhez megfelelően legyen kommunikálva a szervezet minden szintjén, a kollégák rendelkezzenek a változáshoz szükséges tudással és elkötelezettek, illetve motiváltak legyenek a modellek bevezetésével kapcsolatban. [24]

Mindkét modell használata információt szolgáltat a tesztelési folyamatok aktuális állapotáról, illetve a folyamat fejlesztésen keresztül javítja a szoftvertesztelés hatékonyságát. A folyamatok aktuális helyzetéről készült riportok pedig lehetővé teszik, hogy a projekt résztvevői tisztán lássák a folyamatokat és azok képességeit. Ugyanakkor, ha a felmérések nem teljeskörűek és pontosak, akkor a modellek által nyújtott javaslatok alkalmazása nem lesz hatékony az elért eredmények szempontjából. [21]

Mint már korábban említettem a TMMi és a TPI Next modellek felépítése különböző, mivel a TMMi egy lépcsős modell, mely végrehajtása során a fókusz az adott érettségi szintek teljesítésén van, illetve a TPI Next egy folytonos, PDCA alapú modell, ahol a fókusz az egyes területek folyamatos fejlesztésén van. Ugyanakkor a TMMi-ban az adott érettségi szintek korlátozott számú folyamatterületei miatt az adott területek kiemeltebb figyelmet kapnak, mint a TPI Next modellben, ahol egyszerre vizsgáljuk mind a 16 kulcsterületet és a cél a teljes folyamat áttekintése. [24]

Eltérés van a két modell között használt terminológiában, hiszen a TMMi a standard tesztelési terminológiát használja (amely szélesebb körben ismert), míg a TPI Next a TMap alapú terminológiát. A modellek megközelítése is eltérő, mivel a TPI Next modell üzlet vezérelt és szoftvertesztelői megközelítéseket használ, míg a TMMi magas fókuszt fektet a menedzsment elköteleződésére, így egy olyan szervezetben, ahol nem megfelelő a

menedzsment hozzáállása valószínűleg jobban alkalmazható a TMMi modell a menedzsment hozzáállásának kialakítására. [30] Különbség van a modellek által szolgáltatott információk értékében is. A TPI Next modell eredményeit a modellhez tartozó „Tesztelési érettségi mátrix” összegzi, a mátrix pedig lehetővé teszi a teljes folyamat aktuális állapotának gyors áttekintését, illetve a modellben alkalmazott Igen/Nem-es ellenőrző pontok kiküszöbölik a semleges válaszokat. Ezzel szemben a TMMi területekre bontva mutatja meg az eredményeket, és mivel Igen/Részlegesen/Nem típusú ellenőrzőpontokat tartalmaz, ezért bizonyos helyzetekben kialakulhatnak semleges eredmények, melyek alapján nehezebb döntéseket hozni. [26] A TMMi modell bevezetéséhez szükséges tananyagok ingyenesen elérhetőek a TMMi Foundation honlapján, a TPI Next modellhez tartozó mátrixok pedig ingyenesen letölthetőek a Sogeti honlapjáról, viszont magát a modellt leíró dokumentumokat meg kell vásárolni. Emellett a TMMi modellről tapasztalataim alapján több kutatás és információ érhető el az interneten, mint a TPI Next modellről. Az irodalomkutatás során úgy érzékeltem, hogy Európában sokkal ismertebb a TMMi modell, a TPI Next modell inkább Japánban terjedt el. Végül a TMMi Foundation nem csak tananyagot, de auditokat is biztosít a TMMi tanúsítvány megszerzéséhez, miközben Magyarországon nincs olyan szervezet mely TPI Next tanúsítvány megszerzésének lehetőségét biztosítaná.

Az adott szoftvertesztelési folyamathoz legjobban illeszkedő modell kiválasztása komplex feladat, ezért a Komplex Tulajdonságok Összemérése módszer alkalmazásával számszerűsített végeredmény kapható. A módszer alkalmazása során rögzíteni kell az adott modellel kapcsolatos célokat, majd ezek alapján a szakértői csoport meghatározza a vizsgálandó komplex tulajdonságokat, kritériumokat. A modellek tulajdonságai alapján összeállíthatóak az alapadatok. A kvalitatív szempontok számszerűsítése után kialakíthatóak a mértékegységfüggetlen, azonos irányú adatok. Az transzformált adatok egyszerű súlyozása után összegezzük az adatokat a modellekre vonatkozóan, majd a kapott összegek közül a maximális értékű alternatívát választjuk. [31]

3. Anyag és módszer

3.1. GSGroup AS bemutatása

A GSGroup AS vállalat jelenleg 9 országban, 14 irodával, 195000 aktív felhasználóval és 210 alkalmazottal van jelen a nemzetközi piacon és több mint 30 országban nyújtanak szolgáltatásokat. A vállalatot 1992-ben alapították Norvégiában. A cég portfóliója 2 fő termékcsoportot tartalmaz. A termék csoportok külön-külön, illetve együtt is megvásárolhatóak. Az egyik termékcsoport a jármű flottával rendelkező vállalatok számára nyújt flottakezelést segítő digitális megoldásokat, a másik termékcsoport pedig „terepen" dolgozó vállalatok számára nyújt a vállalat operatív irányítását, a menedzsmentet, illetve a mindennapi munkavégzést segítő digitális szolgáltatásokat „Handyman" alkalmazás néven.

3.2. GSGroup Handyman termékcsomag bemutatása

A GSGroup Handyman rendszer megkönnyíti a terepen dolgozó emberek számára az árajánlatok létrehozását, a vevői megrendelések regisztrálását, a munkaórák és szabadságok ütemezését, dokumentumok tárolását és kezelését, illetve a számlázási folyamatokat. Az alkalmazás használata lehetővé teszi a vállalatok számára a magasabb termelékenységet, csökkenti az adminisztrációra fordított időt, gyorsítja a számlázási folyamatokat, javítja a szolgáltatások minőségét, csökkenti a szervízköltségeket, segít a hatékony és jól dokumentált munkafolyamatok kialakításában.

A Handyman rendszer az alábbiakban bemutatott alkalmazásokból épül fel. A rendszer központja a Handyman Office és mobil alkalmazás, ehhez lehet megvásárolni opcionálisan a többi kiegészítő komponenst.

Handyman Office (asztali alkalmazás)

A Handyman Office alkalmazás az irodai dolgozók munkáját segíti. Az alkalmazás a vállalatok saját szervein kerül telepítésre, így az adatok kezelésében nem vesz részt második fél. Lehetőség van megrendelések létrehozására, a megrendelésekhez erőforrás és alapanyag hozzárendelésére, a logisztikai folyamatok menedzselésére, munkavállalói adatok kezelésére, árajánlatok menedzselésére, raktárakban található alapanyagok nyomon követesére és alapanyag rendelések létrehozására, szerződések kezelésére, szabadságok és erőforrások tervezésére, az ügyfelek adatainak, illetve eszközeinek nyilvántartására. A Handyman Office alkalmazás fejlesztésével a Handyman Office csapat foglalkozik. Funkcionalitás szempontjából a Handyman Office alkalmazás egy egyszerűsített (kevesebb konfigurációs lehetőséget biztosító) változata a Handyman as a Service alkalmazás. Az alkalmazás karbantartásához és telepítéséhez viszont nincs szükség saját üzemeltetőkre és szerverre, mivel az alkalmazás telepítése és futtatása a GSGroup AS saját szerverein történik, illetve az üzemeltetést is a GSGroup AS szakemberei végzik. A Handyman as a Service alkalmazás fejlesztésével többnyire a Handyman Office csapat foglalkozik, de vannak olyan fejlesztések, amikben a Handyman Web csapat fejlesztői is közreműködnek.

Handyman Mobile (mobilalkalmazás)

A mobilalkalmazás a telefonos áruházakból egyszerűen telepíthető minden iOS, Android, illetve Windows operációs rendszert futtató okostelefonra. A mobilalkalmazás a szinkronizációs funkció miatt folyamatos kapcsolatban van a korábban bemutatott Handyman Office alkalmazással. A mobilalkalmazás lehetővé teszi a helyszínen szolgáltatást nyújtó szakemberek számára, hogy egyszerűen hozhassanak létre megrendeléseket, ütemezhessék az adott feladatokat, árajánlatok adjanak, szerződéseket kössenek meg, számlát állítsanak ki, lekérdezzék az adott eszközökön korábban elvégzett javításokat és regisztrálják a jelenlegi javítás során mért értékeket, az elvégzett tevékenységeket, illetve az anyagok és munkaórák nyilvántartását. A Handyman Mobil alkalmazás fejlesztésével a Handyman Mobile csapat foglalkozik.

Handyman Connect és Web Office (webes alkalmazás)

A Handyman Connect és Web Office a Handyman Office alkalmazáshoz kapcsolódik a Handyman API[13] kommunikációs csatornán keresztül. A Handyman Connect lehetőséget ad a Handyman Connectet megvásárló ügyfeleknek, hogy az ügyfeleik a webes felületen keresztül megrendeléseket hozzanak létre, illetve nyomon követhessék a már korábban létrehozott rendeléseik állapotát, elfogadják, vagy elutasítsák az árajánlatokat, megtekintsék a rendelésekhez tartozó szerződések adatait, illetve letöltsék az adott szervízelések során elkészült a szakemberek által készített szervíz riportokat. A Web Office alkalmazás tartalmazza a webalapú új funkciókat, például eszközök tömeges importálását és a bizonyos eseményekhez kapcsolódó email-ek konfigurálását. A Handyman Connect és Web Office alkalmazások fejlesztésével a Handyman Web csapat foglalkozik.

3.3. Adatgyűjtési technikák

3.3.1. Adatok forrása

A Handyman részlegen a munkaórák regisztrálása, az ügyfél hibák, a tesztelő csapat által felvett hibák, az új funkciók és hozzájuk tartozó egyéb adatok kezelése az Azure DevOps rendszerben történik. Ez jelentősen megkönnyíti az adatgyűjtést, mivel minden munkavégzéshez kapcsolódó feladatok, a feladatokkal eltöltött idő, a hibák adatai elérhetőek és lekérdezhetőek a rendszerben.

[13] API: Az angol „Application Programming Interface” szóból ered, magyarul “alkalmazásprogramozási felület”. “Az API-k hozzáférést biztosítanak egy adott szoftver vagy eszköz utasításkészletéhez. Az API segítségével egy program anélkül használhatja egy másik program funkcióit, hogy ismernie kellene annak belső működését” Forrás: https://lexiq.hu/api

A Handyman rendszer gyengeségeinek és sérülékenységeinek vizsgálata során a megfelelő értékelés kialakításához fontosnak tartottam a különböző csapatokban dolgozó fejlesztők véleményének összegyűjtését online interjúk formájában a „Szoftverek sérülékenységei ” fejezetben bemutatott 25 darab leggyakoribb és legkritikusabb szoftver gyengeségekkel és sérülékenységekkel kapcsolatban. A felmérés célja, hogy megalapozza az olyan intézkedések bevezetését a Handyman részlegen, amelyek csökkentik a szoftversérülékenységből adódó minőségügyi kockázatokat.

A folyamatelemzéseket a saját tapasztalataim alapján végeztem el, viszont mivel fontosnak tartom a folyamatok teljeskörű és pontos megértését a folyamat fejlesztés elkezdése előtt, ezért, ha egy bizonyos folyamatrészre nem volt megfelelő rálátásom, online interjút készítettem olyan kollégákkal, akik megfelelő rálátással rendelkeznek az adott folyamat részekre.

3.3.2. Adatok feldolgozása

Az előző fejezetben említett Azure DevOps alkalmazás „Lekérdezések” funkciója lehetővé teszi a rendszerben tárolt adatok szűrését az elemekhez tartozó paraméterek alapján. A kapott lekérdezések menthetőek az Azure DevOps rendszerben, illetve a lekérdezett adatokból készíthetőek egyszerűbb kimutatások és diagramok. A diagram készítő funkció lehetővé teszi kör-, oszlop-, halmozott oszlop-, halmozott terület-, terület- és vonal diagram készítését. Ugyanakkor az adatok ábrázolására csak korlátozottan képes az Azure DevOps Diagram készítő funkciója, ezért abban az esetben, ha bonyolultabb adathalmazt szeretnék megjeleníteni, akkor kiexportálom az adatokat .csv formátumba és a Microsoft Excel alkalmazás segítségével készítek diagramokat.

A Handyman rendszer gyengeségeinek és sérülékenységeinek vizsgálata során létrehoztam egy kérdéslistát a 25 leggyakoribb és legveszélyesebb sérülékenységre megjelenésére vonatkozóan, majd a fejlesztőktől kapott részletes válaszokat a Microsoft Excel alkalmazásban

rögzítettem. Az összes interjú elkészítése után Excelben összesítettem a kapott eredményeket termék modulokra vetítve. Vagyis abban az esetben, ha az adott sérülékenységet azonosított a termékben egy interjúban résztvevő személy, akkor az adott termékmodulra vonatkozóan a sérülékenység állapota „valós kockázatot jelent” státuszba kerül. Miután elkészült a termék modulokra bontott lista, összesítettem, hogy a vizsgált 25 darab sérülékenységből hány darab jelenik meg a teljes termékre vonatkozóan.

3.4. Alkalmazott módszerek

A jelenlegi és szoftverfejlesztési és szoftvertesztelési folyamat bemutatása során folyamatábrát alkalmaztam, mivel ez a módszer segíti a folyamat megértését és elemzését, illetve egy jó folyamatábra alapot nyújt a folyamat fejlesztéséhez. A szoftvertesztelési folyamat fejlesztés során a gyakorlatban a két legelterjedtebb modell alkalmazását mérlegeltem. Mivel számos szempontot kell figyelembe venni a modell kiválasztása során, ezért a referencia modellek komplex tulajdonságainak összemérésével hoztam döntést. Azért ezt a módszert választottam, mivel a tulajdonságok összegyűjtése rákényszerít a fontos szempontok összegyűjtésére és azok rangsorolására a súlyozás kialakításánál. Végül a szempontok kiértékelése után kialakul egy szubjektív, konkrét végeredmény arról, hogy melyik referencia modell alkalmazása a jobb.

Az aktuális szoftvertesztelési folyamat érettségi szint felmérése során a TMMi Foundation által kifejlesztett TMMi Lightning Scanning Tool-t használtam, mely egy folyamatauditálási eszköz. Az eszközben található folyamatterületekre vonatkozó kérdéslista kitöltése után eszköz visszajelzést ad arról, hogy a szervezetben található folyamat jelenleg milyen érettségi szinten helyezkedik el a TMMi referencia modellhez képest és az eredmények alapján azonosíthatóak azok a folyamatterületek, melyek fejlesztésre szorulnak az adott folyamat érettségi szint teljesítéséhez.

4. Eredmények

4.1. Handyman termékcsomag szoftvergyengeségeiből és sérülékenységeiből adódó kockázatok felmérése

Ebben a fejezetben a célom feltárni a szoftversérülékenységek kezelésének aktuális helyzetét a GSGroup Handyman részlegen, majd egy a vállaltnál dolgozó fejlesztők körében készített felmérés segítségével megvizsgálni azt, hogy a Handyman termék milyen szinten veszélyeztetett az adott sérülékenységekkel szemben. A felmérések eredményei alapján pedig felvázolom azokat az intézkedéseket, melyek bevezetésével kiszűrhetőek az ilyen típusú termékhibák.

4.1.1. Szoftversérülékenységek kezelésésnek aktuális helyzete

A GSGroup Handyman részlegen jelenleg nem fordítanak kellő figyelmet az a szoftverek sérülékenységeiből adódó minőségre ható kockázatok csökkentésére. A Handyman részleg nem rendelkezik IT biztonsági csapattal, a vállalat nem kér fel külső szakértőket termék kiadás előtt az alkalmazáscsomag sérülékenységeinek feltárására, illetve a fejlesztéssel és teszteléssel foglalkozó kollégák sem végeznek ilyen típusú vizsgálatokat, illetve az alkalmazottak ilyen irányú továbbképzésére sem fordítanak erőforrásokat. Tehát a Handyman termékcsomag esetében magas a kockázata annak, hogy olyan sérülékenységeket tartalmaz a szoftver, amelyet egy esetleges kibertámadás során kitudnak használni a támadók. Ennek következményeként pedig publikussá válhatnak az ügyfeleink személyes adatai, a rendszer elérhetetlenné válhat, vagy a rendszerben tárolt adatok törlődhetnek. Egy ilyen esemény jelentősen rontaná a jelenlegi és a jövőbeli ügyfelek termékkel/céggel kapcsolatos véleményét, illetve bizonyos esetekben akár magas kártérítési büntetést is vonhat maga után. Eddig semmilyen belső felmérés nem készült arról, hogy milyen problémák lehetnek a termékben,

illetve a szervezeti kultúrában sem jelenik meg az, hogy kellőképpen odafigyeljenek az ilyen típusú kockázattokra.

4.1.2. Szoftversérülékenységek megjelenésének felmérése a Handyman termékre vonatkozóan

A szoftvergyengeségekből és sérülékenységekből adódó minőségi kockázatok felmérésére személyes interjúk formájában felmérést végeztem a különböző csapatokban dolgozó szoftverfejlesztéssel foglalkozó kollégák körében. A felmérés célja, hogy rámutasson arra, hogy a termékben milyen mennyiségben jelennek meg bizonyos sérülékenységek illetve, hogy igazolja azt, hogy szükség van olyan intézkedésekre, amelyek csökkentik az IT biztonsággal kapcsolatos kockázatokat. A felmérésbe azért nem vontam a szoftvertesztelés sel foglalkozó személyeket, mert a tesztelő csapatban dolgozó személyek nem rendelkeznek megfelelő rálátással az ilyen típusú problémákra, mivel magasabb szinteken végeznek tesztelést, másrészt a csapat tagjai nem rendelkeznek olyan típusú szaktudással, mely lehetővé tenné az informatika biztonsági problémák észlelését. Ezzel szemben a szoftverfejlesztők rálátnak az adott modulok teljes kódbázisára és a mélyebb szinteken kialakított architektúrális megoldásokra és a megadott szempontok alapján képesek felmérni az adott modulban lévő biztonsági gyengeségeket.

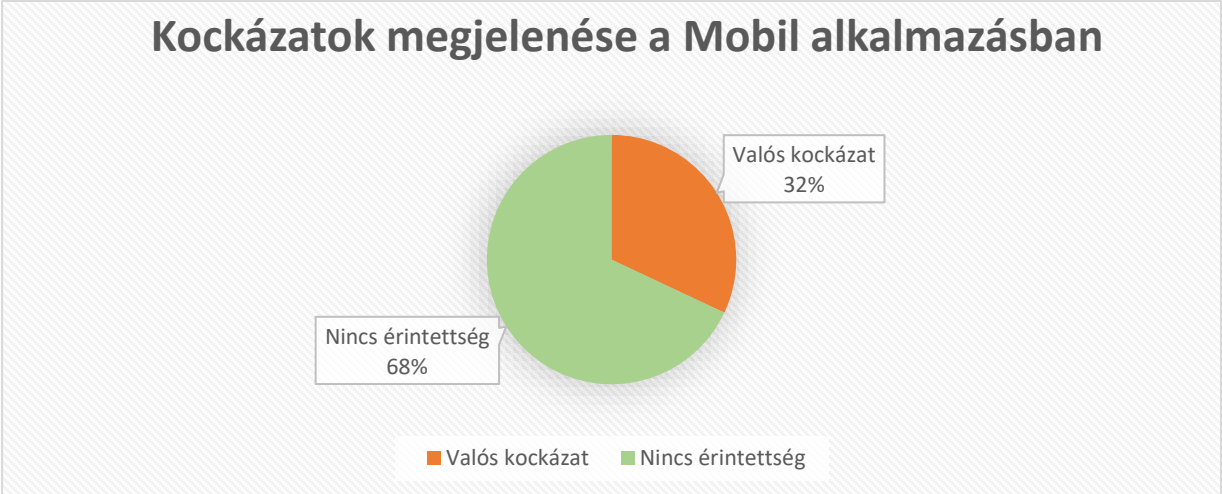
A felmérésben résztvevő személyeket úgy választottam ki, hogy a személyek tudása teljes lefedettséget adjon a Handyman termékcsomag rész moduljaival kapcsolatban. Emellett véleményem szerint minél több ideje dolgozik egy adott személy a vállaltnál, annál komplexebben képes átlátni a rendszer működését és annak gyengeségeit, ezért az adott csapatokból olyan személyeket választottam ki, akik magasabb szintű szakmai tapasztalattal rendelkeznek és régebb óta dolgoznak a GSGroup-nál.

A felmérésben résztvevő személyek rálátása az adott termék modulokra és szakmai tapasztalatuk:

S1. Handyman mobile alkalmazás fejlesztésében résztvevő személy, 3-5 év tapasztalat
S2. Web alapú Handyman alkalmazás modulok, illetve API fejlesztéssel foglalkozó személy 5-7 év tapasztalat
S3. Handyman Office alkalmazás fejlesztésével foglalkozó személy 5-7 év tapasztalat
S4. Kutatás és fejlesztési csapat vezetője – teljes termékcsaládra rálátással rendelkezik 7-9 év tapasztalat

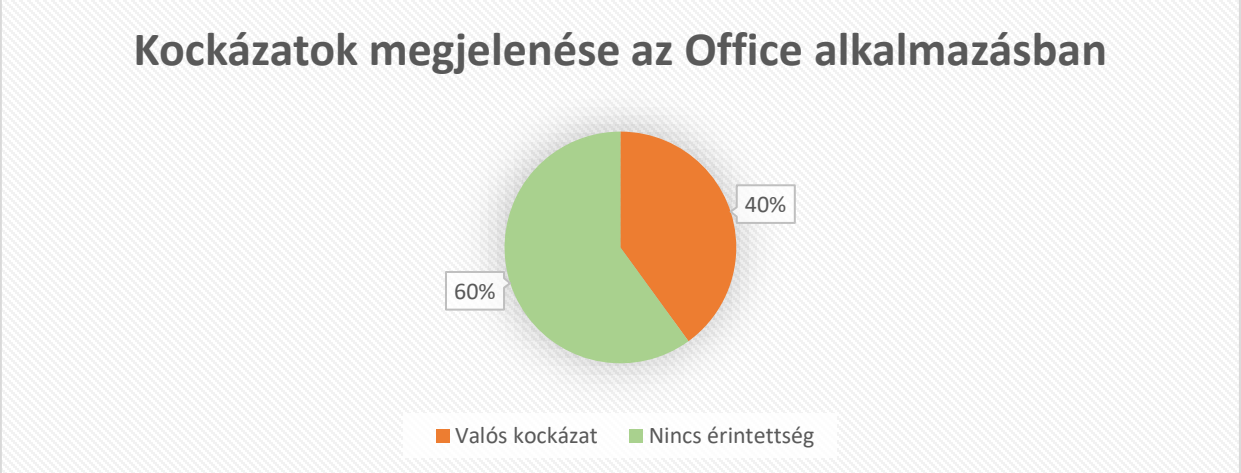
A felmérés során a „Szoftverek sérülékenységei ” fejezetben felsorolt 25 leggyakoribb és legveszélyesebb szoftvergyengeség és szoftversérülékenység megjelenését vizsgáltam az adott részmodulokkal kapcsolatban. A felmérés során az interjúalanyok sok esetben adtak visszajelzést a másik csapathoz tartozó termék modulokkal kapcsolatban is. Ezért a felmérések eredményeit először összesítettem az alkalmazásokra bontva.

A felmérés során a Handyman Mobile alkalmazásra kapott eredményeket az 5. ábra mutatja. A felmérésben felsorolt szoftversérülékenységek és szoftvergyengeségek 32%-a megjelenik a Handyman Mobil alkalmazásban.



5. ábra: Felmérésben felsorolt 25 darab sérülékenységből 8 darab megjelenik a Handyman Mobile alkalmazásban. Saját készítésű ábra, saját kutatás alapján

A felmérés során a Handyman Office alkalmazásra kapott eredményeket a 6. ábra mutatja. A felmérésben felsorolt szoftversérülékenységek és szoftvergyengeségek 40%-a megjelenik a Handyman Office alkalmazásban.



6. ábra: Felmérésben felsorolt 25 darab sérülékenységből 10 darab megjelenik a Handyman Office alkalmazásban. Saját készítésű ábra, saját kutatás alapján

A felmérés során a Handyman termékcsomag webes alkalmazásaira kapott eredményeket a 7. ábra mutatja. A felmérésben felsorolt szoftversérülékenységek és szoftvergyengeségek 44%-a megjelenik a Handyman termékcsomag webes alkalmazásaiban.

7. ábra: Felmérésben felsorolt 25 darab sérülékenységből 11 darab megjelenik a Handyman Office alkalmazásban. Saját készítésű ábra, saját kutatás alapján

Miután elkészültek az egyes részalkalmazásokra vonatkozó kimutatások, összesítettem az adatokat a teljes Handyman termékre vonatkozóan. Abban az esetben, ha az adott sérülékenység megjelenik az egyik részalkalmazásban, akkor az érinti a Handyman alkalmazást is. A Handyman alkalmazásra kapott eredményeket a 8. ábra mutatja. A felmérésben felsorolt szoftversérülékenységek és szoftvergyengeségek 64%-a megjelenik a Handyman termékcsomagban.

8. ábra: Felmérésben felsorolt 25 darab sérülékenységből 16 darab megjelenik a Handyman alkalmazásban. Saját készítésű ábra, saját kutatás alapján

4.1.3. Javaslatok a termékben található szoftversérülékenységek számának csökkentésére

A felmérés eredményei alapján egyértelműen látszik, hogy a Handyman termékcsomag erősen érintett a leggyakoribb szoftver sérülékenységekkel kapcsolatban, ami jelentős kockázatot jelent a termék minőségére nézve. Ennek oka, hogy az ilyen típusú kockázatokra a szervezeten belül nem fordítanak kellő figyelmet, illetve minőségbiztosítási csapat tagjai nem rendelkeznek a megfelelő szintű tudással az ilyen típusú hibák kiszűrésére. Ahhoz, hogy ezeket a kockázatokat mérsékelni lehessen a Handyman termék fejlesztésére vonatkozóan intézkedéseket kell bevezetni.

A szoftverben található sérülékenységek kiszűrésére a termék kiadás előtt három lehetősége van a vállalatnak:

1. Szoftvertesztelői csapat továbbképzése, melynek eredményeként a tesztelési folyamat képes lesz az ilyen típusú hibák detektálására
2. IT biztonsági szakértő felvétele az alkalmazottként
3. Külső szakértői csapat felkérése a termék ellenőrzésére

Szoftversérülékenységek kiszűrésére vonatkozó megoldási változatok vizsgálata szakmai szempontból

A **szoftvertesztelői csapat továbbképzése** során a tesztelő csapat minden tagjának részt kellene vennie egy informatika biztonsági tesztelő képzésen. Magyarországon a Masterfield Oktatóközpont a Szegedi Tudományegyetemmel együttműködve tart IT biztonsági tesztelő képzést, melynek keretein belül a csapat tagjai megismerhetik a biztonsági tesztelés módszertanát, tervezését, integrálását, illetve megtanulhatják, hogy hogyan kell alkalmazni a témával kapcsolatos technikákat és hogyan kell kezelni a kockázatokat. Ugyan a tanfolyamhoz kapcsolódó ISTQB vizsga letételéhez 3 éves biztonsági tesztelői tapasztalat szükséges, de a mi esetünkben a cél nem a tanúsítvány megszerzése, hanem a csapat tagok felruházása olyan tudással, amely lehetővé teszi a biztonsági hibák detektálását. A képzés díja nettó 229 500 Ft/fő. A megoldás előnye, hogy a csapat mély rendszer ismerettel rendelkezik, illetve benne van a mindennapi fejlesztési folyamatokban, ezért közvetlenül a hiba detektálás után a fejlesztő csapat elkezdhetné javítani azokat. Emellett a fejlesztői csapattal való közvetlen kapcsolat miatt „fehér doboz” tesztelést lehetne végezni, melynek lényege, hogy a csapat teljes mértékben rálát az adott megoldásokra és nem csak az bementekre/interakcióra kapott kimeneteket látja. A megoldás hátránya, hogy egy továbbképzett szoftvertesztelő nem lesz képes olyan mély IT biztonsági tesztelést végezni, mint egy IT biztonsági szakértő, ezért a termékben benne maradhatnak bizonyos sérülékenységek.

Tehát 1-5-ös skálán vizsgálva:

- Sérülékenység detektálási hatékonyság: 3
- Detektált sérülékenységek javításának megvalósulása: 5

Az **IT biztonsági tapasztalattal rendelkező alkalmazott** felvételének az előnye, hogy egy ilyen típusú szaktudással rendelkező személy nagyobb tudással fog rendelkezni, mint egy továbbképzett szoftvertesztelő, illetve benne lenne a mindennapi fejlesztési folyamatokban, ezért közvetlenül a hiba detektálás után a fejlesztő csapat elkezdhetné javítani a detektált hibákat. Hátránya, hogy mivel a GSGroup-ban nincs HR részleg, ezért a szoftvertesztelői csapat képzéséhez képest szükség van egy 1-2 hónapos kiválasztási folyamatra, illetve mivel a szakmai állásinterjút a fejlesztő csapat tartja és nincs kifejezetten IT biztonsági szakember a csapatban, ezért a kiválasztás során nem biztos, hogy megfelelően sikerülne felmérni a jelentkezők tudását.

Tehát 1-5-ös skálán vizsgálva:

- Sérülékenységek detektálási hatékonyság: 4
- Detektált sérülékenységek javításának megvalósulása: 5

A **külső szakértői csapat felkérés**ének az előnye, hogy az előző opciókhoz képest, a vizsgálatban résztvevő személyek itt rendelkeznek a legmagasabb szakmai tudással. Ezért minden olyan sérülékenységet észlelni fognak, ami érintheti a rendszert. Hátránya, hogy szakértői csapat nem a fejlesztői csapat része, ezért a korábbi tapasztalatok alapján lehet, hogy a fejlesztő csapat figyelmen kívül fogja a hagyni a detektált hibákat. Emellett az eredmények egy csomagban fognak megérkezni, vagyis nincs lehetőség arra, hogy a hibák javítását a fejlesztő csapat a detektálás után azonnal elkezdje.

Tehát 1-5-ös skálán vizsgálva:

- Sérülékenységek detektálási hatékonyság: 5
- Detektált sérülékenységek javításának megvalósulása: 3

Szoftversérülékenységek kiszűrésére vonatkozó megoldási változatok vizsgálata pénzügyi szempontból

A **szoftvertesztelői csapat továbbképzés**ének költségei a jelenlegi 5 fős szoftvertesztelői csapattal kalkulálva 1 éves időtartamra vizsgálva:

- A 3 napos Masterfield képzés ára (egyszeri költség): nettó 229 500 Ft/fő > **5 főre: nettó 1 147 500 Ft**

Az **IT biztonsági tapasztalattal rendelkező alkalmazott**at felvételének költségei: Egy IT biztonsági szakértő keresete ma Magyarországon br. 500 000 – 1 500 000 Ft/hó között mozog. A számítások során a középértékkel br. 1 000 000 Ft/hó fogok számolni (körülbelül itt helyezkedik el az 5 év tapasztalattal rendelkező szakértők keresete). Ezek alapján a munkáltatói szociális hozzájárulási adót is beleszámolva, a munkáltató költsége havonta: 1 130 000 Ft.

- Külső HR szolgáltatások igénybevétele (1 havi alkalmazotti munkabérrel egyezik meg, egyszeri költség): nettó **1 000 000 Ft**
- Munkavállaló bérköltsége (folytonos költség): 1 130 000 Ft/hó - 1 évre számolva: **13 560 000 Ft**

A **külső szakértői csapat felkérésének** költségei:

1. Egy havi 64 munkaórás tanácsadás, heti összegző meetingekkel, negyedéves automatikus és manuális sérülékenységvizsgálattal (Magyarországon): nettó 1 199 999 Ft/hó - 1 évre számolva: **14 399 988**

4.2. Szoftvertesztelési folyamat elemzése, fejlesztendő területek meghatározása

4.2.1. Aktuális fejlesztési és tesztelési folyamatok elemzése és szemléltetése folyamatábrával

A folyamatok ábrázolása folyamatábra segítségével lehetővé teszi a jelenlegi folyamat megértését, ezáltal fontos kiinduló pontként szolgál a folyamatfejlesztési projektekben. A diplomadolgozatomban először a teljes fejlesztési folyamatot fogom megvizsgálni a Handyman részlegre vonatkozóan, majd külön a szoftvertesztelési folyamatot.

Fejlesztési folyamat elemzése a Handyman részlegen

A fejlesztési folyamat vizsgálata során a kiinduló pont a fejlesztési igény kialakulása befejezési pont pedig az adott fejlesztés átadása az ügyfeleknek. Az elemzés vizsgálata során a célom olyan problémák azonosítása, melyek ronthatják a folyamat hatékonyságát, illetve negatív hatást gyakorolnak a termék minőségre.

A termék **fejlesztési igények** több irányból érkezhetnek: ügyfelek termék használat során felmerült problémai/hibák és termék fejlesztési igények, fejlesztő csapatban felmerült termék javítási ötletek (ezek általában alacsony számban vannak figyelembe véve), illetve vállalati stratégiához kapcsolódó termék fejlesztési igények. Az előbbiekben felsorolt tényezők a fejlesztési folyamat bemenetei.

A Terméktulajdonos az ügyfél igények beérkezését követően megvizsgálja, hogy annak megvalósítása mennyivel növelné a termék értékét, megvalósítható-e az ötlet (ezt a kutatási és

fejlesztési részleg vezetőjével egyezteti, aki pedig szükség esetén egyeztet a szoftver fejlesztőkkel), illetve megvizsgálja, hogy milyen a hozzáadott érték és fejlesztési költségek viszonya. Abban az esetben, ha a Terméktulajdonos úgy dönt, hogy megvalósításra kerül a fejlesztési igény, az beérkező visszajelzésekből és a megvalósíthatósági adatok alapján megfogalmazza a **magas szintű követelmények**et az új termék funkcióhoz kapcsolódóan. A magas szintű követelmények alapján a termék tervező elkészíti a specifikációt. A specifikáció alapján elkészülnek a megvalósítandó új funkciókhoz tartozó **fejlesztői feladatok**, melyeket a termék backlog tartalmaz.

Tevékenységben azonosított problémák:

P01 A specifikáció alapvetően azt a célt szolgálná, hogy leírja, hogy a magas szinten megfogalmazott igényekből kiindulva az adott funkciónak pontosan hogyan kéne működnie. A gyakorlatban viszont gyakoriak az olyan esetek, amikor az elvárások hiányosan kerülnek megfogalmazásra, vagy a termék specifikáció nem kerül frissítésre bizonyos módosítások után. Ennek következményeként a specifikáció sok esetben elavult és hiányos, ezért szoftvertesztelőként nehéz eldönteni, hogy pontosan mi az elvárt működés.

A GSGroup Handyman részlegén a Sprintek 2 hét hosszúságúak. A Sprintek a **Sprint tervezési megbeszéléssel** kezdődnek, mely során először összegyűjtik, hogy a fejlesztők és tesztelők terveznek-e szabadnapot kivenni a következő sprint időtartama alatt, majd a fejlesztőkhöz napi 6 órás munkavégzést rendelnek hozzá, a szoftvertesztelőkhöz pedig 0 órát. Ezek a számok ahhoz szükségesek, hogy a Scrum Master lássa, hogy mennyi erőforrás lesz elérhető a következő két hétben munkaórákra bontva. A szoftvertesztelőkhöz azért rendelnek 0 órát, mert a tesztelői erőforrásokkal nem terveznek, illetve a fejlesztések számlázásakor is csak úgy veszik figyelembe a tesztelői munkákat, hogy a fejlesztői költségek 30%-át hozzáadják az alap fejlesztési költségekhez, viszont ez tartalmazza a talált hibákra fordított fejlesztői költségeket is. Miután megtervezésre kerültek az elérhető erőforrások, a fejlesztők elkezdik felvenni a feladatokat úgy, hogy azok 90%-ban kitöltsék a munkaidejüket (minden feladatnak megvan a becsült ideje), a többi 10% pedig tartalék idő az esetlegesen megjelenő sürgős

problémák megoldására. Az előző sprintekben megvalósított új funkciókat és javított hibákat a szoftvertesztelőkhöz rendeli a Scrum Master (ezekhez a feladatokhoz nem tartozik időbecslés).

Tevékenységben azonosított problémák:

P02 A Scrum módszertanban a tesztelők és a tervezők is fejlesztőnek számítanak, viszont a cégnél alkalmazott módszertanban meg vannak különböztetve a szerepkörök. A szerepkörök megkülönböztetéséből adódik, hogy csak a fejlesztők munkaórái és tevékenysége kerül megtervezésre a Sprint tervezési megbeszéléseken, a szoftvertesztelőké és a termék tervezőé nem. Ebből adódhatnak határidő csúszások a tesztelői/tervezői erőforrások esetleges túlterheltsége miatt.

P03 A számlázásnál figyelembe vett alap fejlesztési idő 30%-a sok esetben nem fedi a valóságot. A gyakorlatban miközben egy szoftverfejlesztő a programkódban 5 sort módosít, addig a szoftvertesztelőknek fel kell tárniuk az összes kapcsolódó beállítást és funkciót stb. és meg kell vizsgálni azt, hogy az adott kombinációkkal is megfelelően működik a szoftver. Ennek következtében a tervezett költségeket jelentősen meghaladhatják a valós költségek.

P04 A Sprint tervezése során létrejön egy dokumentum mely az adott Sprint céljait tartalmazza. A Handyman Mobile csapatban a Sprint tervezése során nem jön létre olyan dokumentum, mely leírná az adott Sprint célját.

A **2 hetes Sprint** során a fejlesztő csapat a Sprint tervezésen felvett feladatokkal foglalkozik. A Sprint során mindennap 15 perces napi megbeszélések vannak, ahol a csapat résztvevői elmondják, hogy mit csináltak előző nap, mi a tervük a mai napra, illetve van-e olyan tényező, ami blokkolja a munkájukat. A folyamatok menedzsmentje és adminisztrálása az Azure DevOps rendszerben történik, ami sok szempontból előnyös. Például automatikusan mindennapra legenerálódik a leégési diagram, mely célja, hogy vizuálisan bemutassa a csapat előrehaladását, és segítsen azonosítani a problémákat, amelyek akadályozhatják a csapatot abban, hogy időben befejezze az adott sprintre meghatározott feladatcsomagot.

A Sprint utolsó napján kerül sor a **Sprint áttekintő megbeszélés**re, ahol a termék tulajdonosoknak és érintetteknek bemutatásra kerülnek a megvalósított funkciók, illetve a résztvevők feltehetik a megvalósítással kapcsolatban felmerült kérdéseiket. A megbeszélés során a Scrum Master megállapítja, hogy melyik Backlog tétel van „Kész” állapotban és melyik nem.

Tevékenységben azonosított problémák:

P05 A Handyman Mobile csapatban nem történik meg a Sprint áttekintő megbeszélés a Sprintek végén. Ennek következményeként a Terméktulajdonos nem látja, hogy melyik Backlog tétel van „Kész” állapotban, illetve az érintettek nem tehetik fel az adott funkciókkal kapcsolatos kérdéseiket és visszajelzéseiket.

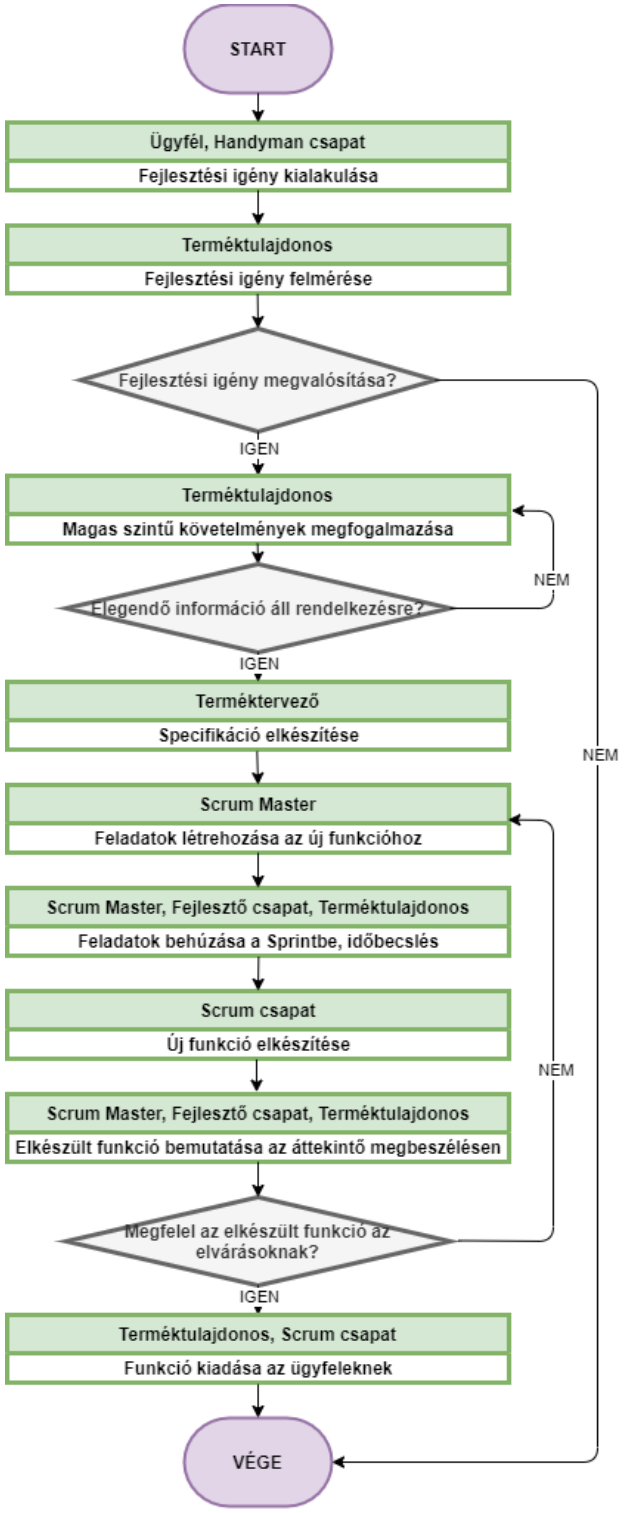
Az áttekintő megbeszélés után közvetlenül jön a **Sprint visszatekintő megbeszélés** (retrospektív), ahol a fejlesztő csapat tagjai elmondják, hogy mik mentek jól az adott sprint során, milyen váratlan feladatok merültek fel, illetve mik azok a tényezők, amiket fejleszteni kellene.

Tevékenységben azonosított problémák:

P06 A „fejlesztendő tényezők” sok hasznos információt tartalmaznak a menedzsment részére, viszont a jelenlegi tapasztalatok alapján ezek az információk a nem lesznek felhasználva a későbbi döntések során.

Jelentős eltérés a Scrum módszertantól, hogy nem minden sprint végén kerül szállításra a termék, hanem általánosan 4-5 havonta van termék kiadás, ahol összefésülik az elmúlt időszakban megvalósított funkciókat, majd átadják az ügyfeleknek. Összességében megállapítható, hogy a Handyman részlegen a Scrum módszertan lényeges eseményei kimaradnak, a módszertan nem teljeskörűen van implementálva, ennek pedig olyan következményei vannak, mint például az információk nem megfelelő áramlása.

A előzőekben bemutatott Scrum módszertanban az új funkciók létrehozásának folyamatát a 9. ábra mutatja be.



9. ábra: Új funkció kifejlesztésének folyamata a Handyman részlegen. Saját készítésű ábra, saját kutatás alapján

Szoftvertesztelési folyamat elemzése a Handyman részlegen

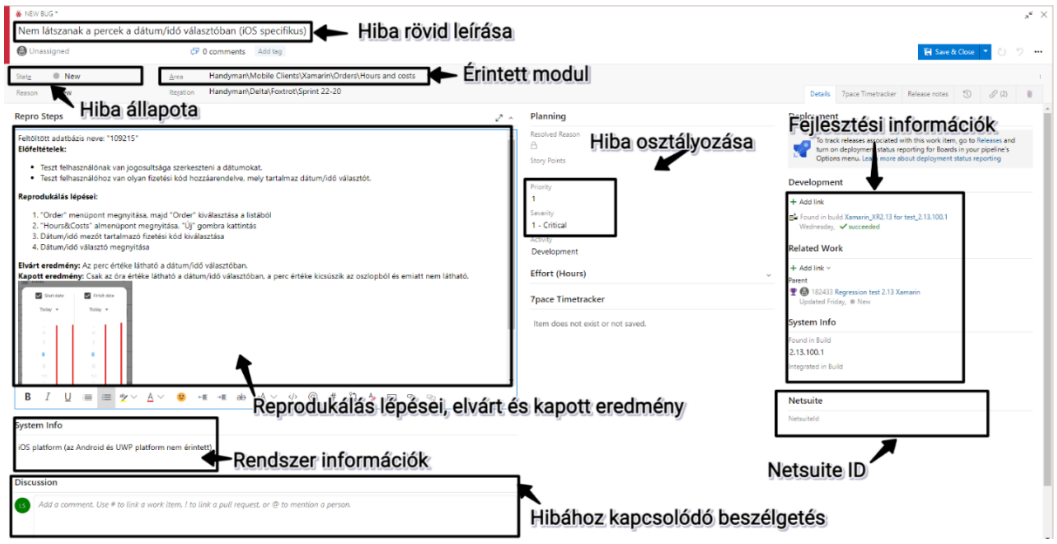
Az egyes fejlesztendő funkciók tartalmát a specifikáció határozza meg. A szoftverfejlesztők a specifikáció alapján lefejlesztik az adott funkciót, ekkor a funkció „Megoldott” állapotba kerül. Az lefejlesztett új funkció átkerül ahhoz a tesztelőhöz, aki az adott Scrum csapathoz tartozik. A szoftvertesztelők megvizsgálják, hogy a fejlesztett funkcionalitás megfelel-e a specifikációnak, illetve azt, hogy vannak-e hibák az adott funkcióban. Abban az esetben, ha a fejlesztett funkcióban nem találnak hibát a tesztelés során, akkor a funkció állapota „Lezárt” lesz.

Tevékenységben azonosított problémák:

P07 Mint korábban említettem a specifikáció sok esetben nem teljeskörű, illetve gyakorlatban olyan is előfordult, hogy logikai hiba volt benne. Ha ezt a fejlesztők nem veszik észre és a tesztelés során derül ki, akkor visszakell menni a folyamat legelejére, ahol tervező javítja a specifikációban található hibákat, ezután a fejlesztők újra fejlesztik az adott funkciót, végül a tesztelők újra tesztelik.

P08 A tesztelők tapasztalat alapú tesztelést végeznek, nincs egy egységesen kialakított munkavégzési eljárás. Ennek következtében jelentős eltérések vannak a különböző tesztelők által végzett munka mélységében és minőségében.

Ha a fejlesztők által megvalósított funkció nem felel meg az elvárásoknak, a specifikációnak vagy hibák vannak benne, akkor a tesztelők létrehoznak egy hibajegyet az Azure DevOps rendszerben. A hibajegy felépítését a 10. ábra mutatja be egy példán keresztül.



10. ábra: Hibajegy felépítése az Azure DevOps rendszerben. Saját készítésű ábra, saját munka alapján

A hibajegyet a Sprint tervezése során fejlesztőkhöz rendelik, majd az adott Sprint során megkezdődik a hibajavítás. Amikor a fejlesztő elkezd dolgozni a hiba javításán a hiba állapota „Aktív” -ra változik. A hibajavítás publikálása után a hiba állapota „Megoldott” -ra változik, ezután hozzárendelésre kerül általában ahhoz a tesztelőhöz, aki a hibajegyet létrehozta és ekkor elindul az újra tesztelés. Újra tesztelésre azért van szükség, mert sok esetben a hiba nem kerül megfelelően javításra, vagy a hibajavítás problémát okoz az alkalmazás kapcsolódó részeiben. Ha az újra tesztelés során a tesztelők megállapítják, hogy a javítás sikeres volt, a hiba állapota „Lezárt” lesz. Ha az új funkcióhoz kapcsolódó minden hiba javításra került, akkor a funkció lezárásra kerül.

Tevékenységben azonosított problémák:

P09 Gyakorlatban a hibák javításának újra tesztelése során nem történik meg az alkalmazás érintett részeinek átvizsgálása. Ebben az esetben magas a kockázata, hogy az esetlegesen a hibajavítás által okozott új hiba nem kerül detektálásra a termék kiadás előtt és benne marad az ügyfeleknek átadott verzióba. Ennek oka, hogy a tesztelők körében nincs egy általánosan meghatározott munkavégzési eljárás, mely tartalmazza az érintett részek felülvizsgálatát a hibák újra tesztelése közben.

A 2-4 hónap alatt megvalósított új funkciók „összefésülésre” kerülnek egy közös kiadási verzióba. A kiadási verziót a tesztelők a regressziós teszt keretében megvizsgálják, melynek célja, hogy megállapítsák, hogy az alkalmazás alapfunkciói megfelelően működnek-e, az összefésült új fejlesztések nem okoznak-e egyéb hibákat, illetve megfelelő-e az új és a régi funkciók együttes működése. A regressziós tesztek és a funkció tesztek során is manuális tesztelés történik, bár elindult egy kezdeményezés a regressziós tesztelés automatizálásra a web csapatban, mely sikeres kivitelezés esetén jelentősen hozzájárulhat a regresszió során felhasznált humán erőforrások csökkentéséhez. A regressziós tesztek során felfedett hibák az előzőekben bemutatott hibakezelési folyamaton mennek keresztül. Ha minden adott verzióban javítandó hiba lezárt állapotba kerül megtörténik az új funkciók átadása az ügyfeleknek.

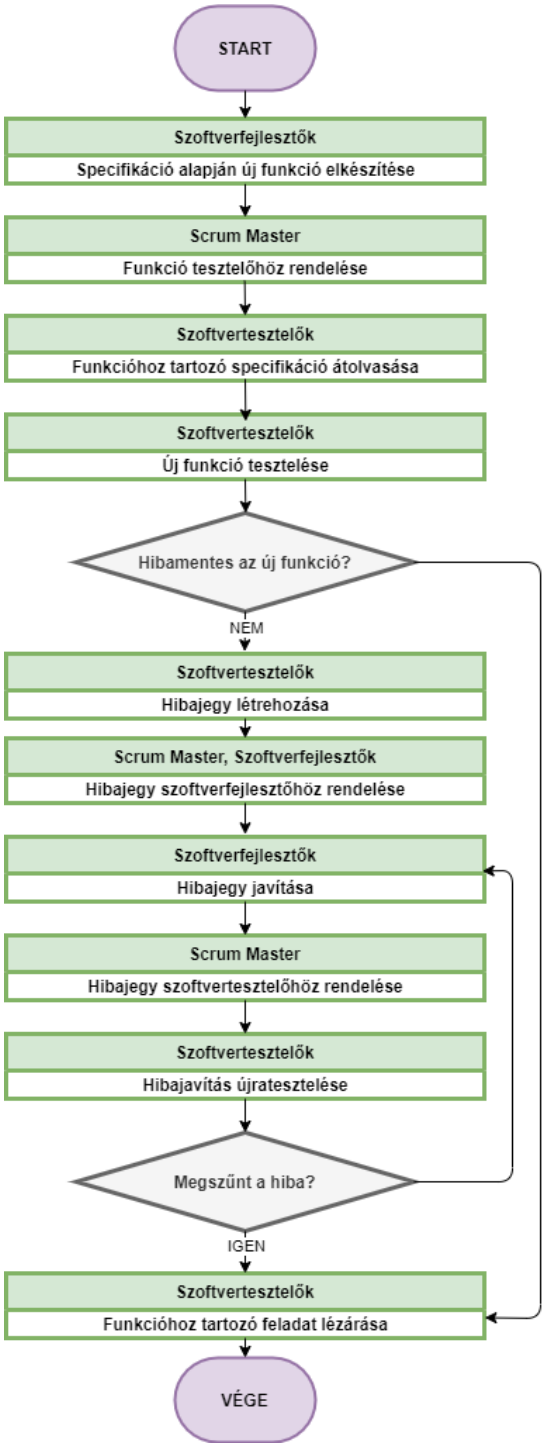
Tevékenységben azonosított problémák:

P10 A regressziós tesztek tervezése során nem áll rendelkezésre egy olyan kimutatás, ami alapján erőforrás hatékonyan meghatározható lenne az adott modulokra alkalmazandó tesztelési mélység. Ennek következményeként a regressziós tesztek elvégzése véleményem szerint a szükségesnél hosszabb időre foglalja le a tesztelői erőforrásokat.

P11 A regressziós tesztekhez nem áll rendelkezésre teszteset lista, ezért abban az esetben, ha külső tesztelők bevonása történik a regressziós teszt során és az adott tesztelő nem rendelkezik megfelelő tudással a termék modullal kapcsolatban, akkor nem biztosítható, hogy a tesztelő képes a megfelelő mélységű tesztelés elvégzésére és a hibák detektálására.

A folyamatok elemzése során 11 problémát azonosítottam. A problémák megoldására tett javaslatok a 5.2 fejezetben kerülnek kifejtésre.

Az új funkciók szoftvertesztelési folyamatának ábrázolását folyamatábra segítségével a 11. ábra mutatja be.



11. ábra: Új funkció tesztelésének folyamata a Handyman részlegen. Saját készítésű ábra, saját kutatás alapján

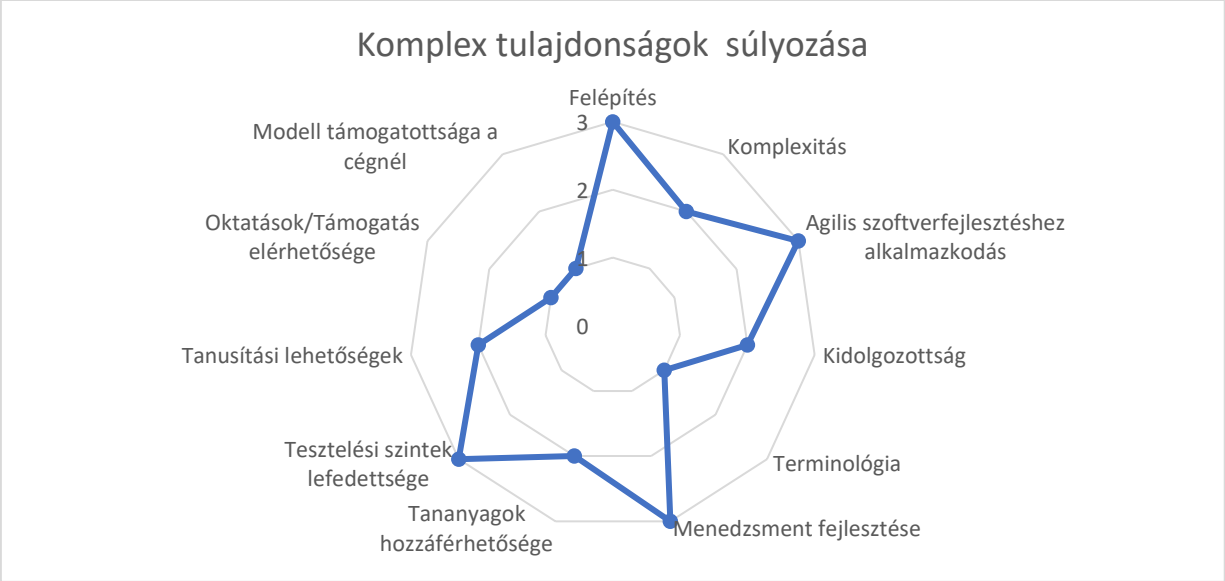
4.2.2. Szoftvertesztelési folyamat fejlesztési referencia modell kiválasztása

Ebben a fejezetben komplex tulajdonságok összemérésével fogok döntést hozni arról, hogy a GSGroup Handyman részlege számára a két legelterjedtebb folyamat fejlesztési referencia modell (TMMi és TPI Next) közül melyik az, amelyik összességében jobban illeszkedik a modell bevezetésével kapcsolatos célokhoz, illetve könnyebben és hatékonyabban bevezethető.

Összehasonlítás során vizsgált komplex tulajdonságok és azok súlyozása

A szoftvertesztelői csapat vezetőjével folyatatott megbeszélések alapján, összeségében egy olyan folyamatfejlesztési referencia modellt keresek, amely jobban illeszkedik az alábbi elvárásokhoz: magában foglalja a szervezetszintű minőséggel kapcsolatos gondolkodásmód kialakítását/fejlesztését, értelmezhető a szoftverteszteléssel foglalkozó kollégák számára, könnyen elérhetőek a modell bevezetését támogató szolgáltatások és tananyagok, illetve tanúsítási lehetőség, a modell rendelkezik az agilis szoftverfejlesztéssel és szoftverteszteléssel kapcsolatos kiegészítéssel, illetve mennyire konkrétan fogalmazza meg az adott fejlesztendő területeket és a fejlesztési irányt és az hogyan illeszkedik a GSGroup Handyman részlegéhez.

A komplex tulajdonságok összemérése során az alábbiakban felsorolt tulajdonságokat 1-3-as skálán való fontossági értékkel fogom figyelembe venni. A kiválasztott komplex tulajdonságokat és azok súlyozását vizuálisan a 12. ábra szemlélteti.



12. ábra: Komplex tulajdonságok során vizsgált szempontok és azok súlyozásának ábrázolása pókháló diagramon. Saját készítésű ábra, saját kutatás alapján.

X01. **Modell felépítése** - Fontosság: 3 – Érték: 1..5

Mivel a szervezetnek nincs tapasztalata folyamatfejlesztéssel kapcsolatos tevékenységekben, ezért véleményem szerint hatékonyabb lenne egy olyan modellt használni, ami egyértelműen kijelöli a fejlesztendő területeket és a szükséges tevékenységeket, illetve nem egyszerre próbálja meg a teljes folyamatot fejleszteni, hanem lépcsőzetesen, így jobban lehetne az adott területekre fókuszálni a fejlesztés során. Ezt az is indokolja, hogy a folyamatfejlesztéshez korlátozottan fognak rendelkezésre állni az erőforrások, illetve biztos vagyok benne, hogy lesznek olyan időszakok (például regressziós tesztek során), amikor akár hetekig nem lesz delegált erőforrás a fejlesztésre. Minél jobban implementálható a modell felépítése a Handyman részlegen, annál magasabb értéket kap a szempont.

X02. **Folyamatfejlesztési modell komplexitása** - Fontosság: 2 – Érték: 1..5

Abban az esetben, ha a modell felépítése és bevezetése kevésbé komplex, egyszerűbb a modell megértése, illetve könnyebben lehet bevezetni a szervezetbe. Emellett minél egyszerűbben és gyorsabban értelmezhetőek a modell által szolgáltatott információk, annál gyorsabban és könnyebben születhetnek meg a folyamatfejlesztéssel kapcsolatos döntések.

Minél kevésbé komplex a modell, illetve a szolgáltatott információk, annál magasabb értéket kap az adott szempont.

X03. **Alkalmazkodás az agilis szoftverfejlesztéshez** - Fontosság: 3 – Érték: Igen/Részlegesen/ Nem

Mivel a Handyman részleg fejlesztőcsapata agilis, azon belül is Scrum módszertannal dolgozik, ezért fontos, hogy a modell ne csak az általános fejlesztési modellekre legyen alkalmazható (például vízesés modell), hanem legyen olyan kiegészítése is, amely az agilis szoftverfejlesztésben való alkalmazását támogatja. Ennél a szempontnál az „Igen" válasz jelenti a magasabb értéket.

X04. **Modell kidolgozottsága, gyakorlati példák** - Fontosság: 2 – Érték: 1..5

A szoftvertesztelő csapat nem rendelkezik folyamatfejlesztési tapasztalattal, ezért véleményem szerint mindenképpen jobb egy olyan modellt alkalmazni, ami részletesen kidolgozott, illetve gyakorlati példákon keresztül mutatja be az adott részfolyamatok, folyamatterületek fejlesztéséhez szükséges tevékenységeket, dokumentumokat stb. Minél több gyakorlati példát szolgáltat a modell, annál magasabb értéket kap az adott szempont.

X05. **Modellben használt terminológia ismerete** - Fontosság: 1 – Érték: Igen/Részlegesen/Nem

A referencia modellben használt terminológia, azért fontos, mert a modell tanulását, megértését jelentősen segíti, ha olyan szaknyelven van megfogalmazva, amit a Handyman csapatban dolgozó szoftvertesztéléssel foglalkozó kollégák ismernek. A csapat számára jól ismert terminológia csökkenti a modell implementálásához szükséges időt. A Handyman csapat szoftvertesztéléssel foglalkozó minden tagja rendelkezik legalább alapszintű ISTQB vizsgával, melynek feltétele az ISTQB [14]fogalomtár (ISTQB Glossary) ismerete. Ennél a szempontnál az „Igen" válasz kapja a magasabb értéket.

X06. **Menedzsment fejlesztésének megjelenése a modellben** - Fontosság: 3 – Érték: 1..5

[14] ISTQB: International Software Testing Qualification Board - Nemzetközi Szoftvertesztelés Minősítő Testület

A GSGroup Handyman részlegén véleményem szerint nincs egy kialakult minőségkultúra, a menedzsment ritkán, illetve alacsony súllyal veszi figyelembe a minőségi tényezőket. Emiatt fontosnak tartom, hogy az alkalmazandó folyamatfejlesztési modell szerves része legyen a minőségkultúra kialakítása a menedzsmentben, illetve a szoftvertesztelés és minőség fontosságának elterjesztése a szervezetben. Minél jobban szerepet kap a menedzsment fejlesztése a modellben, annál magasabb értéket kap ez a szempont.

X07. **Modellhez elérhető anyagok és azok hozzáférhetősége** - Fontosság: 2 – Érték: 1..5

Mivel a szoftvertesztelő csapat nem rendelkezik jelenleg megfelelő szintű szaktudással a folyamatfejlesztési modellekkel kapcsolatban, ezért a folyamatfejlesztés során mindenképpen szükség lesz az elérhető anyagok alapján a csapat képzésére. Ennél a szempontnál előnyt jelent, ha a modellhez kapcsolódó tananyagok hivatalos internetes forrásból könnyen elérhetőek, esetleg ingyenesen letölthetőek a csapat minden tagja számára. Minél jobban hozzáférhetőek és elérhetőek a modellhez kapcsolódó tananyagok, annál magasabb az adott szempont értéke.

X08. **Modell által lefedett tesztelési szintek** – Fontosság: 3 – Érték: Igen/Részleges/Nem

A Handyman részlegen több tesztelési szinten történik ellenőrzés. A szoftverfejlesztők oldaláról készülnek egység tesztek még a tesztelőknek való funkció átadás előtt, a projektek során a tesztelő csapat tagjai végeznek integrációs teszteket, felhasználói elfogadási teszteket, illetve rendszer teszteket. Mivel mind a 4 tesztelési szinten végzünk tesztelést, ezért a teljeskörűség érdekében véleményem szerint jobb egy olyan tesztelés folyamatfejlesztési modell, amely mind a négy tesztelési szintet lefedi. Ha a modell lefedi az általunk használt 4 tesztelési szintet, akkor „Igen” értéket kap az adott szempont.

X09. **Modell tanúsítási lehetőségei Magyarországon** - Fontosság: 2 – Érték: Igen/Részlegesen/Nem

A jelenlegi ügyfélkör nem követeli meg a tanúsítványokat, ugyanakkor a későbbiekben, az ügyfélkör bővítése során bármikor felmerülhetnek olyan ügyféligények, melyek megkövetelik a bizonyos szabványoknak való megfelelést. Emellett véleményem szerint az

ilyen típusú minősítések megkönnyítik az új ügyfelekben a bizalom kiépítését a termék felé. Ebben az esetben az „Igen” válasz jelenti a magasabb értéket.

X10. **Modell bevezetését támogató szolgáltatások/oktatások elérhetősége Magyarországon** - Fontosság: 1 – Érték: „Igen/Részleges/Nem”

Mivel a szoftvertesztelő csapat nem rendelkezik elég tapasztalattal, ezért a modell kiválasztása során azt is figyelembe kell vennem, hogy bizonyos lepéseknél lehet, hogy szükség lesz külső szakértők bevonására vagy oktatások szervezésére. Ebben az esetben az „Igen” válasz jelenti a magasabb értéket.

X11. **Modell támogatottsága a szoftvertesztelők körében**– Fontosság: 1 – Érték: 1..5

Előzetesen már szóba került a szoftvertesztelő csapatban a folyamat fejlesztésének szükségessége. Ugyan a csapat tagok nem rendelkeznek teljeskörű rálátással a modellekre, de az azokban, akik foglalkoztak a témával, már kialakult egy kép arról, hogy melyik modellel dolgoznának szívesebben. Ez a tényező azért kap szerepet, mert minél inkább elkötelezettek a csapattagok egy adott modell alkalmazásával kapcsolatban, annál könnyebb lesz implementálni a változtatásokat.

Modellek összehasonlítása a komplex tulajdonságok alapján

Az vizsgálandó tulajdonságok meghatározása után a következő lépés a tulajdonságok értékelése az adott modellekre vonatkozóan. A modell felépítése (X01) a TMMi esetében 5, a TPI Next esetében 3 pontot kap a TMMi modell szakaszos felépítése miatt. A folyamatfejlesztési modell komplexitása (X02) a TMMi modellnél 4 a TPI Next modellnél 5 pontot kap, mivel a TPI Next modell kevésbé komplex. Mindkét modell alkalmazkodik az agilis szoftverfejlesztéshez (X03) ezért mindkét esetben „Igen” az érték. A modell kidolgozottsága (X04) szempontnál a TMMi több gyakorlati példát használ, ezért a TMMI 5, a TPI Next 4 pontot kap. A modell terminológiája (X05) a TMMi modell esetében ismert „Igen” értéket kap, a TPI Next használ egyedi fogalmakat is, ezért „Részleges” értéket kap. A menedzsment fejlesztése (X06) a TMMi modellben sokkal erőteljesebben jelenik meg, ezért a

TMMi 5, a TPI Next 4 pontot kap. A TMMi modellhez egy kicsivel egyszerűbben elérhetőek az anyagok (X07), ezért a TMMi 5, a TPI Next 4 pontot kap. A TMMi modell teljesen, a TPI Next csak részlegesen fedi le az alkalmazott tesztelési szinteket (X08), ezért a TMMi „Igen", a TPI Next „Részlegesen" értéket kap. A modellek tanúsítása Magyarországon (X09) a TMMi esetében „Igen", a TPI Next esetében „Nem" értéket kap. Magyarországon a TMMi-hoz több modell bevezetését támogató szolgáltatás (X10) érhető el, ezért a TMMi „Igen", a TPI Next „Részleges" értéket kap. Végül a TMMi modell támogatottsága (X11) egy kicsivel magasabb a szervezeten belül ezért a TMMi 5, a TPI Next 4 pontot kap. Az értékelés összesített eredménye a 2. táblázatban látható.

2. táblázat: Vizsgált tulajdonságok étékelése a TMMi és a TPI Next folyamatfejlesztési referencia modellekre vonatkozóan.

Tulajdonság / Modell	**X01**	**X02**	**X03**	**X04**	**X05**	**X06**	**X07**	**X08**	**X09**	**X10**	**X11**
TMMi	5	4	Igen	5	Igen	5	5	Igen	Igen	Igen	5
TPI Next	3	5	Igen	4	Rész.	2	4	Rész.	Nem	Rész	4

Saját készítésű táblázat, saját kutatás alapján

A következő lépés a vizsgált tulajdonságoknál a kvalitatív szempontok számszerűsítése (Igen = 3 pont, Részlegesen = 2 pont, Nem = 1 pont). A vizsgálat során nem használtam olyan értékékeket melyek más irányba mutatnak, ezért a vizsgálat során nincs szükség az azonos irányú adatok kialakítására, minden tulajdonság esetében a nagyobb érték a jobb. Ezek alapján a transzformált adatok kiszámítása során mindig a magasabb érték fogja felvenni az „1" értéket és az adott oszlopban található másik adat pedig ehhez képest fog viszonyított értéket kapni. Ezek alapján a transzformált adatokat a 3. táblázat tartalmazza. A transzformált adatok alapján látható, hogy a 11 vizsgált tulajdonság esetében 10-nél magasabb pontszámot kapott a TMMi modell.

3. táblázat: Vizsgált tulajdonságok transzformált adatai a TMMi és TPI Next folyamatfejlesztési referencia modellre vonatkozóan. Forrás: Saját készítésű táblázat

Tulajdonság / Modell	**X01**	**X02**	**X03**	**X04**	**X05**	**X06**	**X07**	**X08**	**X09**	**X10**	**X11**
TMMi	1	0,8	1	1	1	1	1	1	1	1	1
TPI Next	0,6	1	1	0.8	0,66	0,4	0,8	0,66	0,33	0,33	0,8

Saját készítésű táblázat, saját kutatás alapján

Az adott tulajdonságok fontosságát 1-3-as skálán értékeltem. Ahhoz, hogy a szempontokat egyesíteni tudjam, a súlyrendszer kialakítása során a súlyok összegének 1-nek kell lennie. A fontossági pontok összege 23, amit úgy kell felosztani, hogy megegyezzen 1-gyel. Tehát ezek alapján 1 darab fontossági pont értéke: 1/23=0,0434. A tulajdonságokhoz meghatározott fontossági pontok átalakítását után a tulajdonságokhoz rendelt súlyok: X01 X03 X06 X08 =0.1302, X02 X04 X07 X09=0.0868, X05 X10 X11=0-0434.

Az adott tulajdonságok súlyozási értékkének és az adott modellhez tartozó érték összeszorzásával, majd a kapott eredmények összegzésével kiszámolható az adott modellhez tartozó végső pontszám. A súlyozási értékek és a tulajdonságokhoz tartozó pontszámok összeszorzásából kapott eredményt a 4. táblázat mutatja be, amiről leolvasható, hogy 11-ból 10 esetben a TMMi kapott magasabb pontszámot.

4. táblázat: Tulajdonságokhoz rendelt súlyok és értékek összeszorzásából kapott végső eredmények meghatározása. Forrás: Saját készítésű táblázat

Tulajdonság / Modell	**X01**	**X02**	**X03**	**X04**	**X05**	**X06**	**X07**	**X08**	**X09**	**X10**	**X11**
TMMi	0,1302	0,0694	0,1302	0.0868	0,0434	0,1302	0.0868	0,1302	0.0868	0,0434	0,0434
TPI Next	0,0781	0.0868	0,1302	0,0694	0,0286	0,052	0,0694	0,0859	0,0286	0,0143	0,0347

Saját készítésű táblázat, saját kutatás alapján

Az adott modellekhez kapott súlyozott értékek összeadása után az alábbi értékeket kaptam:

TMMi folyamatfejlesztési referencia modell = 0,98 **= 98%**

TPI Next folyamatfejlesztési referencia modell = 0,678 **= 67,8%**

A kapott eredmények alapján egyértelműen megállapítható, hogy a GSGroup Handyman részleg szoftvertesztelési folyamatinak fejlesztésére a **TMMi folyamatfejlesztési referencia modell a jobb választás**. Az eredmények alapján a továbbiakban a TMMi modellel fogok dolgozni.

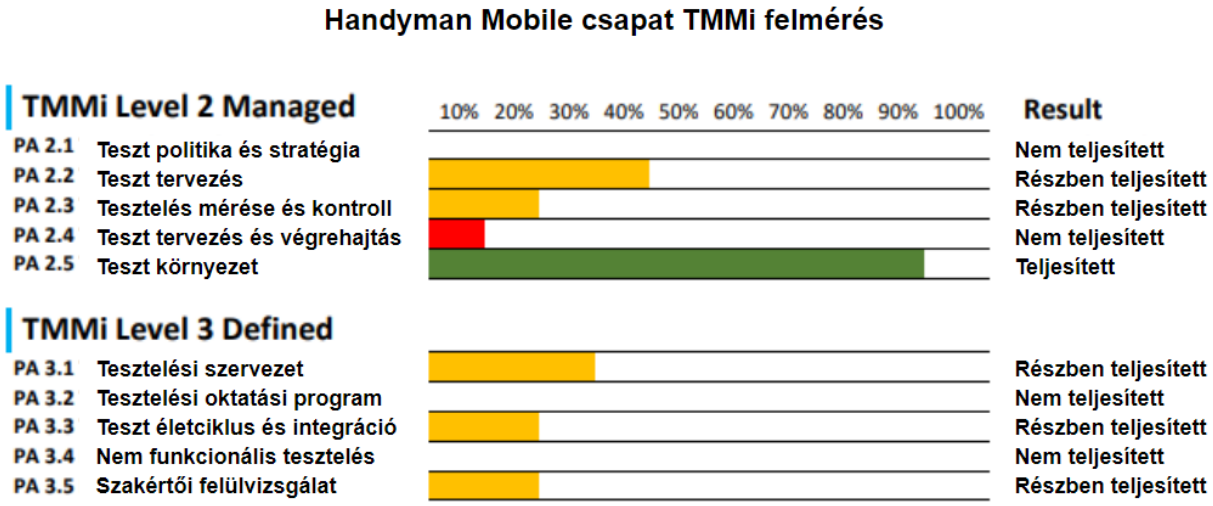
4.2.3. A tesztelési folyamat aktuális TMMi érettségi szintjének felmérése

A „TMMi Lightning Scan Tool" -t használatával felmérhető, hogy a szervezet tesztelési folyamatai jelenleg melyik érettségi szinten helyezkednek el. Az eszköz a 2. (Menedzselt) és 3. (Meghatározott) folyamat érettségi szinteken található folyamatterületek felmérését támogatja.

Mivel Handyman részlegesen az agilis kiáltvány szerinti szoftverfejlesztés és csapatfelosztás történik, ezért nagy hangsúlyt kap a csapatok önmenedzselése. Mivel minden csapat más és más szoftvertesztelőkkel dolgozik, ezért a csapatok között eltérések lehetnek a 2. (Menedzselt) érettségi szinten meghatározott folyamatterületek teljesítettségének mértékében. Ebből adódóan a teljeskörű helyzet áttekintés érdekében mindegyik csapatra (Handyman Mobile, Handyman Office és Handyman Web) külön-külön felmérést végeztem. A 3. (Meghatározott) érettségi szinten a teljes szervezetre vonatkozó folyamat területek vannak, vagyis ezen a szinten nincs eltérés a különböző csapatok között, ezért a 3. érettségi szintre kapott eredmények közösen kerülnek elemzésre, abban az esetben, ha a jelenlegi folyamat teljesíti a 2. érettségi szintet. A Test Maturity Model integration szabályai alapján csak akkor léphet a szervezet a következő szintre, ha az adott szint minden folyamat területe legalább 85%-ban teljesített. Abban az esetben, ha az adott folyamat terület értéke 50 - 85%, akkor az „Majdnem teljesített" állapotban van. Ha a folyamat terület értéke 15 – 50% között van, akkor annak státusza „Részlegesen teljesített". Végül, ha az érték 15% alatt van, akkor az adott folyamat terület állapota „Nem teljesített".

A tesztelési folyamat aktuális TMMi érettségi szintjének felmérése a Handyman Mobile csapatban

A Handyman Mobile csapatban végzett felmérés eredményét az 13. ábra mutatja.



13. ábra: Előzetes TMMi felmérés eredménye a Handyman Mobile csapatban. Saját készítésű ábra, saját kutatás alapján.

A megelőző felmérés eredményei a 2. – Menedzselt érettségi szintre vonatkozóan Handyman Mobile csapatban:

- **Tesztpolitika és Tesztelési stratégia**

  A tesztpolitika, illetve a szervezet és program szintű tesztstratégia **nem valósul meg** sem a csapaton, sem a Handyman részlegen belül. Ebből adódóan a csapaton belül és az érintettek körében nincs egy általánosan kialakult / elfogadott kép a tesztelés céljairól és a teszteléssel foglalkozó szakemberekkel kapcsolatban. Nincsenek meghatározott teljesítmény mutatók a tesztelésre vonatkozóan, illetve nem tisztázott a tesztelés és az üzleti célok kapcsolata.

- **Tesztelés tervezése**

A tesztelés tervezése **részlegesen megvalósul** a csapaton belül, ugyanakkor a kockázatok teljeskörű azonosítása nem jelenik meg a folyamatban, illetve a teszteléshez szükséges erőforrások/idő becsléséhez nincsenek kialakult módszerek és mérőszámok. Mivel a kockázatok azonosítása csak részlegesen történik meg, ezért a megfelelő alkalmazandó tesztelési módszerek meghatározása sem biztosított. A tesztelési terv sem teljeskörű, hiszen nem jelennek meg benne a projektkockázatok és a pontosan definiált tesztelési megközelítés, illetve a menedzsmentnek alacsony a rálátása a teszttervekre.

- **Tesztelés mérése és kontroll**

  A tesztelés mérése és kontrollja **alacsony szinten megvalósul** a csapaton belül. A terv-tény összevetés általában ad-hoc jelleggel megjelenik, viszont nincs a feladat elvégzésére kijelölt személy és maga a tevékenység sem meghatározott. A termék aktuális minőségéről és a tesztelés előrehaladásáról nincs rendszeres tájékoztatás az érdekelt felek irányába. A lefedettség, illetve a tesztelés során talált hibák ad-hoc jelleggel ellenőrizve vannak, viszont itt sincs kijelölt személy a tevékenység elvégzésére, illetve definiált teendőlista. Abban az esetben, ha a tesztelés előrehaladása nem a terveknek megfelelően történik, nincsenek korrekciós intézkedések, például az elvégzendő feladatok átpriorizálása és ebből adódóan gyakori, hogy a termék adott verziója nem készül el a meghatározott határidőre.

- **Teszt tervezés és végrehajtás**

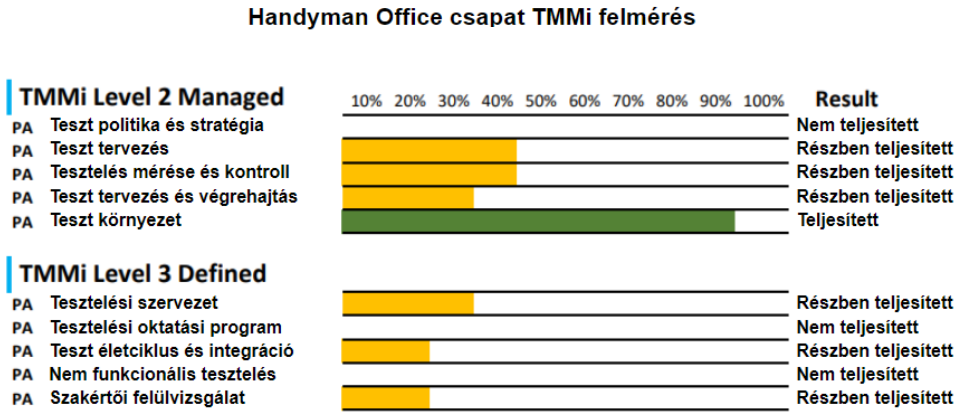
  A teszt tervezés és végrehajtás **nem valósul meg** a csapaton belül. Az új funkciókhoz és a regressziós tesztekhez nincsenek tesztesetek, a tesztelés a tesztelő csapat tagjainak tapasztalata és rendszerismerete alapján történik. A tapasztalat alapú tesztelés a termék tervező által készített specifikáció alapján történik, ugyanakkor a specifikáció minősége, kidolgozottsága sem megfelelő, illetve sok esetben a hibajavítások és új funkciók implementálása után nem kerül frissítésre. Emellett nincsenek definiált eljárások a tesztelés végrehajtására, így jelentős eltérések vannak a tesztelő kollégák által végzett tesztelés mélységében. Nincsenek automatizált tesztek – bár egyrészt a Handyman Mobil alkalmazás tesztelésének automatizálása nagyon nagy erőforrásokat igényelne, másrészt a cégnél dolgozó tesztelők és fejlesztők nem rendelkeznek tapasztalattal a mobilalkalmazás tesztautomatizálás témakörében.

- **Tesztkörnyezet**

  A tesztkörnyezet folyamat terület a csapatban **teljesített** állapotban van. A mobilalkalmazás esetében nincs szükség külön fejlesztői és tesztkörnyezetre, hiszen minden fejlesztői módosítás beépítése után létrejön egy telepítő fájl külön-külön Android, iOS és Windows platformra. Mivel nincsenek tesztesetek, így a tesztesetekhez tartozó tesztadatok kezelése sem valósul meg, ugyanakkor nincs definiált folyamat arra, hogy hogyan kell / hogyan lehetne a teszt adatokat kezelni.

A tesztelési folyamat aktuális TMMi érettségi szintjének felmérése az Office csapatban

A Handyman Office csapatban végzett felmérés eredményét a 14. ábra mutatja.



14. ábra: Előzetes TMMi felmérés eredménye a Handyman Office csapatban. Saját készítésű ábra, saját kutatás alapján.

A megelőző felmérés eredményei a 2. – Menedzselt érettségi szintre vonatkozóan Handyman Office csapatban:

- **Tesztpolitika és Tesztelési stratégia**

  A tesztpolitika, illetve a szervezet és program szintű tesztstratégia **nem valósul meg** sem a Handyman Office csapatban, sem a Handyman részlegen belül. A folyamatterület felmérésének eredménye megegyezik a Handyman Mobile csapat eredményeivel.

- **Tesztelés tervezése**

  A tesztelés tervezése **részlegesen megvalósul** a csapaton belül, ugyanakkor a kockázatok teljeskörű azonosítása nem jelenik meg a folyamatban, illetve a teszteléshez szükséges erőforrások/idő becsléséhez nincsenek kialakult módszerek és mérőszámok. Mivel a kockázatok azonosítása csak részlegesen történik meg, ezért a megfelelő alkalmazandó tesztelési módszerek meghatározása sem biztosított. A folyamatterület felmérésének eredménye megegyezik a Handyman Mobile csapat eredményeivel.

- **Tesztelés mérése és kontroll**

  A tesztelés mérése és kontrollja **részlegesen megvalósul** a csapaton belül. A folyamatterület felmérésének eredménye hasonló a Handyman Mobile csapat eredményeivel, ugyanakkor a Handyman Office alkalmazás nagyobb figyelmet kap a menedzsment részéről így abban az esetben, ha a tesztelés haladása vagy a termék minőség jelentősen eltér a tervezettől akkor az értintettek bevonásával megjelenhetnek korrekciós intézkedések.

- **Teszt tervezés és végrehajtás**

  A teszt tervezés és végrehajtás **részlegesen megvalósul**. Az új funkciókhoz ad-hoc jelleggel készülnek munkafolyamat alapú tesztesetek, ugyanakkor itt is jellemző a tapasztalat alapú tesztelés a termék tervező által készített specifikáció alapján. A Handyman Office termékhez tartozó specifikáció sem teljeskörű és sok esetben elmaradnak a frissítések. A regressziós tesztben meghatározott területek egy részéhez szintén rendelkezésre állnak magas szintű munkafolyamat alapú tesztesetek. Itt sincsenek definiált eljárások a tesztelés végrehajtására, illetve a tesztautomatizálás a rendszer komplexitása és jellege miatt nem jön szóba.

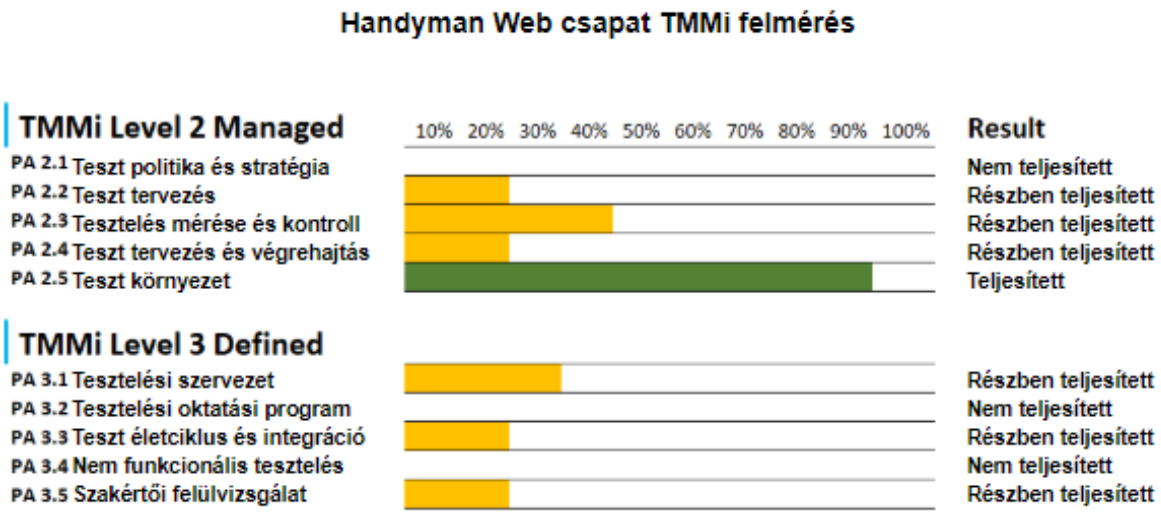
- **Tesztkörnyezet**

  A tesztkörnyezet folyamat terület a csapatban **teljesített** állapotban van. A Handyman Office alkalmazás esetében az új funkciók és hibajavítások a fejlesztői kódok összefésülése után azonnal elérhető tesztelésre. A munkafolyamat alapú tesztesetekhez nem tartoznak

tesztadatok, így azok kezelése sem valósul meg, ugyanakkor nincs definiált folyamat arra, hogy hogyan kell / hogyan lehetne a teszt adatokat kezelni.

A tesztelési folyamat aktuális TMMi érettségi szintjének felmérése a Handyman Web csapatban [E]

A Handyman Web csapatban végzett felmérés eredményét a 15. ábra mutatja.



15. ábra: Előzetes TMMi felmérés eredménye a Handyman Office csapatban. Saját készítésű ábra, saját kutatás alapján.

A megelőző felmérés eredményei a 2. – Menedzselt érettségi szintre vonatkozóan Handyman Web csapatban:

- **Tesztpolitika és Tesztelési stratégia**

  A tesztpolitika, illetve a szervezet és program szintű tesztstratégia **nem valósul meg** a részlegen belül. A folyamatterület felmérésének eredménye megegyezik az előző két csapat eredményeivel.

- **Tesztelés tervezése**

A tesztelés tervezése **részlegesen megvalósul** a csapaton belül, ugyanakkor a kockázatok azonosítása nem jelenik meg a folyamatban, illetve a teszteléshez szükséges erőforrások/idő becsléséhez nincsenek kialakult módszerek és mérőszámok. Mivel a kockázatok azonosítása nem történik meg, ezért a megfelelő alkalmazandó tesztelési módszerek meghatározása sem biztosított. Ebben a csapatban a tesztelés tervezése nem valósul meg formálisan, nem készül tesztelési terv és ebből adódóan a menedzsment és érintettek bevonása sem történik meg a tervezés során. A regressziós tesztek során az Azure DevOps rendszerben tárolt, sok esetben elavult és nem megfelelő lefedettségű tesztesetek futtatása történik. A Web csapat több termékért felelős és vannak olyan alkalmazások, ahol nagyon alacsony lefedettségű tesztelés történik.

- **Tesztelés mérése és kontroll**

  A tesztelés mérése és kontrollja **részlegesen megvalósul** a csapaton belül. A tesztesetek elérhetőek az Azure DevOps rendszerben, ebből adódóan nyomon követhető, hogy hogyan halad a tesztesetek futtatása. A termék aktuális minőségéről és a tesztelés előrehaladásáról nincs rendszeres tájékoztatás az érdekelt felek irányába. Abban az esetben, ha a tesztelés előrehaladása nem a terveknek megfelelően történik, nincsenek korrekciós intézkedések.

- **Teszt tervezés és végrehajtás**

  A teszt tervezés és végrehajtás **részlegesen megvalósul**. A termék egy adott verziójához készültek tesztesetek, de ezeknek alacsony a lefedettsége, illetve sok esetben elavultak. Az új funkciókhoz ad-hoc jelleggel készülnek új tesztesetek, ugyanakkor itt is jellemző a tapasztalat alapú tesztelés a termék tervező által készített specifikáció alapján (a Handyman Webes termékhez tartozó specifikációk a leghiányosabbak és legritkábban frissítettek). Nincsenek definiált eljárások a tesztelés végrehajtására, a tesztautomatizálás bevezetésének tervezése elkezdődött.

- **Tesztkörnyezet**

A tesztkörnyezet folyamat terület a csapatban **teljesített** állapotban van. Mivel nincsenek tesztesetek, így a tesztesetekhez tartozó tesztadatok kezelése sem valósul meg, ugyanakkor nincs definiált folyamat arra, hogy hogyan kell / hogyan lehetne a teszt adatokat kezelni.

4.2.4. Fejlesztendő területek meghatározása az elvégzett TMMi felmérés alapján

Az előzőekben elvégzett folyamat érettségi szint felmérése alapján a 2. érettségi szint teljesítéséhez jelentősen **javítani kell a tesztpolitika és stratégia, tesztelés tervezése, tesztelés mérése és kontroll, illetve a teszt tervezés és kontroll folyamatterületeket** a Handyman részleg minden csapatában, a tesztkörnyezet folyamatterület teljesített állapotban van, bár a 100%-os teljesítettség eléréséhez opcionálisan ez is javítható. Ebben a fejezetben a felmérés alapján meghatározom azokat a specifikus célokat, amelyeket meg kell valósítani az adott folyamatterület teljesítéséhez. A specifikus célok olyan kötelező elemeket jelölnek a folyamatban, melyeknek meg kell valósulniuk az adott folyamatterület teljesítéséhez. Mivel a folyamat fejlesztéshez a TMMi modell lett kiválasztva, ezért a gyakorlati megvalósítás megkönnyítéséhez (ami nem a diplomadolgozatom része) a fejlesztendő folyamatterületeket és a hozzájuk kapcsolódó specifikus célokat összekapcsolom a TMMi modellben alkalmazott jelölésekkel és számozásokkal. Jelölések: [PA], mint Proccess Area, vagyis folyamatterület és [SG], mint Specific Goal, vagyis specifikus cél.

Az előzetes felmérés alapján a „[PA2.1] **Tesztelési politika és tesztelési stratégia**” folyamatterületen 0%-ban teljesített, vagyis TMMi-ban meghatározott **specifikus célok közül egyik sem teljesített**, ezért a területhez tartozó minden specifikus célt meg kell meg kell valósítani. Ezek alapján a megvalósítandó specifikus célok:

- [SG1] Tesztpolitika létrehozása
- [SG2] Tesztelési stratégia létrehozása
- [SG3] Teljesítménymutatók meghatározása és alkalmazása

Az előzetes felmérés alapján a „[PA2.2] **Tesztelés tervezése**” folyamatterület csapattól függően **nem teljesített/részben teljesített** állapotban van, ugyanakkor minden csapatban javítani kell vagy létre kell hozni a folyamatterülethez tartozó összes specifikus célt. Ezek alapján a megvalósítandó, javítandó specifikus célok:

- [SG1] Termékkockázatok értékelésének kialakítása és alkalmazása
- [SG2] Kockázat alapú tesztelési megközelítés meghatározása
- [SG3] Strukturált módszer kialakítása és bevezetése a tesztelés becsléséhez (például idő és erőforrás szükségletek)
- [SG4] Tesztelési terv létrehozása
- [SG5] Teszterv iránti elkötelezettség kialakítása a csapatban

Az előzetes felmérés alapján a „[PA2.3] **Tesztelés monitorozása és ellenőrzése**” folyamatterület **részben teljesített** állapotban van, ugyanakkor egyik specifikus cél sem valósul meg teljesen. Ezek alapján a javítandó specifikus célok:

- [SG1] Tesztelés haladásának monitorozása a teszttervhez képest
- [SG2] Termék aktuális minőségének felülvizsgálata és nyomon követése a terv és az elvárások alapján
- [SG3] Korrekciós intézkedések alkalmazásának bevezetése elvárásoktól való eltérés esetén

Az előzetes felmérés alapján a „[PA2.4] **Teszt tervezés és végrehajtás**” folyamatterület csapattól függően **nem teljesített/részben teljesített** állapotban van. A felmérés alapján egyik csapatban sem teljesülnek teljesen a modellben meghatározott specifikus célok. Ezek alapján a megvalósítandó, javítandó specifikus célok:

- [SG1] Tesztelemzés és tesztesettervezés teszttervezési technikák segítségével
- [SG2] Teszt implementáció végrehajtása
- [SG3] Tesztelés végrehajtása
- [SG4] Tesztelési események (például hiba találat, technikai adósság) menedzselése

Az előzetes felmérés alapján a „[PA2.5] **Tesztkörnyezet**” folyamatterület minden csapatban **teljesített** állapotban van. Vagyis ezen a folyamatterületen nem kell javításokat elvégezni a specifikus célokkal kapcsolatban.

5. Következtetések és javaslatok

5.1. Handyman alkalmazás csomag sérülékenységeivel kapcsolatos következtetések és javaslatok

A 4.1.2-es fejezetben elvégzett felméréseim alapján arra a következtetésre jutottam, hogy a Handyman **termék csomag erősen kitett a szoftversérülékenységekkel szemben**, a vizsgált 25 sérülékenységből 16 darab megjelenik a szoftverben. Véleményem szerint ennek az eredménynek az oka, hogy a termékfejlesztés és tesztelés során a részleg nem fordít kellő figyelmet az ilyen típusú kockázatokra, illetve a minőségbiztosítási csapat tagjai nem rendelkeznek a megfelelő szintű tudással az szoftversérülékenységekből adódó hibák kiszűrésére.

A diplomadolgozatom során elvégzett kutatásaim eredményeként, megállapítottam sérülékenységekből adódó kockázatok jelentősen csökkenthetőek, ha a termékfejlesztés folyamatába bekerülnek olyan tevékenységek, amik a sérülékenységek kiszűrésére és javítására összpontosítanak. A 4.1.3-as fejezetben bemutatott Handyman termékcsomagban található szoftversérülékenységek csökkentésére vonatkozó három megoldási javaslatom tulajdonságait az 5. táblázat tartalmazza:

5. táblázat: Szoftversérülékenységek csökkentésére vonatkozó megoldási javaslatok tulajdonságai.

Megoldási változat / Tulajdonságok	**Tesztelő csapat továbbképzése**	**IT biztonsági alkalmazott felvétele**	**Szakértői csapat felkérése**
Sérülékenység detektálási hatékonyság	3	4	5
Detektált sérülékenységek javításának megvalósulása	5	5	3
Költség / 1 év	1 147 500 Ft	14 560 000 Ft	14 399 988 Ft

Saját készítésű táblázat, saját kutatás alapján.

Az elvégzett számításaim során arra a következtetésre jutottam, hogy a tesztelő csapat továbbképzésének sérülékenység detektálási hatékonysága közepes, de ezzel szemben a detektált sérülékenységek javításának megvalósulása kiemelkedő, a változat bevezetésének költsége pedig 1 147 500 Ft egyszeri költséget jelent. IT biztonsági alkalmazott felvétele esetében a sérülékenység detektálási hatékonyság jó, és a detektált sérülékenység javításának megvalósulása kiemelkedő, a változat bevezetésének költsége egy egyszeri 1 000 000 Ft-os és egy folytonos 13 560 000 Ft/év-es költségből adódik össze, tehát jelentősen magasabb, mint az előző esetben. A szakértői csapat felkérése esetén a sérülékenység detektálási hatékonyság kiemelkedő, viszont a detektált sérülékenység javításának megvalósulása csak közepes szintet ér el, a változat bevezetésének költsége pedig egy folytonos 14 399 988Ft/év-es költség, ami körülbelül megegyezik az IT biztonsági alkalmazott felvételének költségeivel.

Az összesített adatok alapján úgy gondolom, hogy hosszútávon a szoftvertesztelői csapat továbbképzése lenne a legjobb választás, mivel az adott megoldás a többi javaslathoz képest sokkal alacsonyabb egyszeri költséggel jár és már egy ilyen szintű vizsgálat is lehetővé teszi a magasabb kockázatú sérülékenységek detektálását és ezzel jelentősen csökkentené a Handyman termék biztonsági sérülékenységekkel szembeni kitettségét.

5.2. Szoftvertesztelési folyamat elemzése során detektált problémák és azok megoldására tett javaslatok

A 4.2.1-es fejezetben a fejlesztési és szoftvertesztelési folyamat vizsgálata során 11 olyan problémát azonosítottam, amik javításával hatékonyabbá tehető a szoftvertesztelési folyamat. Az elemzésem során azonosított problémák javítására tett javaslataimat a következők:

P01 **Nem megfelelő specifikáció**. Az elemzés során megállapítottam, hogy a specifikáció hiányosságaiból, elavultságából adódóan sok esetben nem lehet eldönteni, hogy a termék megfelel-e a követelményeknek. Mivel a specifikáció bármely részeben találhatóak hiányosságok, ezért véleményem szerint a tesztelőknek meg kell vizsgálniuk a Handyman

Office, Handyman Mobile és Handyman Web termékek működését leíró dokumentumokat is. A vizsgálat során célszerűnek tartom elkészíteni egy listát a hiányos/elavult részekről, majd a listát továbbítani a termék tervező részére, aki ellenőrzi a tesztelők által megjelölt részeket és frissíti azokat, az aktuálisan elvárt termék működés alapján (akár a termék tulajdonossal konzultálva). Ahhoz, hogy a specifikáció naprakész maradjon, javasolt lenne bevezetni egy olyan tevékenységet mely a specifikáció karbantartására irányul. Véleményem szerint megoldás lenne, ha a hibajegyek és új funkciók lezárása előtt a szoftvertesztelő csapat tagjai ellenőriznék, hogy a hozzájuk rendelt hibához tartozó specifikáció rész megfelelően frissítve lett és abban az esetben, ha ez nem történt meg, jelzik a problémát a terméktervező irányába.

P02 Tesztelői erőforrások nincsenek megtervezve a Sprintekhez. Az eddigi tapasztalataim alapján ahhoz, hogy a Sprint folyamata jól mérhető legyen, szükség van a teljes Scrum csapat tevékenységeinek nyomon követesére. Mivel a Sprint tervező megbeszéléseken a tesztelők is részt vesznek, véleményem szerint egyszerűen implementálható lenne a szoftvertesztelők időtervezése a korábbi, illetve jelenlegi Sprintben lévő feladatokra vonatkozóan. Ugyanakkor az intézkedés bevezetésének korai fázisában fontosnak tartom, hogy figyelembe vegyék azt, hogy a szoftvertesztelő kollégáknak nincs tapasztalata a feladatok időbecslésében, ezért lehetnek majd jelentős eltérések a valós és a tervezett eredmény között.

P03 Szükséges szoftvertesztelői erőforrások alul becslése a projekttervezés során. Meggyőződésem, hogy sokkal hatékonyabban lehetne tervezni a projektek erőforrás és időszükségletét, ha az adott fejlesztések időbecslésében nem csak a fejlesztők becsülnék meg az adott feladat elvégzéséhez szükséges időt, hanem a szoftvertesztelők is. Az időbecsléshez pontosításához úgy gondolom, hogy a Scrum csapatnak fel kellene használnia a korábbi hasonló projektekben felhasznált idő és erőforrások mennyiségét (ezek az információk mindenki számára elérhetőek az Azure DevOps rendszerben).

P04 - P05: A Scrum módszertan alapszabályai nincsenek betartva minden Scrum csapatnál. Az elemzésem elvégzése után arra a következtetésre jutottam, hogy Handyman Mobile csapatban nincs olyan személy, aki rendelkezik Scrum Master tapasztalattal és azért nincsenek betartva csapaton belül a Scrum szabályok, mert a csapattagok nincsenek tisztában az adott Scrum események jelentőségével a módszertanban, illetve a csapatnak nincsen Scrum mestere (az egyik szoftverfejlesztő próbálja vezeti a megbeszéléseket). Úgy gondolom, hogy a csapattagok Scrum módszertan irányú oktatásával javítani lehetne a jelenlegi helyzeten. Ugyanakkor a Scrum módszertan szabályainak folyamatos betartásához és működéséhez szükségesnek tartom egy Scrum Mester kijelölését a csapaton belül. Ebben az esetben szerintem költséghatékonyabb megoldás, ha a jelenleg félig Scrum mester szerepet felvevő fejlesztő elvégezne egy Scrum Mester képzést, de természetesen a leghatékonyabb megoldás, ha a csapat kapna egy saját Scrum Mestert.

P06 Visszatekintő megbeszélésen meghatározott fejlesztendő tényezők elhanyagolása. A folyamatelemzésem alapján úgy gondolom, hogy először a Scrum visszatekintő megbeszélés részéve kellene tenni az azonosított problémákra adott megoldási javaslatok kidolgozását. Ezután a kidolgozott fejlesztendő tényezőket újra elő kell venni a Scrum tervező megbeszélésen, feladatokat, felelősöket és delegált időt meghatározni a problémák megoldásához (ezeket az Azure DevOps rendszerben rögzíteni). Ahhoz, hogy megvalósulhasson a folyamatos fejlesztés, szükségesnek tartom, hogy a Sprint során a kijelölt feladat felelősöknek kötelező legyen legalább részlegesen megoldaniuk az adott problémát, majd a Sprint visszatekintő megbeszélésén bemutatni az elért eredményeket.

P07 Specifikációban lévő hibák késői észlelése. A hibaköltségek tízszereződési elmélete alapján, annál olcsóbb egy hiba javítása, minél hamarabban detektálják és javítják a problémát. Ugyanez vonatkozik a specifikációs hibákra is amik a termékfejlesztési folyamat elején jönnek létre. Az elemzés során megállapítottam, hogy a specifikációban lévő hibák utólagos javítása jelentős erőforrásokat igényel. Arra a következtetésre jutottam, hogy a probléma egyszerűen megoldható lenne egy „specifikáció áttekintés” tevékenység bevezetésével a fejlesztési folyamatba, mely során a szoftvertesztelők még a fejlesztés megkezdése előtt megvizsgálják az adott fejlesztendő funkcióhoz elkészült specifikációt. A

gyakorlati megvalósítás során szükségesnek tartom, hogy minden funkció alá létre legyen hozva egy specifikáció áttekintés feladat az Azure DevOps rendszerben, és azok az adott Scrum csapatban dolgozó szoftvertesztelőkhöz legyenek rendelve. Amint a funkcióhoz tartozó specifikáció frissítés feladat elkészül, a terméktervezőnek értesítenie kell az adott tesztelőt arról, hogy megkezdheti a specifikáció áttekintést.

P08 - P09 Eltérések a szoftvertesztelők által végzett munka mélységében. Tapasztalataim alapján közel egy szintre lehetne hozni az adott munkavégzési folyamatokat a szoftvertesztelőkre vonatkozóan, ha bevezetésre kerülne minden hiba és funkció teszteléséhez egy ellenőrző lista, melyet a hiba, vagy funkció tesztelése előtt és közben ki kell tölteni a feladatot végző szoftvertesztelőnek. A létrehozott ellenőrzőlistának tartalmaznia kell többek között az értintett modulok, platformok stb. azonosítását, és a kapcsolódó tesztesetek megírását. Ennek a megoldásnak a másik előnye, hogy a funkciótesztelés során kialakított tesztesetekből létre lehetne hozni tesztesetbázisokat a Azure DevOps rendszer „Teszt tervek" moduljában, ezzel pedig gyorsítható lenne a regressziós tesztek átfutási ideje.

P10 Nincsenek mérőszámok a regressziós tesztek során a prioritások meghatározására. A regresszió prioritások meghatározásánál szükségesnek tartom az összes változtatás összegyűjtését, ami az adott verzióban történt. Tapasztalataim alapján ez egyszerűen megtehető az Azure DevOps Lekérdezések funkcióján keresztül. A kilistázott hibákhoz és funkciókhoz minden esetben tartozik „Érintett modul" paraméter, ez alapján össze lehet gyűjteni, hogy melyek azok a modulok, amelyek érintettek voltak a változtatásokban. Ugyanakkor a gyakorlatban nem minden esetben van jól beállítva a hibákhoz és funkciókhoz tartozó" Érintett modul" paraméter, ezért a végleges lista elkészítse előtt ajánlott végig nézni az összes kilistázott elemet, és ellenőrizni az adatok helyességét. Az „Érintett modulok" lista alapján egyértelműen meghatározhatóak azok a területek melyek nem érintettek a változtatásokban, tehát a regressziós teszt során alacsony prioritással kell figyelembe venni őket. Ezután a többi modulhoz az érintettség foka alapján el lehet dönteni, hogy közepes vagy teljeskörű tesztelést kell végezni az adott modulhoz. Úgy gondolom, hogy abban az esetben, ha csak egy darab alacsony prioritású hiba van az adott

a modulban, akkor a tesztelés szintje közepes, ha viszont tartozik új funkció vagy magas prioritású hiba az adott modulhoz, akkor a tesztelés szintje teljeskörű.

P11 **Nincsenek tesztesetek a regressziós tesztekhez.** A folyamat vizsgálata során arra jutottam, hogy a probléma megoldására a Sprintek során a szoftvertesztelőknek időt kell delegálni az adott alkalmazások tesztesetlistájának elkészítéséhez, majd a tesztesetek priorizálásra. Ahhoz, hogy ezek a feladatok a Sprintek során ne felejtődjenek el, javaslom, hogy az Azure DevOps rendszerben feladatok legyenek létrehozva az adott modulok teszteseteinek létrehozására vonatkozóan és a feladatokhoz A Sprint tervezés során kijelölésre kerüljenek a felelősök. Ugyanakkor korábban említettem, hogy eltérések vannak az adott tesztelő kollégák által alkalmazott technikákban, ezért a teszteset bázis kialakítása előtt javasolt létrehozni egy olyan szabályzatot, mely tartalmazza a teszteset létrehozás során alkalmazandó módszereket, technikákat, formai követelményeket stb. mindezt minta példákkal kiegészítve, ezzel biztosítva az egységes tesztbázis kiépítését.

5.3. Szoftvertesztelési folyamat fejlesztésére javasolt modell, fejlesztendő folyamatterületek és elvégzendő feladatokra tett ajánlások

A 4.2.2 fejezetben elvégzett szoftvertesztelési referencia modell alkalmasságának vizsgálata során kiderítettem, hogy a GSGroup Handyman részlegén egyértelműen **a TMMi modell alkalmazása javasolt.** A TMMi modell 98%-ban felel meg a szoftverfejlesztési modellel szemben támasztott követelményeknek, miközben a TPI Next csak 67,8%-ban.

A 4.2.3-as fejezetben a TMMi Lightning Scan Tool segítségével elvégzett felmérésem eredményéből látszik, hogy a Handyman részleg **szoftvertesztelési folyamata jelenleg az 1-es „Kezdő" érettségi szinten** helyezkedik el. A második érettségi szinten található folyamatterületek közül egyedül a „Teszt környezet" teljesíti a 85%-os teljesítettséget, a többi folyamatterület jelentős fejlesztésekre szorul. A felmérésem során megvizsgáltam a harmadik

érettségi szinten található folyamatterületeket is, a felmérés eredménye alapján egyik értéke sem éri el a „Teljesített” szintet. A szoftvertesztelési folyamat fejlesztését **a második érettségi szinten elhelyezkedő, nem teljesített folyamatterületek fejlesztésével** ajánlott elkezdeni a következő sorrendben: „Tesztelési politika és stratégia”, „Tesztelés tervezése”, „Tesztelés monitorozása és ellenőrzése”, „Teszt tervezés és végrehajtás”, mivel ezek a folyamatterületek egymásra épülnek. Ahhoz, hogy a jelenlegi szoftvertesztelési folyamat teljesítse a 2. érettségi szintet, a felméréseim alapján a következőkben megfogalmazott tevékenységeket kell elvégezni (az általam felsorolt sorrendben) a Handyman részlegen.

A „**Tesztelési politika és stratégia**” folyamatterületen elvégzett felmérésem eredményei alapján először létre kell hozni egy tesztelési politikát, mely összhangban van a szervezet általános üzleti politikájával. Mivel a Handyman részlegen belül a menedzsmentnek alacsony a rálátása a tesztelési folyamatokra és tesztelés céljaira, ezért véleményem szerint a tesztelési politika kidolgozása a menedzsment bevonásával elősegítené a szoftvertesztelés fontosságának elterjesztését menedzsment szinten, melyet kritikus szempontnak tartok a folyamatfejlesztés szempontjából, hiszen a vezetői hozzájárulás nélkül nem lehet kiépíteni megfelelő vállalati kultúrát, mely támogatja a folyamatok fejlesztését. A tesztelési politikának meg kell határoznia a tesztelési folyamat fejlesztésének céljait, melyeket majd a későbbiek során át lehet alakítani olyan teljesítmény mutatókká, mint például az adott verziók után visszaérkező ügyfél hibák száma. Ezután következő lépésként a tesztelési politika alapján ki kell alakítani a tesztelési stratégiát, mely tartalmazza az alkalmazandó tesztelési szinteket (például a rendszer tesztelés), a tesztszintekhez meghatározza a célokat (például 80%-os specifikáció lefedettség elérése), felelősségi köröket és a fő feladatok be és kilépési kritériumait (például mikor tekinthető „Kész” -nek az új funkciók tesztelés szempontjából). Emellett fontosnak tartom, hogy a tesztelési stratégia tartalmazza a tesztelési/teszteset tervezési technikákat, melyeket jelenleg a Handyman részlegen a gyakorlatban ad-hoc módon alkalmaznak. A tesztelési stratégia elkészítése során véleményem szerint figyelni kell az adott csapatok és termékek közötti eltérésekre és ezeket a tesztelési stratégiában külön-külön jelezni kell, vagy akár az is egy megoldás lehet, hogy egy általános szintű tesztstratégiát minden csapat egyedi jellemzői alapján személyre szabnak, így végül minden csapat rendelkezni fog egy tesztelési stratégiával,

mely illeszkedik az általános tesztelési stratégiához. Úgy gondolom, hogy egy jól elkészített tesztelési stratégia a kiindulópont jelentheti a hatékonyabb szoftvertesztelési folyamat kialakításához a Handyman részlegen.

A „**Tesztelés tervezése**” folyamatterületen elvégzett felmérésem alapján a fejlesztés során a TMMi modell szabályait követve határoztam meg a szükséges lépéseket. Első lépésként el kell végezni egy átfogó kutatást a termék, a szervezet, a követelmények és a fejlesztési folyamat kockázataival kapcsolatban az érintettek és a menedzsment bevonásával. A folyamatelemzések során szerzett nézőpontom alapján, úgy gondolom, hogy a kutatás gyakorlati megvalósításához olyan csoportokot kellene kialakítani, ahol például az Handyman Office vonatkozóan jelen vannak a legtapasztaltabb fejlesztők, tesztelők, a termék tulajdonos a termék tervező és a fejlesztési és kutatási részleg vezetője. A tesztelési stratégia és a kockázatértékelés eredménye alapján ki kell alakítani egy tesztelési megközelítést, mely meghatározza, hogy mely követelményeket (például nem funkcionális követelmények), milyen mértékben, hogyan és mikor kell tesztelni, itt szintén fontosnak tartom, hogy nem csak a tesztelő csapat vegyen részt a kialakításban, hanem az érintettek is. A tesztelési megközelítésben megfogalmazott tevékenységekhez költség, erőforrás, eredmények, illetve infrastruktúrával kapcsolatos információkat kell hozzárendelni, ezek későbbi gyakorlati adminisztrálása tapasztalataim alapján az Azure DevOps rendszerben megvalósítható, illetve a korábban adminisztrált adatokat ajánlott felhasználni használni az költség/idő/erőforrás értékek megfelelő becsléséhez. Ezek alapján létrehozható a formális Teszterv dokumentum, mely a későbbiek során majd alapot ad a tesztelési tevékenységek elvégzéséhez és ellenőrzéséhez.

A „**Tesztelés monitorozása és ellenőrzése**” folyamatterületen végzett elemzésem eredményei és a TMMi szabályai alapján a terület fejlesztéséhez első lépésként a tesztelési munkatermékek, feladatok, költségek és ütemezés állapotát össze kell vetni a teszttervben meghatározottokkal, a gyakorlati bevezetést könnyíti, hogy az Azure DevOps-ban történő adminisztrálás miatt nincs szükség külön adatgyűjtő adminisztrálási technikák bevezetésére. Mivel a felmérésem során kiderült, hogy jelenleg nincsenek termékminőségre vonatkozó mutatók, ezért a minőség értékeléséhez be kell vezetni olyan mutatókat, mint például a

termékkockázatok mérséklése, a talált hibák száma, vagy a teszt kilépési kritériumhoz viszonyított állapot. A tesztelés monitorozása sem valósul meg, ezért ki kell alakítani olyan technikákat, melyekkel adatokat lehet gyűjteni például a teszteseményekről, meg lehet vizsgálni a tervezett előrehaladás és termékminőség állapotát, ezeket az eredményeket a szabályok alapján a vizsgálati összefoglaló jelentésben kell dokumentálni, viszont az agilis fejlesztés miatt el lehet térni attól, hogy külön dokumentumot kelljen létrehozni. A monitorozás során szintén ajánlott az Azure DevOps automatikus diagram generálási funkciójának használata, ahol egyedi paraméterek meghatározásával generált diagramok segítségével megjeleníthetőek a vizsgált tulajdonságokra vonatkozó aktuális adatok. Abban esetben a vizsgálati összefoglaló és a terv között eltérések vannak, korrekciós intézkedéseket kell bevezetni, az intézkedések megvitatása véleményem szerint a napi Scrum megbeszélések keretében megtörténhet. Végül be kell vezetni a tesztelési projekt kockázatkezelését (jelenleg egyáltalán nem valósul meg), mely során a tesztelési tervet meghiúsító nagyobb problémák mielőbbi azonosítása és megoldása valósul meg, az azonosított problémák megoldásainak kitalálására javasolt ötletbörzék megtartása a Scrum csapat tagjaival és az érintettekkel.

A „**Teszt tervezés és végrehajtás**” folyamatterület felmérése során arra jutottam, hogy először meg kell határozni az alkalmazandó teszttervezési technikákat és eszközöket - ez a tevékenység véleményem szerint a tesztelési stratégia létrehozása során, vagy stratégia csapatra való személyre szabása során is megtörténhet. Ezután úgy gondolom, hogy a következő lépés a teszteset bázisok létrehozása, vagy már meglévő tesztbázisok esetén a felülvizsgálat és frissítés, majd meg kell határozni a hozzájuk tartozó tesztkörülményeket. Ezek adminisztrálását az Azure DevOps rendszer Teszt tervezés modul konfigurációkezelés és teszteset kezelés alfunkciója támogatja. Miután elkészültek a tesztesetek, elkezdhető a teszteljárások kialakítása, melyben a tesztelési műveletek végrehajtási sorrendben vannak felsorolva (szintén konfigurálható az Azure DevOps rendszerben). A TMMi szabályai alapján a létrehozott teszteseteket a teszteljárásokat alkalmazva kell végrehajtania a szoftvertesztelő csapatnak. A folyamatterületen belül a végrehajtáshoz kapcsolódó hibakezelést már nem kell kialakítani, mivel a felméréseim alapján kiderült, hogy ez jelenleg is megvalósul a szervezeten belül.

6. Összefoglalás

Diplomamunkám során elvégeztem egy informatikai cég vállaltirányítási szoftvert készítő részlegén az agilis szoftverfejlesztési és szoftvertesztelési folyamatok elemzését, az elemzés során azonosított problémák megoldására (például a Scrum módszertan elemeinek nem megfelelő alkalmazására) javaslatokat fogalmaztam meg.

A szoftvertesztelési folyamatok fejlesztéséhez számos modell közül lehet választani. A megfelelő szoftvertesztelési folyamat fejlesztési modell kiválasztásánál összegyűjtöttem a leggyakrabban alkalmazott TMMi (Test Maturity Model integration) és TPI Next (Test Process Improvement Next) modellek tulajdonságait. A megfelelő modell kiválasztásához a komplex tulajdonságok összemérésének módszerét alkalmazva megállapítottam, hogy a TMMi folyamat fejlesztési referencia modell alkalmazható jobban az adott részleg szoftvertesztelési folyamatának javítására. A TMMi Foundation által készített eszköz segítségével felmértem a szoftvertesztelési folyamat aktuális érettségi szintjét. A felmérés alapján a vizsgált szoftvertesztelési folyamat érettségi szintje a legalacsonyabb „Kezdő” szinten van. Elemzést végeztem a második érettségi szinten található folyamatterületek állapotáról és a nem megfelelően teljesített TMMi folyamatterületekhez meghatároztam az adott szint teljesítéséhez elvégzendő tevékenységeket.

A szoftverfejlesztők körében végzett felmérésen keresztül megvizsgáltam, hogy a részleg által fejlesztett termékben, mekkora százalékban jelennek meg a leggyakoribb és legveszélyesebb szoftversérülékenységek. A felmérés során kiderült, hogy a vizsgált 25 sérülékenységből és szoftvergyengeségből 16 darab megjelenik a termékben, vagyis bebizonyosodott, hogy szükség van olyan intézkedések bevezetésére, melyek csökkentik az ebből adódó kockázatokat. Javaslatokat fogalmaztam meg a termékben található sérülékenységek számának csökkentésére, melyek közül a megoldások költségének és hatékonyságának vizsgálata után a szoftvertesztelő csapat informatika biztonsági továbbképzését tartom a legjobb választásnak.

7. Summary

During my diploma thesis, I analysed the agile software development and software testing processes of an IT company's enterprise management software development department, and I formulated proposals to solve the problems identified during my analysis (for example, the inappropriate application of the Scrum methodology).

There are several models available for developing software testing processes. To choose the right software testing process development model, I have collected the characteristics of the most commonly used TMMi (Test Maturity Model integration) and TPI Next (Test Process Improvement Next) models. To select the appropriate model, I used the Comparison of Complex Properties method to determine that the TMMi process improvement reference model is more applicable to improving the software testing process in the given department. I used a tool created by the TMMi Foundation to assess the current maturity level of the software testing process. Based on the survey, the maturity level of the software testing process is at the lowest "Initial" level. I have analysed the status of the process areas at the second maturity level and for the TMMi process areas that are not satisfactorily completed, I have determined the activities to be performed to achieve the level.

Through a survey of software developers, I examined the percentage of the most common and dangerous software vulnerabilities in the products developed by the department. The survey showed that 16 out of the 25 vulnerabilities and software weaknesses examined appear in the product, demonstrating the need to implement measures to reduce the risks. I have formulated recommendations to reduce the number of vulnerabilities in the product. I found that IT security training for the software testing team is the best option after examining the cost and effectiveness of my suggested solutions.

Irodalomjegyzék

[1] *K. Dr. Balla, „Szoftveripar sajátosságai,” in Szoftveripar sajátosságai, Verlag Dashöfer Szakkiadó Kft, 2008.*

[2] *S. Nagarajah, „Software Development vs. Manufacturing,” [Online]. Available: https://www.globalapptesting.com/blog/is-software-development-like-manufacturing. [Hozzáférés dátuma: 05 04 2023].*

[3] *M. Sami, „From manufacturing to Software development — what can we learn!,” 06 11 2021. [Online]. Available: https://medium.com/@melsatar/from-manufacturing-to-software-development-what-can-we-learn-4dfe8832c376. [Hozzáférés dátuma: 05 04 2023].*

[4] *H. Krasner, „The Cost of Poor Quality Software in the US: A 2018 Report,” Consortium for IT Software Quality, 2018.*

[5] *Z. Bíró, „A SZOFTVERFEJLESZTÉS MINŐSÉGBIZTOSÍTÁSA,” Fiatal Műszakiak Tudományos ülésszaka, Kolozsvár, 1997.*

[6] *Materfield Oktatóköpont, „Szoftvertesztelés helyzete Magyarországon 2018,” Materfield Oktatóközpont, Magyarország, 2018.*

[7] *V. Medina, Minőségmenedzsment (Oktatási segédlet), Gödöllő: Szent István egyetem, 2019.*

[8] *C. M. U. CERT, „Top 10 CERT/CC Blog Posts on Vulnerabilities and SSL Tools,” 2015.*

[9] *H. Krasner, „The cost of Poor Software Quality in the US: A 2022 report - From problem to solutions,” CISQ (Consortium for Information &Software Quality), 2022.*

[10] *D. Radigan, „Escaping the black hole of technical debt,” [Online]. Available: https://www.atlassian.com/agile/software-development/technical-debt. [Hozzáférés dátuma: 23 04 2023].*

[11] *Common Weakness Enumeration, „2022 CWE Top 25 Most Dangerous Software Weaknesses,” 2022. [Online]. Available: https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html. [Hozzáférés dátuma: 04 02 2022].*

[12] *A. Dr. Horváth, „A szoftver sérülékenységek kihasználási módozatai –Informatikai*

támadások, támadók és biztonság 2013-2016.," INFOTA, 2016.

[13] *D. R. Muhammad , „Importance of Secure Software Development Processes and Tools for Developers," Taylor's University, 2020.*

[14] *SQA Logic, „How does cybersecurity and software quality assurance synergize?," [Online]. Available: https://sqalogic.com/how-does-cybersecurity-and-software-quality-assurance-synergize/. [Hozzáférés dátuma: 04 02 2022].*

[15] *Dynatrace, „2022 CISO Research Report," Dynatrace, 2022.*

[16] *Jinjin Li, „Agile Software Development," Technische Universitt Berlin, Berlin, Germany.*

[17] *M. Beedle, A. Bennekum van, C. Alistar, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas és B. Kent, „Kiáltvány az agilis szoftverfejlesztésért," 2001. [Online]. Available: http://agilemanifesto.org/iso/hu/manifesto.html. [Hozzáférés dátuma: 2023].*

[18] *K. Schwaber és J. Sutherland, A Scrum Útmutató, Scrum.Org & Scrum Inc., 2014.*

[19] *C. Drumond, „Scrum - A guide to scrum: what it is, how it works, and how to start," [Online]. Available: https://www.atlassian.com/agile/scrum. [Hozzáférés dátuma: 20 04 2023].*

[20] *P. Dr. Benedek, Szerző, Folyamatábra készítés a gyakorlatban. [Performance]. BME GTK.*

[21] *T. QA, „Software Testing Process Improvement Models," [Online]. Available: https://tryqa.com/software-testing-process-improvement-models-tmmi-tpi-next-ctp-step/. [Hozzáférés dátuma: 07 04 2022].*

[22] *Z. G. Balogh, Á. Beszédes , T. Gergely, Leitold Ferenc, A. Pethő és G. L. Szőke, Éves továbbképzés az elektronikus információs rendszer biztonságáért felelős személy számára, Budapest: NKE, 2014.*

[23] *TMMi Foundation, „Model Aims and Objectives," [Online]. Available: https://www.tmmi.org/model-aims-and-objectives/. [Hozzáférés dátuma: 08 04 2023].*

[24] *S. Alone és G. Kerstin, „Evaluation of Test Process Improvement approaches," University of Gothenburg, Göteborg, Svédország, 2013.*

[25] *TMMi Foundation, „TMMi modell,” [Online]. Available: https://www.tmmi.org/tmmi-model/. [Hozzáférés dátuma: 07 04 2023].*

[26] *M. M. Forró, „Tesztelési folyamatok fejlesztése multimodelles megközelítéssel a LuftHansa Systems Hungária Kft.-nél,” Budapest, 2016.*

[27] *v. E. Veenendaal, „Test Maturity Model integration (TMMi®) Guidelines for Test Process Improvement,” TMMi Foundation;, United Kingdom.*

[28] *T. Aaltio, „Test Process Improvement with TPI Next,” in Sogeti Finland Oy, Lvov, 2013.*

[29] *A. Ewijk, B. Linker, M. Osterwijk, B. Visser, G. Vries, L. Wilhemus és R. Mardelis, TPI NEXT - Business Driven Test Process Improvement, Sogeti, 2013.*

[30] *ISTBQ, Certified Tested Expert Level Syllabus . Improving Testing Process, ISTQB, 2010.*

[31] *M. Dr. Daróczi , Szerző, Az ezközkiválasztás lehetséges módszerei. [Performance]. MATE, 2021/2022 őszi félév.*

[32] *Nemzeti Kibervédelmi Intézet, „Robothálózat (botnet),” 09 03 2020. [Online]. Available: https://nki.gov.hu/it-biztonsag/tudastar/robothalozat-botnet-2/.*

[33] *J. Szefner, „Tesztkörnyezet specifikáció,” 12 07 2010. [Online]. Available: https://tesztelesagyakorlatban.hu/tesztkornyezet-specifikacio/. [Hozzáférés dátuma: 08 04 2023].*

Ábrajegyzék

Táblázatjegyzék

MATE
MAGYAR AGRÁR- ÉS ÉLETTUDOMÁNYI EGYETEM

Szent István Campus, Gödöllő
Cím: 2100 Gödöllő, Páter Károly utca 1.
Tel.: +36-28/522-000
Honlap: https://godollo.uni-mate.hu

NYILATKOZAT

Alulírott SZTARACSEK LILLA, a Magyar Agrár- és Élettudományi Egyetem, SZENT ISTVÁN Campus, MŰSZAKI MENEDZSER szak nappali/levelező* tagozat végzős hallgatója nyilatkozom, hogy a dolgozat saját munkám, melynek elkészítése során a felhasznált irodalmat korrekt módon, a jogi és etikai szabályok betartásával kezeltem. Hozzájárulok ahhoz, hogy Záródolgozatom/Szakdolgozatom/Diplomadolgozatom egyoldalas összefoglalója felkerüljön az Egyetem honlapjára és hogy a digitális verzióban (pdf formátumban) leadott dolgozatom elérhető legyen a témát vezető Tanszéken/Intézetben, illetve az Egyetem központi nyilvántartásában, a jogi és etikai szabályok teljes körű betartása mellett.

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem*

Kelt: 2023 év 04 hó 28 nap

Sztaracsek Lilla
Hallgató

NYILATKOZAT

A dolgozat készítőjének konzulense nyilatkozom arról, hogy a Záródolgozatot/Szakdolgozatot/Diplomadolgozatot áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A Záródolgozatot/Szakdolgozatot/Diplomadolgozatot záróvizsgán történő védésre javaslom / nem javaslom*.

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem*

Kelt: 2023 év 04 hó 30 nap

Belső konzulens

*Kérjük a megfelelőt aláhúzni!

NYILATKOZAT

a diplomadolgozat[1] nyilvános hozzáféréséről és eredetiségéről

A hallgató neve: SZTARACSEK LILLA

A Hallgató Neptun kódja: I57UWZ

A dolgozat címe: MINŐSÉGBIZTOSÍTÁSI FOLYAMATOK HATÉKONYSÁGÁNAK VIZSGÁLATA A SZÉTTELEPÍTÉSBEN

A megjelenés éve: 2023

A konzulens tanszék neve: MŰSZAKI MENEDZSMENT TANSZÉK

Kijelentem, hogy az általam benyújtott diplomadolgozat[2] egyéni, eredeti jellegű, saját szellemi alkotásom. Azon részeket, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem, s az irodalomjegyzékben szerepeltettem.

Ha a fenti nyilatkozattal valótlant állítottam, tudomásul veszem, hogy a Záróvizsga-bizottság a záróvizsgából kizár és a záróvizsgát csak új dolgozat készítése után tehetek.

A leadott dolgozat, mely PDF dokumentum, szerkesztését nem, megtekintését és nyomtatását engedélyezem.

Tudomásul veszem, hogy az általam készített dolgozatra, mint szellemi alkotás felhasználására, hasznosítására a Magyar Agrár- és Élettudományi Egyetem mindenkori szellemitulajdon-kezelési szabályzatában megfogalmazottak érvényesek.

Tudomásul veszem, hogy dolgozatom elektronikus változata feltöltésre kerül a Magyar Agrár- és Élettudományi Egyetem könyvtári repozitori rendszerébe.

Kelt: 2023 év 05 hó 02 nap

Sztaracsek Lilla
Hallgató aláírása

[1] A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.
[2] A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.