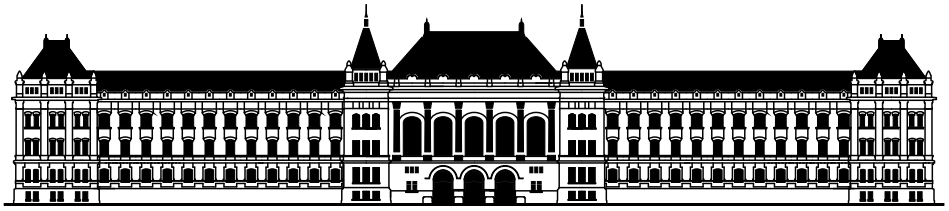

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM

Modellek és algoritmusok agilis szoftvertervezéshez és ütemezéshez

írta

Szőke Ákos

Konzulens: Prof. Pataricza András, DSc.

Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

March, 2014

BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Modellek és algoritmusok agilis szoftvertervezéshez és ütemezéshez

PhD. disszertáció összefoglaló

írta Szőke Ákos

A kutatásam középpontjában a szoftver kibocsátások és az iterációk tervezése helyezkedik el, amelyek a lokális és az elosztott agilis szoftverfejlesztési szervezetek fejlesztéseinek koordinálását támogatják. A kutatásaim legfőbb eredménye egy integrált agilis tervezési megközelítés (Integrated Agile Planning Approach – IAPA) megtervezése, alkalmazása és validációja. A megközelítés a fejlesztési koordinátorokat támogatja a szoftverfejlesztés összetettségének és dinamizmusának leküzdésében fél-automatikus kibocsátási és iterációs tervek megvalósításával, amelyek általam kidolgozott fogalmi modellekre, optimalizációs modellekre és algoritmusokra épülnek.

Az ipari szoftverfejlesztés egy meglehetősen összetett és dinamikus folyamat. A szoftverfejlesztő cégek sikeressége a fejlesztési folyamatok hatékonyságától és eredményességétől függ, amelyben a fejlesztések koordinálása központi szerepet tölt be. A koordináció gyakran olyan döntési problémába ütközik, amelynek az a célja, hogy meghatározza a következő szoftverkibocsátásokba (release) kerülő funkciók (feature) körét. Ennek a döntési problémának az eredménye a kibocsátási tervben (release plan) valósul meg. Ez a terv leírást tartalmaz arról, hogy mely funkció melyik fejlesztési ütemben legyen implementálva annak érdekében, hogy a kibocsátások maximális üzleti értéket biztosítsanak. Amint a funkciók köre meghatározottá válik, a koordináció egy másik döntési problémával szembesül, amelynek célja annak a meghatározása, hogy a funkciók megvalósítása milyen módon történjen. Ennek a döntéshozatalnak az eredménye az iteráció tervben valósul meg, amelyben a funkciók megvalósítási feladataihoz erőforrásokat allokálunk a megadott megszorítások figyelembe vételével.

A bemutatott tézisek alapvetően különböznek a jelenlegi – főként intuitív – módszerektől a benne rejlő információ-egységesítő és matematikai optimalizációs megközelítése következtében. A szoftvermérnöki és menedzseri információ integrációjával a bemutatott algoritmikus optimalizációs módszerek könnyen megbirkóznak az összetett döntési helyzetekkel, jobb áttekintést adnak a fejlesztésre vonatkozóan, ezen kívül lehetővé teszik, hogy az üzleti prioritásokban bekövetkező változásokra új tervek előállításával gyorsan reagálni lehessen.

Az IAPA megközelítés a Pythia kutatási projekt keretében valósult meg, amelyet részben egy GVOP-s pályázat is támogatott.

Az IAPA megközelítés három prototípusként lett implementálva, amelyeket egy szoftverfejlesztő cégnél sikeresen napi szinten alkalmazzák évek óta. A megközelítés hatékonyságát és eredményességét ipari esettanulmányok és ipari adatokon alapuló szimulációk támasztják alá.

Contents

Kivonat **ii**

Köszönetnyilvánítás **iv**

1 Bevezetés **v**
1.1 Motiváció és a kutatás relevanciája . . . v
1.2 A kutatásom célkitűzései . . . vi
1.3 Kutatási módszertan . . . vii

2 Új tudományos eredmények **viii**
2.1 Tézis 1 . . . ix
2.1.1 Agilis kibocsátás tervezésének matematikailag precíz megfogalmazása . . . x
2.1.2 A Tézis 1 gyakorlati validációja . . . x
2.1.3 Implementáció a tapasztalati validációhoz . . . xii
2.2 Tézis 2 . . . xii
2.2.1 Agilis iteráció tervezés matematikailag precíz megfogalmazása . . . xiii
2.2.2 A Tézis 2 gyakorlati validációja . . . xiv
2.2.3 Implementáció a tapasztalati validációhoz . . . xvi
2.3 Tézis 3 . . . xvi
2.3.1 Agilis kibocsátás tervezés elosztott kiterjesztésének matematikailag precíz megfogalmazása . . . xvi
2.3.2 A Tézis 3 gyakorlati validációja . . . xvii
2.3.3 Implementáció a tapasztalati validációhoz . . . xviii

3 Az eredmények gyakorlati alkalmazása **xviii**

4 Publikációk jegyzéke **xx**

5 Hivatkozások **xxiii**

6 További hivatkozások **xxiv**

Köszönetnyilvánítás

Elsőként konzulensemnek, Dr. Pataricza Andrásnak, szeretném megköszönni az elmúlt években nyújtott segítséget. Ez idő alatt sokat tanultam és meg vagyok győződve arról, hogy ez a tudás a jövőben is hasznomra lesz. Ő ajánlotta a kutatás kezdeti irányát, amely számos párhuzamos irányba is elvezetett, ezek eredményei juttattak a jelen disszertációhoz.

A kutatásaimhoz sok támogatást kaptam Dr. Strausz Györgytől is, aki a munkahelyemen – a Multilogic Kft-nél, ahol több, mint 10 éve projektvezetőként és technológiai vezetőként dolgozom – egyben kollégám is. Kapcsolatunk a PhD fokozatszerzés előttire vezethető vissza. Beszélgetéseink a disszertáció készítésében sokat segítettek.

Kollégáimnak a Multilogic Kft-nél nagyon hálás vagyok az inspiráló környezetért. Különböző témákban való beszélgetéseink sokat segítettek a kutatásaimban. Szintén köszönetemet fejezem ki Dr. Balogh Andrásnak, Dr. Ráth Istvánnak (BME MIT FTSRG/OptXware), Millinghoffer Andrásnak, Dr. Antal Péternek (BME MIT) és Kiss Gábornak (Multilogic) a gyümölcsöző együttműködésért a Pythia projektben, amelynek eredménye a PYTHIA PROJECT PLANNER™ prototípus.

Köszönetemet fejezem ki Károlyi Gábornak (Magyar Posta IT Fejlesztési részlegének igazgató helyettese) és a csapatának, Udvardy Magdolnának (APEH IT Stratégiai részlegének igazgatója), Dr. Dolgos Sándornak (BBraun Medical Hungary IT Fejlesztési részlegének vezetője), Halácsy Péternek (Prezi.com alapítója és technológiai vezetője) és Prolan Zrt. kollégáinak a saját területeiken nyújtott szakmai támogatásért. A közös projekteken való együttműködés nagyon érdekes és értékes volt számomra. Az inspiráló beszélgetések jelentős előrehaladást tettek számomra lehetővé a rövid együttműködések ellenére is.

Szintén nagyon hálás vagyok Dr. Marjan van den Akker-nek (Utrecht-i egyetem), hogy elérhetővé tette számomra adatait az algoritmusaim validálásához, Dr. Zornitza Racheva-nak (Twente-i egyetem), Dr. Vladimir Mandic-nak és Dr. Kari Liukkunen (Oulu-i egyetem) és Dr. Val Casey-nek (Dundalk-i egyetem) a gyümölcsöző beszélgetésekért, együttműködésért, bíztatásért és támogatásért.

Értékes visszajelzést és inspirációt kaptam számos agilis szoftverfejlesztésben jártas gyakorlati szakembertől is, akikkel a Budapest Agile Meetup előadás sorozaton találkoztam, melyeket az Agile Alliance Hungary szervezett meg.

A munkámat részben a GVOP (GVOP-3.3.3-05/1.-2005-05-0046/3.0) K+F pályázati forrás támogatta, amelynek kivitelezésében a Multilogic, BME MIT tanszéke és az OptXware működött közre.

Végül, jelen disszertációt azoknak szeretném ajánlani, akik megtettek mindent, hogy biztos hátteret szolgáltassanak nekem – szeretetet, bíztatást és támogatást kaptam tőlük: Klára (feleségem), Klárika és Zita (lányaim), Éva és Sándor (szüleim). Támogatásuk nélkül, ez a munka nem készült volna el.

1 Bevezetés

A modern szoftverfejlesztési megközelítések (pl. RUP, XP, FDD, Scrum [29, 30, 31, 32]) *iteratív fejlesztési folyamaton* alapulnak, amelyek különböző tényezők (pl. alkalmazási terület, fejlesztési környezet) figyelembe vételével különböző legjobb gyakorlatokat valósítanak meg. A különféle iteratív folyamatok közös jellemzője a lépcsőzetes megközelítés, amely ciklikus fejlesztési folyamatot eredményez. Ez a fejlesztési életciklusban különböző szoftvertermékverziók átadási folyamatát határozza meg. A folyamat minden ciklusban egy szoftververzió kezdeti átadási tervével indul és a szoftver termék megrendelőnek történő átadásával végződik. Az iteratív fejlesztés központi eleme a szoftververziók ütemezésének harmonizációja a megrendelő által igényelt funkciók prioritásainak figyelembe vételével. A folyamat az ismétlődő átadások során a szoftvertermékek egy fokozatosan fejlődő részhalmazait állítja elő, amelyek a megrendelő prioritásai alapján fokozatosan fedik le az igényeket. Az átadások során szerzett felhasználói tapasztalatok és visszajelzések hatást gyakorolnak a fejlesztésre. A megközelítés alapelve megrendelői nézőpontból származtatható: a cél egyedi részproblémákra adott megoldások átadása az előre definiált idő, terjedelem, minőségi és erőforrás megszorítások figyelembe vételével [30, 32].

Az 1990-es évek végén számos új módszertan a figyelem középpontjába került. Mindegyik különböző új és megváltoztatott régi elvek kombinációit hirdette. Azonban, mindegyik 1) a fejlesztők és üzleti szakértők szoros együttműködésének; 2) a fejlesztők és megrendelők személyes kommunikációjának (mivel hatékonyabb, mint az írásos forma); 3) az új üzleti értéket képviselő funkciók gyakori átadásának; és 4) az erősen kooperatív és önszervező csapatoknak a szükségességét hangsúlyozta [33]. Ezek a vélemények formálták az agilis szoftverfejlesztés elveit.

Az agilis megközelítés a tradicionális terv-alapú (értsd merev) módszertanok a '90-es évek végi sikertelenségeire született válaszul, annak érdekében, hogy támogassa a megrendelői igények gyakori változásainak megfelelő fejlesztési folyamat adaptációját. A tradicionális módszerek 'racionalizált és mérnöki' megközelítésként foghatóak fel, mivel azt állítják, hogy a problémák részleteikben specifikálhatóak, a problémákra előre látható optimális megoldások adhatóak, ezért prediktív tervezési stratégiát követnek. Ez a megközelítés részletes tervezést, kodifikált folyamatokat és szigorú újrafelhasználást ír elő annak érdekében, hogy a fejlesztés hatékony és kiszámítható legyen [34, 35]. A terv-alapú megközelítéssel szemben az agilis folyamatok a kiszámíthatatlanságra való válaszadást tűzik ki célul úgy, hogy 'az emberekre és a kreativitásukra helyezi a hangsúlyt a folyamatok helyett' [34, 35]. Az agilis módszerek tipikusan adaptív megközelítést használnak, amelyben háromszintű tervezést hajtanak végre gyakori visszacsatolásokkal: egy nagy szemcsézettségű hosszú időtartamra vonatkozót (kibocsátás), normál szemcsézettségű rövid időtartamra vonatkozót (iteráció) és finom szemcsézettségű napi tervet [36, 37, 38]. Minden egyes szint a saját és a fölötte levő szint céljainak megvalósulásáért felelős.

1.1 Motiváció és a kutatás relevanciája

A szervezetek számára az agilis folyamatok számos előnyt nyújtanak, ideértve a gyorsabb befektetés megtérülést (Return of Investment – ROI), a magasabb minőséget, a magasabb megrendelői elégedettséget [39]. Ezek a folyamatok megalapozott módszertani támogatással nem rendelkeznek az agilis kibocsátás- és iterációtervezésre vonatkozóan – ellentétben a hagyományos, terv-alapú megközelítésekhez képest. A korábban hivatkozott felmérés [39] arra mutatott rá, hogy a 26 legproblémásabb tényező közül a kibocsátás tervezés az ötödik, az iterációtervezés pedig a második. Ezek mellett a felmérés arra is felhívja a figyelmet, hogy a legtöbbször hivatkozott problémák közül 13 az agilis módszerek cégen

belüli adoptálására vonatkozik. Az öt legfontosabb adoptálási probléma közül három a következő volt: 1) a menedzsment kontrolljának elvesztése (36%), 2) a folyamat elején való tervezés hiánya (33%) és 3) a kiszámíthatóság hiánya (27%). Ezek a hiányosságok szorosan összefüggnek a jelenlegi agilis tervezés *informális* gyakorlatával. Ezek a kritikák a megalapozott agilis tervezési módszerek fontosságát hangsúlyozzák, amely hiányosság az agilis módszerek újdonságával magyarázható. Jelen kutatás célja, hogy módszertani támogatást adjon az agilis kibocsátás- és iterációtervezéshez mind az egyazon helyen, mind az elosztott környezetben történő fejlesztési szituációkban.

Az ipari projektek során is – projektvezetőként és CTO-ként dolgozom 10 éve – a korábban említett problémákkal és nehézségekkel találkozom. Ezek a személyes tapasztalatok is megerősítették bennem az agilis kibocsátás- és iterációtervezés hiányosságait, hiszen ezen hiányosságokkal a projekt szponzorok (mind a fejlesztői, mind a megrendelői) a projekt finanszírozásáról nem győzhetőek meg. Ezért ezek a hiányosságokat a gyakorlatomban intuitív módon kézzel történő projekt tervezéssel hidaltuk át. Azonban az ilyen módon való tervezés szuboptimális és néha ellentmondásos projekt terveket eredményeznek.

1.2 A kutatásom célkitűzései

Egyre jobban megfigyelhető az az igény, hogy a fejlesztési költségeket és a piacra jutás idejét csökkentsük, valamint, hogy a fejlesztés minőségét növeljük. Ezek az igények az automatizált szoftvermérnöki módszerek és eszközök megjelenését katalizálják mind a projekt tervezésére és ütemezésre, mind a döntéstámogatásra vonatkozóan [40]. Bár az agilis tervezés manuális támogatására számos törekvés megfigyelhető [41, 38], algoritmikus megoldás jelenleg nem létezik az agilis megközelítés relatív újdonsága miatt.

A tradicionális és az agilis szoftver folyamat végrehajtási különbözőségeire tekintettel az agilis kibocsátás- és iteráció tervezésre vonatkozó döntéstámogatás projekttervezési aspektusa *fontos és új* kutatási terület, hiszen az informális tervezési és ütemezési megközelítések nagyon munkaigényesek és sok hibázási lehetőséget foglalnak magukba. Ezen kívül, a bemeneti adatokban való kismértékű módosítás (pl. követelmények, megszorítások és célok) a teljes terv újra elkészítését követelheti meg – a bemeneti adatokban történő folyamatos változás pedig az agilis környezetek alapvető karakterisztikája.

A kutatásom célja a fenti problémák alapján a kibocsátás- és iteráció szintű tervezés döntései támogatásának vizsgálata volt. A kutatásom célkitűzései a következők voltak:

RO1 *Az agilis szoftverfejlesztés tervezési produktivitásának növelése* a fejlesztési folyamat különböző fázisaihoz kapcsolódó interaktív és fél-automatikus módszerek bevezetésével.

RO2 *Az agilis szoftverfejlesztés tervezési kognitív komplexitásának csökkentése* úgy, hogy az összetett döntési szituációkat matematikai modellekkel formularizáljuk és oldjuk meg.

RO3 *Az agilis szoftverfejlesztés tervezési minőségének növelése* azzal, hogy a legfontosabb tervezési faktorokat (pl. dependendenciák, kapacitások) figyelembe véve a kockázati faktorokat csökkentsük.

RO4 *Az agilis szoftverfejlesztés tervezési döntéseit támogatjuk* azzal, hogy a specifikus projekt kontextusok figyelembe vételével a lehető legjobb projekttervet készítjük el, amelyben figyelembe vesszük a megrendelők és felhasználók visszajelzéseit a kapacitások, megszorítások és prioritások változtatásával.

RO5 *Az elosztott agilis szoftverfejlesztési csapatok kommunikációjának és koordinációjának növelése* azzal, hogy fél-automatikus módszereket vezetünk be a feladatoknak az elosztott csapatokhoz való allokálásához.

1.3 Kutatási módszertan

A bemutatott célkitűzések meghatározták a kutatásom irányát. A kutatás első lépése az agilis projekttervezés tervezési terének feltérképezése volt. Ennek eredményeképpen megalkottam egy konzisztens, ontológia-alapú információs modellt, amely az agilis projekttervezés tervezési terének komponenseit specifikálja. Ez a modell magába foglalja a főbb fogalmakat és közöttük lévő kapcsolatokat.

Következő lépésként az agilis projekttervezést három szintjét három különböző kombinatorikus optimalizációs problémaként formularizáltam, annak érdekében, hogy az adott szinten a döntéstámogatás hatékonyságát növeljem. A kidolgozott megoldásom fontos újdonsága a *matematikai precizitású probléma megfogalmazás*.

Harmadik lépésként a megalkotott optimalizációs problémákat megoldó algoritmusokat alkottam. Az algoritmusokat a PROPAS^{TM} (Project Planning and Scheduling) Matlab toolbox-om részeként implementáltam. A toolbox komponenseit az MIT licensz[1] alatt publikáltam[2], amely lehetővé teszi a komponensek szabad hozzáférését, felhasználását és validációját.

Utolsó lépésként az agilis projekttervezési módszereim validálását végeztem el. A tapasztalati validációhoz kifejlesztettem egy MS SharePoint alapú eszközt – nevezetesen SERPA^{TM} – a megalkotott projekttervezési fogalmi modell alapján. Ezt az eszközt a validációhoz szükséges valós fejlesztésekből származó adatgyűjtéshez használtam. A megalkotott eszközt egy szoftverfejlesztő cégnél évek óta napi szinten használják projektek tervezéséhez és monitorozásához.

Két validáció típust használtam, hogy a téziseimet több perspektívából elemezzem, ellenőrizzem és megerősítéssem – ezt a megközelítést *háromszögelésnek* nevezik. Az egyik típus a valós életbeli vizsgálat (kis számú valós kontextus), a másik típus az ezt kiegészítő laboratóriumi vizsgálat (nagy számú mesterséges kontextus) [42].

A valós életbeli vizsgálat lehetővé tette, hogy a téziseimet valós körülmények között egy szoftverfejlesztő cégnél validáljam. Bár minden agilis fejlesztési folyamat megvalósítása különböző, a vizsgált szoftverfejesztő cégnél történt implementáció tipikusnak tekinthető.

Másrészről a laboratóriumi vizsgálat lehetővé tette, hogy a valós életbeli vizsgálat eredményeit általánosítsuk nagyszámú reprezentatív problémákon történő vizsgálattal (120, 360 és 480 különböző eset). Annak érdekében hogy tipikus agilis fejlesztési szituációkat modellezhessek a generált adathalmazokat körültekintően állítottam elő: mind a *probléma komplexitás faktorait* (pl. funkciók, csapatok mérete, temporális megszorítások), mind a *környezet faktorait* (pl. elosztottság) figyelembe vettem, amelyek az agilis tervezés paraméterei. Ezen faktorok értékeit használtam fel a laboratóriumi vizsgálat (szimuláció) bemeneteiként.

A probléma komplexitás faktorok (paraméterek) értékeinek meghatározásához irodalom kutatást ([43, 44, 45, 46, 47]), felméréseket ([37, 39, 48, 49, 50]), agilis szakemberekkel való beszélgetéseket (különösen az Agile Hungary meetups[3]) és saját tapasztalatokat vettem alapul.

A két különböző validáció típust három jól ismert tudományos módszer segítségével végeztem el:

[1] The MIT licence http://opensource.org/licenses/MIT
[2] https://bitbucket.org/aszoke/matlab vagy http://www.kese.hu/
[3] http://www.meetup.com/AgileHungary/

1. **Post mortem analízis**: valós környezetbeli szoftverfejlesztés adataiból reprezentatív adathalmazt választottam ki a manuális és az algoritmikus tervezési módszerem összehasonlításához. A vizsgálatok célja az volt, hogy a két módszer kimenetét megvizsgáljuk úgy, hogy a manuálissal megegyező bemeneti változókat használjunk az algoritmikus módszernél is.
2. **Esettanulmány**: egy valós életbeli szoftverfejlesztési pilot projektben az algoritmikus megközelítésemet alkalmaztam a manuális tervezéssel párhuzamosan azért, hogy a két különböző megközelítést összehasonlíthassam. Az esettanulmányhoz egy UML alapú integrált ütemező alkalmazás is elkészült. A PYTHIA PROJECT PLANNER™ prototípus az Eclipse platformon UML2 technológiára épülve készült el.
3. **Szimuláció**: mesterségesen generált nagyszámú reprezentatív esetet alkottam a tervezés minőségének mérésére vonatkozóan. Ezzel a módszerrel a megoldás performanciáját és az általa szolgáltatott eredmények minőségét is vizsgálhatjuk, valamint a különböző agilis módszerek alkalmazásából eredő ingadozásokat is kiszűrhetjük.

A disszertációm 7. fejezetének 6.1-6.3-as táblázatai foglalják össze a kutatási módszertant: melyek voltak az azonosított problémák, az általam szolgáltatott megoldások, a validálció eszközei és eredményei.

2 Új tudományos eredmények

Jelen tézisek legfőbb eredménye az integrált agilis tervezési megközelítés (Integrated Agile Planning Approach – IAPA) megtervezése, implementálása és validálása. Az IAPA megközelítés a fejlesztés koordinátoroknak a komplex és dinamikus szoftver kibocsátás- és iteráció tervezési munkáját matematikai modellekkel és algoritmusokkal támogatja. A bemutatott megközelítés a jelenlegi megközelítéshez képest az alábbi aspektusokban tér el:

1. *A bemutatott megközelítés az **agilis tervezési és ütemezési problémák matematikai precizitású megfogalmazására** ad megoldást. A probléma megfogalmazás egy menedzseri (erőforrás és temporális) és szoftvermérnöki (követelmények leszállítása, hibajavítások) adatok információ fúzióját adja, annak érdekében, hogy egy közös, konzisztens információtárat szolgáltasson mind a fejlesztők, mind a koordinátorok részére. Ez a konzisztens tár mind az egyazon helyen, mind az elosztottan működő szoftverprojektek kibocsátás- és iteráció tervezéséhez szükséges adatait képes tárolni. Ez az információtár egy biztos hátteret ad az ütemezéssel kapcsolatos döntések meghozatalához.*
2. *A bemutatott megközelítés **matematikai optimalizáción** alapul, amely az előbbi közös információ táron **alapulva képes fél-automatikus tervgenerálást végezni**. A megalkotott algoritmusok komplex tervezési szituációkban is képesek döntéstámogatást nyújtani, amely a fejlesztési folyamat jobb átláthatóságát, 'mi-lenne-ha' analízisét (what-if-analysis) és a fejlesztési prioritásokból szükségszerűen származó változásokhoz gyors adaptációt képes elősegíteni.*

A tézisek további fontos jellemzőkkel bírnak: a bemutatott optimalizációs modellek általánosabb tervezési és ütemezési probléma osztályokat reprezentálnak. Hiszen a bemutatott modellek csak a problémák struktúráját definiálják, az ütemezendő elemek (jelen esetben funkciók és feladatok) más kontextusban lehetnek más elemek is (például csomagok, CPU feladatok). Továbbá, a bemutatott probléma megfogalmazások tovább finomíthatók megszorítások relaxációjával (pl. temporális megszorítások enyhítésével), illetve megszorítások erősítésével (pl. erőforrás intenzitás megszorítások bevezetésével) annak érdekében, hogy más hasonló tervezési és ütemezési problémákra megoldást szolgáltassunk.

Három tézist különböztetek meg a kutatási célokra vonatkozóan 1.2. A *Tézis 1* és a *Tézis 2* rendre az agilis kibocsátás tervezésre és iteráció tervezésre vonatkozó fogalmakat, modelleket és algoritmusokat mutatja be. A *Tézis 3* az elosztott környezetekre vonatkozó kiterjesztését tartalmazza az elosztott agilis kibocsátás tervezésnek. Valamennyi bemutatott tézis az IAPA építőköveit alkotják. A 1 ábra a téziseimet az agilis fejlesztési cikluson belül foglalja össze, hogy így egy vizuális áttekintést adjon a kutatásaimra vonatkozóan: a *Tézis 1* az agilis kibocsátás tervezésre, a *Tézis 2* az agilis iteráció tervezésre, végül a *Tézis 3* az elosztott agilis környezetek funkció disztribúciójára vonatkozik. Az ábra az előző vezérléstechnikai analógiát alkalmazva a különböző tervezési szintek bemeneteire és kimeneteire mutat rá.

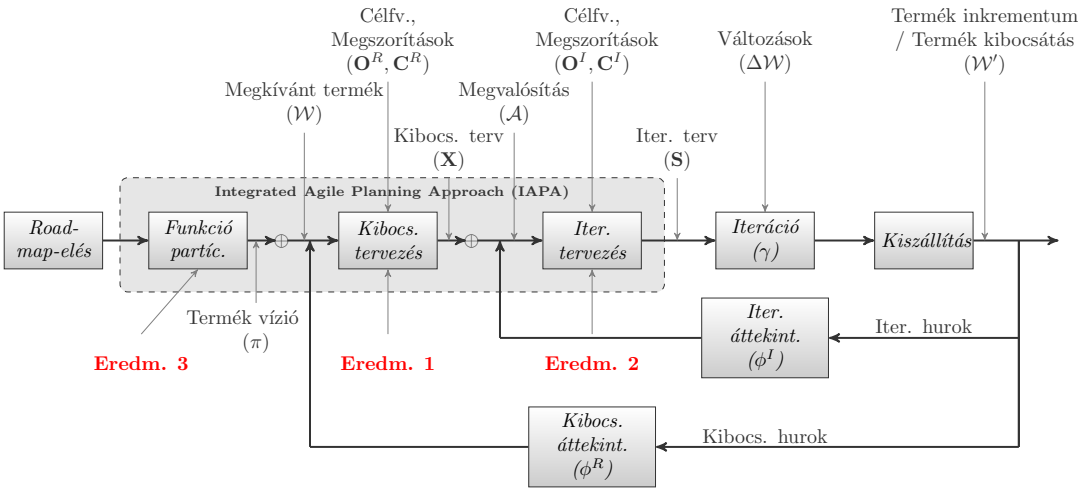

FIGURE 1: Agilis tervezési ciklusok.

2.1 Tézis 1

A következőkben az agilis kibocsátás tervezéssel kapcsolatos fogalmaimat, modelljeimet, és algoritmusaimat mutatom be. Ez a kidolgozott megközelítés egyenletesebben és jobban kitöltött iterációkat (az iterációk nagyobb, mint 80%-a optimális) eredményez, továbbá megakadályozza az erőforrások túlterhelését a manuális megközelítésekkel szemben. (Ez a tézis a disszertáció 4. fejezetében található meg.)

Először a tézisemet foglalom össze, majd a szimuláció és post mortem analízis alapú validációkat mutatom be, végül néhány alkalmazással kapcsolatos gyakorlati megjegyzés található.

C1: Az agilis kibocsátás tervezéshez egy új módszert (nevezetesen Agile Release Scheduling (ARS)) dolgoztam ki, amely a kibocsátásokkal kapcsolatos döntéstámogatás hatékonyságát növeli. [C6, E13, B2, G18, I26]

C1.1 *Megalkottam az agilis kibocsátás ütemezésnek egy konzisztens, ontológia-alapú információs modelljét (nevezetesen Agile Release Scheduling Model (ARSM)) ahhoz, hogy az agilis kibocsátás ütemezés tervezési terének komponenseit specifikáljam. Ez a modell magában foglalja a főbb fogalmakat és a közöttük lévő kapcsolatokat, továbbá agilis kibocsátás tervezést támogató eszközök koncepcionális modelljeként is alkalmazható.* [C6]

C1.2 *Az agilis kibocsátás ütemezést egy optimalizációs problémaként (nevezetesen Agile Release Scheduling Problem (ARSP)) fogalmaztam meg annak érdekében, hogy a csapat erőforrás kapacitását és az iterációk számának minimalizálását, mint célfüggvényt figyelembe vegye a funkció-allokáció megvalósításához. A kidolgozott megoldásom fontos újdonsága a matematikai precízitású probléma megfogalmazás. A probléma megfogalmazást egy kiterjesztett BINÁRIS TÖBBSZÖRÖS PAKOLÁSI optimalizációs modellel oldottam meg, amely a kibocsátások széles körét támogatni tudja: 1) dátum- és szkóp-vezérelt ütemezéssel, 2) funkciók közötti dependenciákkal, 3) csapat kapacitással, 4) funkció prioritással, 5) lépcsőzetes átadással és átadott funkciók megrendelői szempontú maximalizálásával.* [C6, E13, B2]

C1.3 *Megalkottam egy branch-and-bound többszörös pakolási algoritmust, amely képes a globális optimum megtalálására (idő komplexitása $\mathcal{O}(o * n^2)$, ahol n és o a funkciók számossága és az iterációk számossága)* [C6, E13].

2.1.1 Agilis kibocsátás tervezésének matematikailag precíz megfogalmazása

A kibocsátás tervezés célja ($\mathfrak{S}_{AR}$), hogy egy *megvalósítható* nagy szemcsézettségű fejlesztési tervet határozzon meg, amely megadja, hogy mely funkciók ($\mathcal{W}$) mely iterációkban ($\mathcal{I}$) legyenek átadva ($\mathbf{X}$ – funkció-iteráció összerendelés). A kibocsátás tervezés *optimalizált* verzióját a potenciálisan megvalósítható alternatívák közüli szélsőértékű terv kiválasztásával lehet származtatni. Ezt a feladatot egy PAKOLÁSI problémaként (KNAPSACK problem – KP) reprezentálhatjuk, amelyben a leszállítandó elemek egy halmaza (funkciók) az üzleti értékükkel (üzleti prioritás) és méretükkel (szükséges ráfordítás) jellemezettek. Továbbá az a cél, hogy egy vagy több diszjunkt iterációt kiválasztva, az iterációkba kerülő elemek méreteinek összege nem haladja meg az iterációk méretkorlátját és az egyes iterációkba kerülő elemek üzleti értéke maximalizált legyen [51].

Megalapozott döntéstámogatás nélkül a fejlesztés koordinátornak a következő tipikus megszorításokat (pl. $\mathbf{C}^R$) *informálisan* (implicit és intuitív módon) szükséges kezelnie: 1) *iterációk* ($\mathcal{I}$ – amikhez a funkciókat hozzá kell rendelni), 2) *dependenciák* ($\mathcal{D}$ – temporális megszorítások a funkciók között), 3) *erőforrás kapacitások* ($\mathcal{R}$ – fejlesztők, e – csapat hatékonysági faktor, $\mathbf{c}$ – kibocsátás alatti erőforrás igény), 4) *prioritások* ($\mathbf{p}$ – minden egyes funkció leszállításának fontossága), 5) *ráfordítás* ($\mathbf{w}$ – minden egyes funkció kifejlesztéséhez szükséges ráfordítás), 6) *kibocsátás hossza* (l^R), és 7) *iteráció hossza* (l^I_k).

Ennek következtében, a tervek optimalitása (értsd maximális üzleti érték) nagymértékben múlik a fejlesztés koordinátor 'megérzésein' – mindamellett a projekt tervek optimalitása kulcsfontosságú mind technikai, mind gazdasági szempontból is. Formálisabban fogalmazva azt mondhatjuk, hogy az agilis kibocsátás tervezés tervezési tere a következő faktorokból áll: $(\mathcal{W}, \mathcal{I}, \mathcal{D}, \mathcal{R}, e, \mathbf{c}, \mathbf{p}, \mathbf{w}, \mathbf{l^R}, \mathbf{l^I_k}, \mathbf{X})$ – röviden $(\mathcal{W}, \mathbf{C}^R)$ –, és célfüggvénye, hogy a következő optimális leképezést megtalálja:

$$\mathfrak{S}_{AR} : (\mathcal{W}, \mathbf{C}^R) \rightarrow \mathbf{X} \tag{1}$$

ahol a fenti leképezés a maximális érték leszállítását adja meg.

2.1.2 A Tézis 1 gyakorlati validációja

A tézisem validálása érdekében először hét, a valós életből származó reprezentatív adathalmaz post mortem analízisét végeztem el, amely adatok egy szoftverfejlesztő cégtől származtak [52]. Minden agilis

fejlesztési folyamat megvalósítása különböző, mégis a vizsgált szoftverfejesztő cégnél történt implementáció tipikusnak tekinthető a szervezet méretére (6 fő), az alkalmazott agilis módszerre (Scrum-szerű fejlesztési folyamat) és technikákra (XP fejlesztési praktikák) való tekintettel. Ennél a szervezetnél a kibocsátás ütemezési folyamat tipikus agilis tervezési lépésekből áll. A vizsgálat célja az volt, hogy a két módszer kimenetét megvizsgáljuk úgy, hogy a manuálissal megegyező bemeneti változókat használjunk az algoritmikus módszernél is.

A fentieken kívül további 120 hipotetikus adathalmazt generáltam, amelyeken szimulációval végeztem el a validálást az ütemezési probléma paramétereinek változtatásával: erőforrás kapacitások ($\mathcal{R}$), iteráció kapacitás (**c**), kibocsátás hossz (l^R), iteráció hossz (l^I_k) és dependenciák ($\mathcal{D}$)). Ez utóbbival a megoldás performanciáját és az általa szolgáltatott eredmények minőségét is vizsgálhattuk, valamint a különböző agilis módszerek alkalmazásából eredő ingadozásokat is kiszűrhetjük.

Post mortem analízis és számítási tapasztalatok

A következő kutatási kérdéseket fogalmaztam meg a vizsgálat során:

- **C1Q1**: *Milyenek a munkaerő követelmények az idő függvényében?* (értsd a $\mathcal{R}$ dinamikája);
- **C1Q2**: *Mennyi iterációra van szükség az egyes kibocsátásokhoz?*; (értsd $o \triangleq l^R/l^I$) és
- **C1Q3**: *Kontingencia puffereknek miként történik az allokációja?* (értsd $BpR \triangleq \Delta\mathcal{R} = \mathcal{R}_{planned} - \mathcal{R}_{actual}$)

A historikus esetben az összes iterációnak 17%-a több, mint 20%-kal, 59%-a kevesebb, mint 20%-kal, de egyenlő vagy több, mint 10%-kal és 17%-a kisebb, mint 10%-kal volt alul- illetve túlterhelve. Az összes iterációnak csak a 7%-a volt optimális. Ezzel szemben az optimalizált esetben az összes iterációnak 0%-a több, mint 20%-kal, 4%-a kevesebb, mint 20%-kal, de egyenlő vagy több, mint 10%-kal és 12%-a kisebb, mint 10%-kal volt alul- illetve túlterhelve, azonban az összes iterációnak a 84%-a optimális volt. Ebből következően az optimalizált esetben: a munkaerő követelmények (v.ö. **C1Q1**) egyenletesebben és jobban kitöltött iterációkat eredményeztek. Ez azt jelenti, hogy az algoritmus törekszik az erőforrások túlterhelésének megakadályozására, amelyek gyakran kiszállítási kockázatot hordoz magában. Az algoritmus továbbá arra is törekszik, hogy az erőforrások alulterhelését elkerülje, amely gazdaságtalan iteráció végrehajtást eredményez.

A kibocsátásonkénti iteráció-számot tekintve (vö. **C1Q2**) az optimalizált eset kisebb eltérést mutatott: néhány szituációban (a 7 esetből 3-ban) az algoritmusnak további iterációt kellett hozzáadnia a kibocsátáshoz, annak érdekében, hogy elkerülje az erőforrás túlterhelést. Ezért a valós életbeli helyzetben az algoritmust iteratív módon érdemes használni az ütemezési paraméterek változtatásával (pl. a l^I vagy a $\mathcal{R}$), hogy gazdaságilag még jobb ütemezést szolgáltassanak.

Végül, ha a kibocsátások pufferét tekintjük (vö. **C1Q3**), amelyet kontingenciák kialakítására használják a gyakorlatban, szintén jelentős eltérés figyelhető meg. Az optimalizált eset az algoritmus puffer allokációs tulajdonságára mutat rá: a temporális bufferek (vö. BpR) a kibocsátás végére tevődnek át az optimalizációs kritérium miatt (annyi elemet tegyünk egy iterációba, amennyit csak lehet). Ez a karakterisztika azt jelzi, hogy a kontingencia a kibocsátás végére kerül úgy, hogy az algoritmus a dependenciákat is figyelembe veszi. Ez a kibocsátás csúszásának csökkentése miatt ajánlott [53].

A mesterségesen generált 120 esetre vonatkozóan a kibocsátás tervezés minőségét az elméleti maximális erőforrás kihasználástól (értsd $\mathbf{c}$) való távolsággal (Δ) definiáltuk. Ez a távolság csak az iterációk alulterhelését jellemzi, mivel a túlterhelést a $\mathbf{c}$ megszorítás miatt kerüli el a kibocsátás ütemezési modell. A szimuláció eredményeképpen statisztikailag azt állíthatjuk, hogy a dependenciák száma ($\mathcal{D}$) negatívan korrelál az erőforrás kihasználással, ezért a kibocsátási terv minőségével is. Másrészről mind a funkciók száma, mind az iteráció kapacitása ($\mathbf{c}$) pozitívan korrelál a terv minőségének a mértékével. Azonban, minden szimulációs esetben az optimális pakolások az esetek 95%-ban teljesültek. Valamennyi paraméterezett szimulációt a historikus esettel összevetve – annak ellenére, hogy a szimulációs probléma sokkal komplexebb, mint a historikus – az algoritmikus megközelítés a manuális megközelítést kibocsátási terv minőségben jelentősen meghaladta. A mesterségesen előállított reprezentatív adatokon történt szimuláció megerősítette a post mortem analízis eseményeit.

A valós életből származó reprezentatív adatok, valamint a nagyszámú generált reprezentatív adathalmazokon (120) mutatott eredmények is egyöntetűen azt mutatják, hogy a megközelítésem az agilis kibocsátás tervezés terén a döntéstámogatásban jelentős gyakorlati értékkel bír.

2.1.3 Implementáció a tapasztalati validációhoz

A tapasztalati validációhoz egyrészt kifejlesztettem egy MS SharePoint alapú eszközt – nevezetesen SERPA™ – a kibocsátási tervezés fogalmi modellje alapján (vö. **C1.1**). Ezt az eszközt a korábban említett cégnél [52] jelenleg is használják, amely az eszköz hasznosságát támasztja alá. Továbbá a kibocsátás tervezési algoritmust Matlab-ban [54] implementáltam (vö. **C1.2-3**). Az algoritmus a PROPAS™ Matlab toolbox-om részeként implementáltam `mksched` algoritmus néven.

2.2 Tézis 2

A következőkben az agilis iteráció tervezéssel kapcsolatos fogalmaimat, modelljeimet, és algoritmusaimat mutatom be. Ez a megközelítés jelentősen növeli az erőforrások terhelés elosztását ($\approx 4-5\times$), jelentősen gyorsítja az iteráció ütemezésének előállítását ($> 50\%$), növeli a projekt végrehajtási hatékonyságát ($> 10\%$) és növeli a projekttervezés hatékonyságát ($> 50\%$) a manuális megközelítéssel összehasonlítva. (Ez a tézis a disszertáció 5. fejezetében található meg.)

Először a tézisemet foglalom össze, majd egy esettanulmányt és egy szimuláció alapú validációt mutatok be. Végül néhány alkalmazással kapcsolatos gyakorlati megjegyzés teszek.

C2: Az agilis iteráció tervezéshez egy új módszert (nevezetesen Agile Iteration Scheduling (AIS)) alkottam meg, amely az iterációkkal kapcsolatos döntéstámogatás hatékonyságát növeli. [C5, I26, D7, E9, E14, E11, E8, I25, G18, G16]

C2.1 *Megalkottam az agilis iteráció ütemezésnek egy konzisztens, ontológia-alapú információs modelljét (nevezetesen Agile Iteration Scheduling Model (AISM)) ahhoz, hogy az agilis iteráció ütemezés tervezési terének komponenseit specifikáljam. Ez a modell magában foglalja a főbb fogalmakat és a közöttük lévő kapcsolatokat, továbbá az agilis iteráció tervezést támogató eszközök koncepcionális modelljeként is alkalmazható.* [C6, C5]

C2.2 *Az agilis iteráció ütemezést optimalizációs problémaként (nevezetesen Agile Iteration*

Scheduling Problem (AISP)) fogalmaztam meg azért, hogy a funkciók implementálásának sorrendezése megvalósítható legyen. A megoldásom fontos újdonsága a matematikai precízitású probléma megfogalmazás. A probléma megfogalmazást egy ERŐFORRÁS MEGSZORÍTÁSOS PROJEKT ÜTEMEZÉSI (RESOURCE-CONSTRAINED PROJECT SCHEDULING problem – RCPSP) optimalizációs modellel oldottam meg, amely figyelembe veszi a temporális megszorításokat és a végrehajtási idő minimalizációs ütemezési célfüggvényt. A probléma megfogalmazása az RCPSP egy kiterjesztése, amely az iteráció tervezések széles körét képes támogatni: 1) előzetes erőforrás hozzárendeléssel (értsd bizonyos feladatokat az erőforrásokhoz rendelhessünk az ütemezés előtt) és az 2) iteráció időtartam timebox-os vezérléssel. [C5]

C2.3 *Megalkottam egy heurisztikus ütemezési algoritmust az RSPSP alapú problémára. Az algoritmus mohó stratégiát követ, ezért a globális optimumot nem biztosítja, a gyakorlati alkalmazások szempontjából (idő komplexitása $\mathcal{O}(n+m)$) mégis megfelelő megoldást ad.* [C5]

C2.4 *Megalkottam egy informális modellt, amely UML diagramokat tervezési célfüggvénnyel és megszorításokkal képes kiegészíteni. A kiterjesztés a standard UML Profile kiterjesztési mechanizmus felhasználásával készült el [55]. Megalkottam egy matematikai formulát ahhoz, hogy az adott elem kifejlesztéséhez szükséges erőforrásokat meghatározhassuk. Ehhez az elterjedt Use Case alapú Use-case Point (UCPM) becslési módszert alkalmaztam [56]. Információ fúzióval és a megalkotott formulával fél-automatikus modell-vezérelt tervezést lehetett megvalósítani a korábbi eredmények alkalmazásával (**C2.2-3**). Továbbá megalkottam egy gráf transzformációt, amely egy ütemezésből egy AoN (Activity-on-Node) gráf transzformációt tesz lehetővé az ütemezés vizualizációja érdekében. Így lehetővé válik elterjedt projekttervezési eszközökbe való importálás megvalósítása is [57].* [E9, E14, E11, E8]

C2.5 *Megvalósítottam az UML-alapú specifikációk egy általános üzleti célokkal való kiterjesztését is, hogy a Goal-oriented Requirements Language (GRL) nyelv segítségével, egy közös, konzisztens tárat szolgáltasson a menedzsment és a fejlesztés részére. Ez az információ fúzió könnyen lehetővé teszi a rendszerkövetelmények és az üzleti szükségletek egymáshoz való igazítását. Ezen kívül lehetővé teszi az ütemezés automatikus generálását is a korábbi eredmények alkalmazásával (**C2.2-4**).* [E9, E14, E11, E8]

2.2.1 Agilis iteráció tervezés matematikailag precíz megfogalmazása

Az iteráció tervezés ($\mathfrak{S}_{AI}$) célja az, hogy egy *megvalósítható*, finom szemcsézettségű fejlesztési tervet határozzon meg, amely ütemezi az iterációba ($\mathbf{S}$) választott funkciók implementációját ($\mathbf{S}$ – feladatfejlesztő hozzárendelés ($\mathcal{A}$)) [38]. Az iteráció tervezés *optimalizált* verzióját a potenciálisan megvalósítható alternatívák közül a szélsőértékű terv kiválasztásával lehet származtatni. Ezt a feladatot egy ERŐFORRÁS-MEGSZORÍTÁSOS PROJEKT ÜTEMEZÉSI problémaként (RESOURCE-CONSTRAINED PROJECT SCHEDULING problem – RCPSP) reprezentálhatjuk, amelyben az erőforrás allokáció során az aktivitások (fejlesztési feladatok) végrehajtásához időintervallumokat rendelünk úgy, hogy figyelembe vesszük mind a temporális, mind az erőforrás megszorításokat (erőforrások elérhetősége) és a minimális projektvégrehajtási időt, mint célfüggvényt [58].

Az iteráció tervezés szokásosan egy összetett feladat, hiszen a következő tipikus megszorításokat ($\mathbf{C}^I$) kell figyelembe venni: 1) *precedenciák* ($\mathbf{P}$ – temporális precedenciák a megvalósítási feladatok között), 2) *erőforrás terhelések kiegyensúlyozása* ($\mathcal{R}$ – erőforrás túlterhelések elkerülése), 3) *ráfordítás* ($\mathbf{w}$ – minden egyes feladat implementálásának ráfordítása), 4) *előre hozzárendelés* ($\mathbf{a}$ – a megfelelő fejlesztő kijelölése bizonyos feladatokhoz) és 5) *iteráció hossza* (l^I – iteráció végének a határideje a lépcsőzetes átadás miatt).

A tradícionális megközelítésekben, az ütemezést gyakran projektütemező szoftver segítségével állítják elő (pl. [57]), amelyek segítenek a megszorítások és célfüggvények kezelésében. Mivel ezek a tradícionális megközelítések manuális munkát igényelnek és relatíve hosszú ideig tartanak (néhány óra), ezért ezek túl nehézkesek az IDP-hez (különösen az agilis környezetekben), emiatt gyakran mellőzik őket. A megfelelő döntéstámogatás nélkül a terveket *informálisan* (implict és intuitív módon) állítják elő, a diszkrepanciák a csapat megbeszélések alatt oldódnak fel [36, 38]. Az informális megközelítések kis projektekben jól működnek, azonban ahogy a méret és a komplexitás nő, az ütemezés nagyon összetett feladattá válik, amely eszköztámogatást igényel [59, 60]. Formálisabban fogalmazva, azt mondhatjuk, hogy az agilis iteráció tervezési tere a következő faktorokból áll: $(\mathcal{A}, \mathbf{P}, \mathcal{R}, \mathbf{w}, \mathbf{a}, \mathbf{l^I}, \mathbf{S})$ – röviden $(\mathcal{A}, \mathbf{C}^I, \mathbf{S})$ –, és célfüggvénye, hogy a következő optimális leképezést megtalálja:

$$\mathfrak{S}_{AI} : (\mathcal{A}, \mathbf{C}^I) \rightarrow \mathbf{S} \tag{2}$$

ahol a fenti leképezés a minimális végrehajtási időt adja meg.

2.2.2 A Tézis 2 gyakorlati validációja

A Tézis 2 **C2.1**, **C2.2** és **C2.3** részeinek validálásához elsőként egy post mortem analízist hajtottam végre négy valós életbeli reprezentatív adathalmazon, amelyek egy szoftverfejlesztő cégtől származtak [52]. Továbbá egy longitudinális vizsgálatot (egy esettanulmányt) végeztem, ahhoz, hogy a **C2.4** és **C2.5** részeket egy valós életből származó szoftver projekttervezési helyzetben értékeljem. A pilot projekt egy valós alkalmazás teljes modulja volt a korábban említett cégnél [52]. Minden agilis fejlesztési folyamat megvalósítása különböző, mégis a vizsgált szoftverfejlesztő cégnél történt implementáció tipikusnak tekinthető a szervezet méretére (6 fő), az alkalmazott agilis módszerre (Scrum-szerű fejlesztési folyamat) és technikákra (XP fejlesztési praktikák) való tekintettel. Ennél a szervezetnél az iteráció ütemezési folyamat tipikus agilis tervezési lépésekből áll (ld. 2.2.1 fejezetet). A vizsgálat célja az volt, hogy a két módszer kimenetét megvizsgáljuk úgy, hogy a manuálissal megegyező bemeneti változókat használjunk az algoritmikus módszernél is.

További 480 különböző mesterségesen megalkotott reprezentatív adathalmazon végeztem szimulációs vizsgálatot – az iterációs probléma különböző parmétereinek a változtatásával (ld. 2.2.1 fejezetet: technikai feladatok fejlesztési erőforrás igénye (w), erőforrás kapacitások ($\mathcal{R}$), iteráció kapacitása ($\mathbf{c}$), iteráció hossza (l^I_k) és precedenciák ($\mathbf{P}$)) – annak érdekében hogy betekintést nyerjünk a módszer performanciájával és az az előállított terv minőségével kapcsolatban a különböző agilis fejlesztési szituációk statisztikai ingadozásainak kiszűrésével.

A C2.1, C2.2 és C2.3 post mortem analízise és számítási tapasztalatai

A következő kutatási kérdések merültek fel a vizsgálat során:

Az optimalizáció alapú iteráció ütemezés hogyan értékelhető az informálissal szemben az alábbiak alapján

- **C2Q1**: *az erőforrások időbeli terhelése* (értsd dinamizmus $\mathcal{R}$);

- **C2Q2**: *a tervek minősége* (értsd l^I-t nem haladhatja meg és P-t automatikusan kezeljük) és
- **C2Q3**: *a tervek megvalósíthatósága* (értsd a $\mathcal{R}$ terhelése)

A szimuláció megmutatta, hogy a módszer könnyen megbirkózik a precedencia megszorításokkal, jelentősen növeli az erőforrások terhelésének elosztását (cca. $4-5\times$), magasabb minőségű és alacsonyabb kockázattal megvalósítható ütemtervet generál és a fejlesztő csapatok döntéseit jobban támogatja (vö. **C2Q1**).

Az eredményeket a variáció koefficienseként (a szórás normalizált mértéke) kifejezve, az optimalizáció alapú ütemezéssel $\approx$ 5-ször jobb erőforrás kiegyenlítést lehet elérni, mint az intuitív módszerrel (vö. **C2Q1**). Az algoritmikus módszer az összetett döntési szituációkat is könnyedén kezeli – mivel a feladatok közötti precedenciákat figyelembe veszi és az erőforrás túlterhelést is elkerüli – a historikus esettel szemben, ahogy ezeket a napi megbeszélések során kezelik. Ezek következtében az algoritmikus módszer ezen két képessége biztosítja a magasabb minőséget és alacsonyabb kockázattal megvalósítható tervet a historikus esettel szemben (vö. **C2Q2-3**). A mesterségesen előállított reprezentatív adatokon történt szimuláció megerősítette a post mortem analízis eseményeit.

A valós életből származó reprezentatív adatok, valamint a nagyszámú generált reprezentatív adathalmazokon (480 különböző eset) való futtatások eredményei is egyöntetűen azt mutatják, hogy a megközelítésem az elosztott agilis kibocsátás tervezés terén a döntéstámogatásban jelentős gyakorlati értékkel bír.

A C2.4 és a C2.5 esettanulmány alapú analízise

Egy valós életbeli szoftverfejlesztési szituációban egy esettanulmányt végeztem a módszerem (ld. **C2.4**, **C2.5**) értékeléséhez [61]. A pilot projekt egy ipari alkalmazás teljes modulja volt [52].

A módszerem validálásához a következő hipotéziseket fogalmaztam meg:

- **C2H1**: *A módszer (és eszköz) a szoftverkövetelmény metrikákkal, tervezési megszorításokkal és célfüggvényekkel való RCPSP-ként való megfogalmazása nagyobb produktivitást (értsd kevesebb manuális munkát) tesz lehetővé a projekt kibocsátás tervezésben az intuitív módszerekhez képest.*;
- **C2H2**: *A módszer az ütemezési algoritmus különböző paraméterezésével 'mi-lenne-ha' analízist tesz lehetővé, amely hatásosabb döntéstámogatást tesz lehetővé (értsd számos alternatívából való választással) az iteráció tervezési döntésekben.*;
- **C2H3**: *A módszer a követelményspecifikációból való iteráció terv származtatásával precízebb követelményszintű nyomon követést tesz lehetővé.*;
- **C2H4**: *Követelmény metrikák, projekttervezési megszorítások és célfüggvények integrálása hatékonyabb projekt végrehajtást eredményeznek kevesebb szinkronizációs és dokumentációs overhead mellett.*; és
- **C2H5**: *Az AISP-nek a szoftver követelmény specifikációból való származtatása nagyobb produktivitást enged meg a projekttervezésben a hagyományos, főként manuális, projekttervezési módszerrel szemben.*

Az esettanulmány rámutatott, hogy a módszer 1) jelentősen képes gyorsítani az ütemezés előállítását ($> 50\%$), 2) jobb döntéstámogatást biztosít, 3) precízebb Use case szintű követelmény nyomonkövetést tesz lehetővé, továbbá 4) a módszer lehetővé teszi a több, mint 10%-os hatékonysági növekedést a projekt végrehajtásban, valamint 5) a több, mint 50%-os tervezési hatékonyság növekedést mutat kevesebb szinkronizációs és dokumentációs overhead mellett.

2.2.3 Implementáció a tapasztalati validációhoz

A tapasztalati validációhoz a SERPA™ web alkalmazás koncepcionális modelljét kiegészítettem az iterációs tervezéssel (vö. **C2.1**). Továbbá a kibocsátás tervezési algoritmust Matlab-ban [54] implementáltam (vö. **C2.2-3**). Az algoritmus a PROPAS™ Matlab toolbox-om részeként implementáltam `lscap` algoritmus néven, amelyet a szimuációk során is használtam. Ezeken kívül egy UML alapú integrált ütemező alkalmazást is implementáltam. A PYTHIA PROJECT PLANNER™ prototípus az Eclipse platformon UML2 technológiára épülve [62, 63] készült el.

2.3 Tézis 3

A következőkben az elosztott agilis kibocsátás tervezéssel kapcsolatos fogalmaimat, modelljeimet, és algoritmusaimat mutatom be. Ez a módszer $\approx 14-18$-szer kevésbé intenzív kommunikációs és koordinációs igényt tesz szükségessé, valamint 40%-szor jobb erőforrás kihasználást biztosít a tradicionális elosztott agilis kibocsátás tervezéshez képest. (Ez a tézis a disszertáció 6. fejezetében található meg.)

Először a tézisemet foglalom össze, majd a validációjukat mutatom be, végül néhány alkalmazással kapcsolatos gyakorlati megjegyzés található.

C3: Az elosztott agilis kibocsátás tervezéshez egy új módszert (nevezetesen Feature Partitioning Method) alkottam meg, amely elosztott agilis környezetekben a kibocsátásokkal kapcsolatos döntésetámogatás hatékonyságát növeli. [C4, A1, E13, C3]

C3.1 *Definiáltam egy Feature Architectural Similarity Analysis (FASA) elnevezésű analitikus lépést, hogy a funkciók (leszállítandó funkcionális és nem-funkcionális követelmények) között architekturális hasonlóságot lehessen kifejezni. Ezt a hasonlóságot arra lehet használni, hogy az azonos szoftver modul halmazokban implementálandó funkciókat azonosítsuk.* [C4, A1, C3]

C3.2 *Definiáltam egy Feature Chunk Construction lépést – Feature Chunk Construction optimalizációs probléma megfogalmazásával (FCCP) – amely a fejlesztési feladatok elosztását szabályozza a különböző helyeken annak érdekében, hogy a kommunikációs és koordinációs szükségleteket minimalizálja az elosztott csapatok között. A megoldásom fontos újdonsága a matematikai precízitású probléma megfogalmazás. A probléma formulációt egy minimum k-vágás (MINIMUM K-CUT) gráf partícionálási optimalizációs modellel oldottam meg. Ezzel a megközelítéssel, az elosztott csapatok közötti kommunikációs és a koordinációs komplexitás jelentősen csökkenthető a fejlesztési feladatok meghatározott partíciók szerinti elrendezésével.* [C4]

2.3.1 Agilis kibocsátás tervezés elosztott kiterjesztésének matematikailag precíz megfogalmazása

Az agilis kibocsátás tervezés elosztott kiterjesztése ($\mathfrak{A}_F$) döntéshozási folyamatként definiálható, ahol a cél az, hogy meghatározzuk azon *megvalósítható* funkció-csoportokat, amelyeket az egyes elosztott csapathoz rendelünk. A helyi agilis kibocsátás tervezés ezen funkciócsoportokon végezhető el. Az agilis kibocsátás tervezés elosztott kiterjesztésének *optimalizált* verzióját a potenciálisan megvalósítható alternatívák közüli szélsőértékű terv kiválasztásával lehet származtatni. Ezt a feladatot egy MINIMUM

VÁGÁS gráf particionálási problémaként (MINIMUM CUT) reprezentálhatjuk, amelyben a cél a csomópontok (fejlesztő csapatok) közötti olyan élek egy halmazának (kommunikációs utak) megkeresése, amelyeknek az eltávolításával a gráf összefüggő komponensekre (funkció partíciókra) particionálható [64]. A minimalizációs célfüggvény a csapatok közötti szinkronizációs és kommunikációs igény intenzitásának minimalizálására törekszik. A célfüggvénnyel az elosztottságból eredő negatív hatásokat minimalizálhatjuk – úgy, mint a csapat produktivitásának csökkenése, nagyobb produkciós intervallumok, megnövekvő kommunikációs költségek és az elosztott csapatok között nehezebb folyamatkoordináció [43, 41, 65, 66].

A funkció csoportokat ($\mathbf{W}^*$) a közöttük lévő (*architekturális hatás* nézőpontból tekintett) *kohézió* alapján határozzuk meg. Minél magasabb a funkciók közötti kohézió, annál erősebb az igény arra, hogy ezek a funkciók egy csoportba kerüljenek. A funkciók közötti kohézió azonosítására egy bináris relációt ($\otimes$) vezettünk be a funkciók ($\mathbf{W}$) és a szoftver modulok ($\mathcal{M}$) között. Ez a reláció azt jelöli, hogy az adott funkció az adott szoftvermodulban lesz kifejlesztve. A célunk az, hogy egy csoportba kerüljenek azok a funkciók, amelyek hasonló modulhalmazban kerülnek kifejlesztésre, tehát ezek implementálása hasonló architekturális hatással rendelkezik. Ezzel a megközelítéssel jelentősen csökkenthető a kommunikációs igény és koordinációs komplexitás, ha a kifejlesztendő funkciókat az azonosított csoportok szerint osztjuk szét az elosztott csapatok ($\mathcal{T}$) között. Tradicionálisan a funkció csoportok manuális előállítása relatíve hosszú ideig tart (néhány óra), valamint az optimalizációs célfüggvény (a funkciók k kohézív funkciócsoportba való partícionálása) manuálisan nehezen kivitelezhető.

Formálisabban fogalmazva, azt mondhatjuk, hogy az agilis kibocsátás tervezés elosztott kiterjesztésének tervezési tere a következő faktorokból áll: $(\mathcal{W}, \mathcal{M}, \mathcal{T}, \otimes, \mathbf{W}^*)$, és célfüggvénye, hogy a következő optimális k-particiót megtalálja:

$$\mathfrak{A}_F : (\mathcal{W}, \mathcal{M}, \mathcal{T}, \otimes) \rightarrow \mathbf{W}^* \tag{3}$$

2.3.2 A Tézis 3 gyakorlati validációja

A tézisem validálása érdekében először hét, a valós életből származó reprezentatív adathalmaz post mortem analízisét végeztem el, amelyek egy szoftverfejlesztő cégtől származtak [52]. Minden agilis fejlesztési folyamat megvalósítása különböző, mégis a vizsgált szoftverfejesztő cégnél történt implementáció tipikusnak tekinthető a szervezet méretére (18 fő), az alkalmazott agilis módszerre (Scrum-szerű fejlesztési folyamat) és technikákra (XP fejlesztési praktikák) való tekintettel. Ennél a szervezetnél a kibocsátás ütemezési folyamat tipikus agilis tervezési lépésekből áll. A fix csapattagok különböző lokációkban dolgoztak, ezért gyakran nem látták egymást és nem tudtak egymással személyesen találkozni. A kommunikáció főként videokonferenciákon, telefonon és email-eken alapult; mivel az összes fejlesztő magyar volt, nem voltak nyelvi, kultúrális és időzóna eltérésből fakadó korlátok. A vizsgálat célja az volt, hogy a két módszer kimenetét megvizsgáljuk úgy, hogy a manuálissal megegyező bemeneti változókat használjunk az algoritmikus módszernél is.

További 360 különböző mesterségesen generált reprezentatív adathalmazon végeztem szimulációs vizsgálatot – az elosztott agilis kibocsátás tervezési probléma különböző parmétereinek a változtatásával (ld. 2.3.1 fejezetet: funkció számosság ($\mathbf{W}$), elosztott csapat jellemzők ($\mathcal{T}$), modul vonatkozás ($\otimes$) – annak érdekében hogy betekintést nyerjünk a módszer performanciájával és az az előállított terv minőségével kapcsolatban a különböző agilis fejlesztési szituációk statisztikai ingadozásainak a kiszűrésével.

Post mortem analizis és számítási tapasztalatok
A következő kutatási kérdéseket fogalmaztam meg a vizsgálat során:

A Feature Partitioning Method hogyan értékelhető az informálissal megoldással szemben

- **C3Q1**: *kommunikációs és koordinációs szükségletek alapján* (értsd $\nabla_{CCI} = f(\phi^I, \phi^R)$);
- **C3Q2**: *erőforrás terhelés szerint* (értsd $\nabla_{Res} = f(\mathcal{R})$); és
- **C3Q3**: *tervek megvalósíthatósága alapján* (értsd a $\mathcal{R}$ terhelése)

Az eredmények alapján az optimalizált eset 1) a fejlesztési munkát képes kohézív funkció csoportokra bontani olyan módon, hogy az elosztott csapatok nagyjából különböző modul halmazokon dolgozzanak. A szétbontás következtében a kommunikáció leginkább a közösen megvalósított funkcióknál fordul elő és 2) nagyjából $\nabla_{CCI} \approx 14 - 18$-szer kevésbé intenzív kommunikációs és koordinációs igényt támaszt, mint az intuitív megközelítés (vö. **C3Q1**), 3) nagyjából $\nabla_{Res} = 40\%$-kal jobb erőforrás kihasználást nyújt (amely alacsonyabb implementációs kockázatot és jobb erőforrás kihasználást jelent az erőforrások alul- és túlterhelésének elkerülésével értsd **C3Q2**), és 4) magasabb minőségű funkció elosztási tervet szolgáltat a funkciócsoportok fél-automatikus előállításával (amely lehetővé teszi a funkciók pillanatok alatti újrapartícionálását annak érdekében, hogy 'mi-lenne-ha' analízist végezhessünk az agilis környezetek állandóan változó környezetéhez való adaptációhoz – vö. **C3Q3**). A mesterségesen előállított reprezentatív adatokon történt szimuláció megerősítette a post mortem analízis eseményeit.

A valós életből származó reprezentatív adatok, valamint a nagyszámú generált reprezentatív adathalmazokon (360 különböző eset) való futtatások eredményei is egyöntetűen azt mutatják, hogy a megközelítésem az elosztott agilis kibocsátás tervezés terén a döntéstámogatásban jelentős gyakorlati értékkel bír.

2.3.3 Implementáció a tapasztalati validációhoz

A tapasztalati validációhoz a Kernighan-Lin (KL) gráf partícionáló módszert [67, 68] Matlab-ban [54] implementáltam). Az algoritmust a PROPAS^{TM} Matlab toolbox-om részeként implementáltam, amelyet a szimulációk során is használtam.

3 Az eredmények gyakorlati alkalmazása

A kutatás által kitűzött célok egy K+F-es Pythia nevű projekt keretén belül megvalósultak meg[4] [73]. Az IAPA megközelítést egy nagy nemzetközi bank számos projektjében sikeresen alkalmazták.

A Pythia projektbe beletartozott az elméleti eredmények ipari körülmények közötti vizsgálata is, amelyhez három prototípust is készítettem:

- **SERPA^{TM} *prototípus***: Megvalósítottam egy MS SharePoint alapú web alkalmazást [74, 52]. A SharePoint egy böngésző alapú kollaborációs- és dokumentum menedzsment platform, amely lehetővé teszi különböző listák létrehozását. A korábban említett kibocsátás- és iteráció tervezési

[4] A fejlesztést részben egy GVOP-s pályázat támogatta (GVOP-3.3.3-05/1.-2005-05-0046/3.0). Az implementációban a Multilogic Kft. [52] és az OptXware Kft. [69] is rész vett. A tézisek esettanulmány alapú validációja a Multilogic Kft-nél [52] történtek, a szimulációhoz használt adathalmazok a Magyar Posta IT fejlesztési részlegétől [70], az APEH IT részlegétől [71], a Prolan Zrt-től [72] és a Multilogic Kft-től [52] származtak.

információs modellek elemei, mint SharePoint listák lettek implementálva. A portált így kollaborációs munkaterületként lehetett definiálni a fejlesztők és a menedzsment számára, hogy valamennyi tervezéshez szükséges információt össze lehessen gyűjteni. Ezzel a web alkalmazással a fejlesztők felbonthatják a funkciókat technikai feladatokká, megjelölhetik a precedenciákat az egyes funkciók között, erőforrás becsléseket végezhetnek, feladatok és hibajavítások státuszait állíthatják be, amely egyedileg bevitt információt a projekt minden tagjával meg lehet osztani a kommunikáció elősegítése érdekében. Ez a prototípus valójában egy kész szoftver. Ennek a szoftvernek a használata egy szoftverfejlesztő cégnél már évek óta a napi munka része, amely segít a projektek tervezésében és monitorozásában.[5].

- **PROPAS^{TM} *prototípus***: Megvalósítottam a korábban bemutatott kibocsátás- és iteráció tervezési és funkció partícionáló algoritmusokat Matlab-ban, hogy a Serpa-n gyűjtött adatokon alapulva tervezési döntéstámogatást adjon. A PROPAS^{TM} toolbox nemcsak az implementált tervezési és particionáló algoritmusokat tartalmazza, hanem számos gráfelméleti algoritmussal és vizualizációs képességel is bír[6].
- **PYTHIA PROJECT PLANNER^{TM} *prototípus***: Implementáltunk[7] egy speciális integrált ütemezési megoldást az Eclipse platformon az UML2 technológia segítségével [62, 63]. Az implementációhoz a népszerű UML2 Profile mechanizmusát használtuk fel annak érdekében, hogy a tervezési adatokat az UML modellbe injektáljuk, majd XML kimenetet állítottunk elő, amelyet egy népszerű ütemezési eszköz bemeneteként használtunk [57]. A szoftver specifikálása, megtervezése és részleges implementációja (nevezetesen az informális modell és a tervezési algoritmusok) kivitelezése volt a feladatom. Ez az eszköz nemcsak ütemezési képességel bír, hanem Bayes-háló (Bayes Beliefs Network – BBN) alapú következtetést is magában foglal a projekttervek bizonytalanságainak analíziséhez. Az eszköz lehetővé teszi a valószínűségi következtetéseket és a BBN alapú döntéstámogató rendszerek készítését[8].

A megközelítés hatékonyságát és hatásosságát tapasztalati bizonyítékok támasztják alá, amelyeket két ipari esettanulmány ([E9]) és három reprezentatív adathalmazon végzett post mortem szimuáció is bizonyít ([C5, E13, C6]). További vizsgálatok még szükségesek lehetnek az IAPA megközelítés teljeskörű értékeléséhez. Azonban a valós életből származó reprezentatív adahalmazokon bemutatott szimulációs eredmények, valamint az esettanumányok eredményei is azt mutatják, hogy a megközelítésem az agilis tervezés terén a döntéstámogatásban gyakorlati értékkel bír.

Végezetül, azt hiszem, hogy a bemutatott tézisek jól illeszkednek a SEMAT célkitűzéseibe azzal, hogy jól megalapozott elméleti és módszertani támogatást szolgáltatnak a szoftvermérnöki terület agilis kibocsátás- és iterációtervezés kérdéskörében.

[5] A szoftvert a Multilogic Kft használja.
[6] `https://bitbucket.org/aszoke/matlab` vagy `http://www.kese.hu/`
[7] Az implementációt a Multilogic Kft. és az OptXware Kft. együttesen valósította meg.
[8] A BBN alapú döntéstámogatási képesség implementálása a Multilogic és a Budapesti Műszaki Egyetem Méréstechnikai és Információsrendszerek Tanszék Intelligens Rendszerek Kutatócsoportjának együttműködésével valósult meg. [75].

4 Publikációk jegyzéke

Könyvfejezet

[A1] Akos Szoke. “Horizons in Computer Science Research”. In: ed. by Thomas S. Clary. Vol. 5. New York: Nova Science Publishers, 2011. Chap. A Workload-balancing Method for Agile Distributed Software Development Environments, pp. 161–193.

Cikk szerkesztett könyvben

[B2] Akos Szoke. “Bin-packing-based Planning of Agile Releases (OUTSTANDING RESEARCH PAPER)”. In: *ENASE 2008/2009 Revised Best Papers*. Ed. by Stefan Jablonski Leszek Maciaszek César González-Pérez. Vol. 64. Communications in Computer and Information Science (CCIS). May 9-10, 2009. Milano, Italy: Springer-Verlag, May 2010, pp. 133–146.

Külföldön megjelent idegen nyelvű folyóiratcikk

[C3] Akos Szoke. “Optimized Feature Distribution in Distributed Agile Environments”. In: *PROFES ’10: Proceedings of the 11th International Conference on Product Focused Software Process Improvement*. Ed. by Muhammad Ali Babar, Matias Vierimaa, and Markku Oivo. Vol. 6156. Lecture Notes in Computer Science. June 21-23, 2010. Limerick, Ireland: Springer-Verlag, June 2010, pp. 62–76. ISBN: 978-3-642-13791-4.

[C4] Akos Szoke. “A Feature Partitioning Method for Distributed Agile Release Planning (BEST RESEARCH PAPER)”. In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by Will Aalst et al. Vol. 77. Lecture Notes in Business Information Processing ISBN: 978-3-642-20677-1. May 10-14, 2011. Madrid, Spain: Springer-Verlag, May 2011, pp. 27–42.

[C5] Akos Szoke. “Decision Support for Iteration Scheduling in Agile Environments”. In: *PROFES ’09: Proceedings of the 10th International Conference on Product Focused Software Process Improvement*. Ed. by Frank Bomarius et al. Vol. 32. Lecture Notes in Business Information Processing. June 15-17, 2009. Oulu, Finnland: Springer-Verlag, June 2009, pp. 156–170. ISBN: 978-3-642-02151-0.

[C6] Akos Szoke. “Conceptual Scheduling Model and Optimized Release Scheduling for Agile Environments”. In: *Information and Software Technology* 53.ISSN: 0950-5849 (June 2011), pp. 574–591.

Magyarországon megjelent idegen nyelvű folyóiratcikk

[D7] Akos Szoke, Orsolya Doban, Andras Pataricza. “Quality-driven Optimized Resource Allocation”. In: *Periodica Politechnica Electrical Engineering* 54 (Feb. 2010), pp. 71–78. URL: http://www.pp.bme.hu/ee/2010_1/pdf/ee2010_1_07.pdf.

Nemzetközi konferencia-kiadványban megjelent idegen nyelvű előadás

[E8] Akos Szoke. “Use case-driven Project Planning in System Development Projects”. In: *51st EOQ Annual Congress*. European Organization for Quality Annual Congress. May 22-23, 2007. Czech Society for Quality. Prague, Czech Republic, May 2007.

[E9] Akos Szoke. “A Proposed Method for Release Planning from Use Case-based Requirements”. In: *Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications*. Euromicro SEAA. September 3-5, 2008. Parma, Italy: IEEE Computer Society, Sept. 2008, pp. 449–456.

[E10] Akos Szoke. “Quality-driven Software Development”. In: *The Fourth Conference of PhD Students in Computer Science*. Ed. by Tibor Csendes. CS2. Volume of extended abstracts July 1–4, 2004. Institute of Informatics of the University of Szeged. Szeged, Hungary, July 2004, p. 118.

[E11] Akos Szoke. “A Proposed Method for Project Planning from UML Models”. In: *Work in Progress Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications*. Ed. by Erwin Grosspietsch and Konrad Klöckner. Euromicro SEAA ISBN: 978-3-902457-20-3. September 3-5, 2008. Parma, Italy: SEA-Publications: SEA-SR-20, 2008.

[E12] Andras Pataricza, Orsolya Doban, Akos Szoke. “Costs/Benefits of Using Formal Methods”. In: *Proceedings of The International Conference on Dependable Systems and Networks*. DSN. Supplemental Volume of the 2004 International Conference on Dependable Systems and Networks June 28 - July 1, 2004 http://2004.dsn.org/sessions.html. Palazzo dei Congressi, Florence, Italy, June 2004, pp. 104–105.

[E13] Akos Szoke. “Agile Release Planning through Optimization”. In: *ENASE 2009 Fourth International Conference on Evaluation of Novel Approaches to Software Engineering Proceedings*. Ed. by Leszek Maciaszek Stefan Jablonski. May 9-10, 2009. INSTICC. Milan, Italy: INSTICC Press, May 2009, pp. 149–160.

[E14] Akos Szoke. “A Proposed Method for for Automated Project Scheduling using Goals and Scenarios”. In: *Proceedings of the 16th IEEE International Requirements Engineering Conference*. IEEE Requirements Engineering. September 8-12, 2008. IEEE Computer Society. Barceona, Spain: IEEE Computer Society, Sept. 2008, pp. 339–340.

Magyar nyelvű folyóiratcikk

[F15] Akos Szoke. “Szoftverprojektek Bayes-háló-alapú kockázatanalízise”. In: *Magyar Távközlés* 2.HU ISSN: 0856-9648 (June 2006), pp. 18–24.

Magyar nyelvű konferencia-előadás

[G16] Akos Szoke. “Quality Metrics and Models in Software Development Processes”. In: *Proceedings of the 11th PhD Mini-Symposium*. ISBN: 963 420 785 5. February 3-4, 2004. Budapest University of Technology and Economics. Budapest, Hungary, Feb. 2004, pp. 36–37.

[G17] Akos Szoke. “Project Risk Management Based on Bayes Belief Net Using EMF”. In: *Proceedings of the 13th PhD Mini-Symposium*. ISBN: 963 420 853 3. February 6–7, 2006. Budapest University of Technology and Economics. Budapest, Hungary, Feb. 2006, pp. 70–71.

[G18] Akos Szoke. “Improving Quality in Agile Software Development”. In: *Proceedings of the 12th PhD Mini-Symposium*. February 8-9, 2005. Budapest University of Technology and Economics. Budapest, Hungary, Feb. 2005, pp. 42–43.

Publikáció más területen

[H19] Akos Szoke, András Förhécz, and György Strausz. “Versioned linking of semantic enrichment to legal documents”. In: *Artificial Intelligence and Law Journal* Special 25th Anniversary Issue (4 2013), pp. 485–519.

[H20] Akos Szoke, András Förhécz, and György Strausz. “A Unified Change Management of Regulations and their Formal Representations based on the FRBR Framework and the Direct Method”. In: *Legal Knowledge and Information Systems - JURIX 2012*. Ed. by Burkhard Schäfer. Frontiers in Artificial Intelligence and Applications. IOS Press, 2012, pp. 147–156.

[H21] Akos Szoke et al. “Linking Semantic Enrichment to Legal Documents”. In: *Proceedings of the Workshop on Modelling Policy-making (JURIX-MPM 2011)*. Ed. by Adam Wyner and Neil Benn. 2011, pp. 11–15.

[H22] Gabor Korosi, Akos Szoke. “Elektronikus közbeszerzés II.” In: *Jegyző és Közigazgatás* 6.ISSN: 1589-3383 (Nov. 2003), p. 43.

[H23] Peter Halacsy, Gabor Kiss, Akos Szoke. “Az információ-visszakeresés modelljei”. In: *Magyar Távközlés* 3 (2003), pp. 30–37.

[H24] Akos Szoke, Krisztián Mácsár, and György Strausz. “A Text Analysis Framework for Automatic Semantic Knowledge Representation of Legal Documents”. In: *Workshop on Network Analysis in Law (In conjunction with ICAIL 2013: 14th International Conference on AI and Law)*. Ed. by Radboud Winkels. Rome, May 2013.

Egyéb nem publikáció értékű munka

[I25] Akos Szoke. *Quality Metrics and Models*. Tech. rep. Budapest University of Technology and Economics, Jan. 2004.

[I26] Akos Szoke, Orsolya Doban, Andras Pataricza. *Minőségvezérelt projektmenedzsment*. Magic days (presentation). Visegrad, Hungary. June 2005.

[I27] Akos Szoke. *IT projektmenedzsment kvantitatív alapú döntéstámogatása*. Metódus day (presentation). Budapest, Hungary. Feb. 2007.

[I28] Akos Szoke. *Esettanulmány: Szoftverfejlesztési folyamat nemzetközi banki környezetben*. Agile Alliance Hungary Meetup (presentation). Budapest, Hungary. Mar. 2011.

5 Hivatkozások

[29] Per Kroll and Philippe Kruchten. *The rational unified process made easy: a practitioner's guide to the RUP*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0-321-16609-4.

[30] Kent Beck and Cynthia Andres. *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional, Nov. 2004. ISBN: 0321278658.

[31] Steve R. Palmer and Mac Felsing. *A Practical Guide to Feature-Driven Development*. Pearson Education, 2001. ISBN: 0130676152.

[32] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN: 0130676349.

[34] Barry Boehm. "Get Ready for Agile Methods, with Care". In: *Computer* 35.1 (2002), pp. 64–69. ISSN: 0018-9162.

[35] Tore Dybå. "Improvisation in Small Software Organizations". In: *IEEE Softw.* 17.5 (2000), pp. 82–87. ISSN: 0740-7459.

[36] Craig Larman. *Agile and Iterative Development: A Manager's Guide*. Pearson Education, 2003. ISBN: 0131111558.

[37] Tsun Chow and Dac-Buu Cao. "A survey study of critical success factors in agile software projects". In: *Journal of System and Software* 81.6 (2008), pp. 961–971. ISSN: 0164-1212.

[38] Mike Cohn. *Agile Estimating and Planning*. NJ, USA: Prentice Hall PTR, 2005. ISBN: 0131479415.

[39] VersionOne. *7th Annual Survey: 2012, The State of Agile Development*. Full Data Report. 2013.

[40] B. Nuseibeh and S. Easterbrook. "Requirements engineering: a roadmap". In: *ICSE - Future of SE Track*. 2000, pp. 35–46.

[41] Tore Dybå and Torgeir Dingsøyr. "Empirical studies of agile software development: A systematic review". In: *Information and Software Technology* 50.9-10 (2008), pp. 833–859. ISSN: 0950-5849.

[42] Claes Wohlin et al. *Experimentation in Software Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers, 2000. ISBN: 0-7923-8682-5.

[43] P. Abrahamsson et al. *Agile software development methods - Review and analysis*. Tech. rep. 478. VTT PUBLICATIONS, 2002.

[44] Scott W. Ambler. "Survey Says: Agile Works in Practice". In: *Dr. Dobb's Journal* 31 (Mar. 2006), pp. 62–64.

[45] Sanjiv Augustine. *Managing Agile Projects*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005. ISBN: 0131240714.

[46] Andrew Begel and Nachiappan Nagappan. "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study". In: *ESEM '07: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 255–264. ISBN: 0-7695-2886-4.

[47] T. Chau, F. Maurer, and Grigori Melnik. "Knowledge Sharing: Agile Methods vs. Tayloristic Methods". In: (2003), pp. 302–307.

[48] VersionOne. *5rd Annual Survey: 2010, The State of Agile Development*. Full Data Report. June 2010.

[49] VersionOne. *4rd Annual Survey: 2009, The State of Agile Development*. Full Data Report. June 2009.

[50] VersionOne. *3rd Annual Survey: 2008, The State of Agile Development*. Full Data Report. June 2008.

[51] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990. ISBN: 0-471-92420-2.

[53] Oya I. Tukel, Walter O. Rom, and Sandra Duni Eksioglu. "An investigation of buffer sizing techniques in critical chain scheduling". In: *European Journal of Operational Research* 172.2 (July 2006), pp. 401–416.

[55] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004. ISBN: 0321245628.

[56] Bente Anda et al. "Estimating Software Development Effort Based on Use Cases - Experiences from Industry". In: *4th International Conference on the UML*. Lecture Notes in Computer Science. Springer, 2001, pp. 487–502. ISBN: 3-540-42667-1.

[58] Christoph Schwindt. *Resource Allocation in Project Management*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2005.

[59] G. Ruhe and M.O. Saliu. "The art and science of software release planning". In: *Software, IEEE* 22.6 (Nov.-Dec. 2005), pp. 47–53. ISSN: 0740-7459.

[60] Aybüke Aurum and Claes Wohlin. "The fundamental nature of requirements engineering activities as a decision-making process". In: *Information & Software Technology* 45.14 (2003), pp. 945–954.

[61] Barbara Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger. "Case Studies for Method and Tool Evaluation". In: *IEEE Software* 12.4 (1995), pp. 52–62. ISSN: 0740-7459.

[63] F. Budinsky et al. *Eclipse Modeling Framework*. Addison-WesleyProfessional, 2003.

[64] Nili Guttmann-Beck and Refael Hassin. "Approximation Algorithms for Minimum -Cut". In: *Algorithmica* 27.2 (2000), pp. 198–207.

[65] J.D. Herbsleb and A. Mockus. "An empirical study of speed and communication in globally distributed software development". In: *Software Engineering, IEEE Transactions on* 29.6 (June 2003), pp. 481–494. ISSN: 0098-5589.

[66] Balasubramaniam Ramesh et al. "Can distributed software development be agile?" In: *Commun. ACM* 49.10 (2006), pp. 41–46. ISSN: 0001-0782.

[67] B. W. Kernighan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graphs". In: *The Bell system technical journal* 49.1 (1970), pp. 291–307.

[68] P. Fjallstrom. *Algorithms for graph partitioning: A survey*. 1998.

6 További hivatkozások

[33] Agile Alliance. *What is Agile Software Development?* Accessed on 27 Feb 2010. 2011. URL: http://www.agilealliance.org.

[52] Multilogic Ltd. *Multilogic Homepage*. 2008. URL: http://www.multilogic.hu.

[54] Mathworks Inc. "Matlab Homepage". In: (2008). Accessed on 28 May 2008.

[57] Microsoft Corp. *Microsoft Office Project 2003 SDK*. Accessed on 4 Sept 2007. 2003.

[62] Eclipse. *Eclipse Homepage*. URL: http://www.eclipse.org.

[69] OptiXware Llc. *OptXware Homepage*. URL: http://www.optxware.com.

[70] Hungarian Post Corp. *Hungarian Post Homepage*. 2008.

[71] Hungarian Tax and Financial Control Administration (APEH). *APEH Homepage*. Accessed on 4 Sept 2008.

[72] Prolan Corp. *Prolan Homepage*. Accessed on 4 Sept 2008. URL: http://www.prolan.hu.

[73] Multilogic Ltd. *Pythia Homepage*. URL: http://www.multilogic.hu/pythia/.

[74] Microsoft Corp. *Microsoft SharePoint 2007*. Accessed on 4 July 2008. 2008. URL: http://www.microsoft.com/sharepoint/.

[75] Budapest University of Technology, Dept. of Measurement Economics, and Intelligent Systems Research Group Information Systems. *ISRG Homepage*.