HADMÉRNÖK

XII. Évfolyam 1. szám – 2017. március

AZ AGILIS SZOFTVERFEJLESZTÉS ALKALMAZÁSÁNAK LEHETŐSÉGEI A MAGYAR HONVÉDSÉG SZÁMÁRA

POSSIBLE APPLICATIONS OF THE AGILE SOFTWARE DEVELOPMENT FOR THE HUNGARIAN DEFENCE FORCES

GEREVICH János

(ORCID: 0000-0001-7236-4514)

gerevich.janos@agilexpert.hu

Absztrakt

Még a közelmúltban is a hadászat volt az egyik inkubátora a különböző műszaki fejlesztéseknek, előbb jelentek meg egy új technológia katonai alkalmazásai és csak utána következhetett a civil használatba vétel. Példa lehet erre a GPS vagy akár az Internet is. Napjainkra ez a trend változni látszik, vannak olyan műszaki területek, melyek újításait a honvédelem és a hadászat már csak a kifejlődésüket követően veszi át. Egy ilyen terület a szoftverfejlesztés is, jelenleg rengeteg eszközre és architektúrára készülnek a szoftverek különböző programozási nyelveken, szerteágazó technológiai alapokon, funkciók sokaságát megvalósítva. A számtalan új és gyorsan változó követelményhez az informatikának is alkalmazkodnia kellett, új szoftverfejlesztési módszerek jelentek meg, melyek gyűjtőnév gyanánt az agilis szoftverfejlesztési módszertan nevet kapták. Az új módszerek alkalmazásában a sok éves múltra visszatekintő nagy szervezetek általában szkeptikusak, jelen cikk ezt a problémát tárja fel, majd ezt követően egy lehetséges megoldást kínál rá.

Kulcsszavak: *szoftvertechnológia, projekt-módszertan, Scrum, szoftverfejlesztés, szoftvertervezés, automatizált tesztelés, hadiipar, pályázati kiírás*

Abstract

Even in the recent past the army was the incubator of various technical developments, usually a new technology appeared in military applications first and the civil usage just followed it later. For example, it could be the GPS or the Internet as well. Today, this trend seems to be changing. There are innovations in different technical areas which have been adapted by the defence and the military after their inventions only. The software development is such an area, currently uncountable functions are made for wide variety of devices and architectures in different programming languages based on several technologies. The information technology had to adjust to the new and rapidly changing requirements, therefore several new software development methods were born and these are called as agile software development methodologies. Usually the large organizations with long history are sceptical about the new methods; this article helps us to get to know better with this problem and gives us an insight about the possible solution for it.

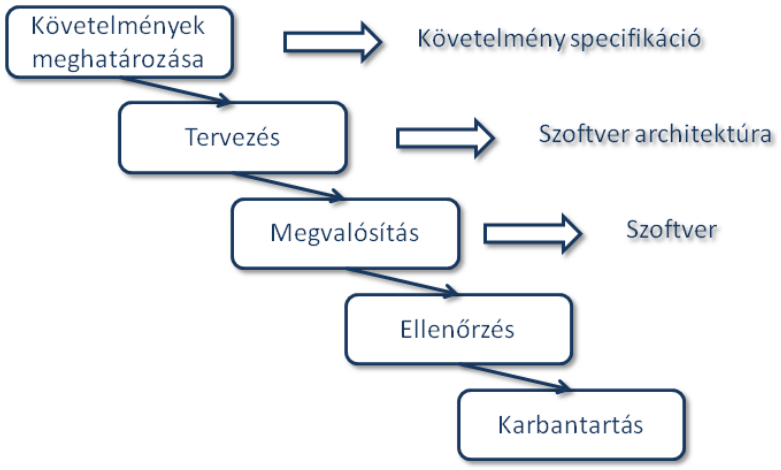
Keywords: *software technology, project methodology, Scrum, software development, software design, automated testing, defence industry, calling for tender*

A kézirat benyújtásának dátuma (Date of the submission): 2017.01.15.
A kézirat elfogadásának dátuma (Date of the acceptance): 2017.03.21.

BEVEZETÉS

Napjainkban az informatikai projektek műszaki projektként jelennek meg a köztudatban, ugyanakkor műszaki projekt egy építészeti, gépészeti és természetesen egy szoftverfejlesztési projekt is. Egy tipikus építészeti projekt lehet egy létesítmény megtervezése majd felépítése. Gépészeti projekt lehet egy fegyverfejlesztés. Az építészeti projektek során statikusok, külső és belső építészek finomítják, vizsgálják át a terveket, míg végül egy elfogadott komplett terv születik, mely alapján felépül a létesítmény. Egy gépészeti projektben hasonlóan először a tervek készülnek el, majd a tervek alapján készül el az eszköz, melynek tulajdonságait laboratóriumi körülmények között vizsgálják. A felmerülő hibákat a tervek finomításával és újabb prototípusok készítésével javítják.

A fenti példákban kézzelfogható, fizikailag létrejövő végeredményt kapunk, az építészet kapcsán egy kész épületet, egy eszközfejlesztés esetében egy prototípust. Az informatikai projektek a kézzelfoghatóság szempontjából szerteágazóak lehetnek. Egy szerverpark beüzemelése karakterisztikáját tekintve építészeti projekthez hasonlítható, tervezés, megvalósítás, átadás (tesztelés). Egy informatikai rendszer üzemeltetése hasonlítható egy gépészeti projekthez, ahol létező fizikai és létező szoftver rendszereket hangolnak össze, szabnak testre, majd az együttes működésüket tesztelik, egyfajta prototípusként. Az informatika egy különleges és izgalmas ágazata a szoftverfejlesztés, mely jellegét tekintve eltér az imént említett informatikai projektektől. Az informatikában is évtizedeken át széleskörűen alkalmazták a vízesés modellt [1], amely az alábbi lépéseket azonosítja a projekt megvalósítása során:



1. ábra Eredeti vízesés modell [2]

A fenti statikus modell feltételezi, hogy minden produktív lépés végterméke egy teljes értékű dokumentum vagy megoldás, melyet a későbbiekben már nem kell módosítani. Ez a modell megfelelő lehet egy szoftver alap architektúrájának vagy prototípusának kialakításához, de nehézkesen vagy egyáltalán nem alkalmazható egy változó szoftverre, mellyel kapcsolatban folyamatosan igények merülnek fel a felhasználók, illetve a megrendelő részéről.

Úgy gondolom, hogy itt fontos megjegyezni, hogy minden szoftver esetében két különálló egységet azonosíthatunk: architektúra és funkcionalitás. Az architektúra kialakításhoz a vízesés modell célszerű lehet, azonban manapság ezek a feladatok ritkábban fordulnak elő, mert az informatikai ipar kész architektúrákat biztosít, ezekre példa a .NET Framework [3] vagy a Java Enterprise Edition [4].

Az architektúra kapcsán fontos előre tudni, hogy milyen fizikai eszközökön fog az adott szoftver futni, milyen számítási teljesítménnyel lehet kalkulálni, milyen fizikai környezetben fogják alkalmazni a kifejlesztendő megoldást. Ezen az alkalmazási szinten elengedhetetlen a lépcsőzetes tervezés – megfelelő választás lehet a vízesés modell, azzal a megkötéssel, hogy az elkészülő végtermék egy futtató környezet vagy egy keretrendszer, mely a későbbiekben alapja lehet bármilyen alkalmazásnak a legegyszerűbb oktatói programok és a nagyvállalati szoftverek [5] széles spektrumán. Napjaink alkalmazott szoftverei a két véglet közötti számtalan egyedi alkalmazásra mutatnak példát.

Ha az architektúrát fixnek tekintjük, akkor az adott architektúra által biztosított virtuális térben az egyedi alkalmazások matematikai leképezések halmazaként is tekinthetőek, melyekről akár matematikai módszerekkel is eldönthető, hogy megoldják-e a kitűzött feladatot. [6] Nagyon fontos megjegyezni, hogy az egyedi alkalmazások implementálása során már csak információval dolgoznak a szoftverfejlesztők, ez egy jelentős különbség más műszaki projektekhez képest. Az iménti felismerésből az következik, hogy valójában hatalmas szabadsággal rendelkeznek a programozók a szoftvert futtató virtuális környezetben. Ezért is különösen fontos a megfelelő módszerek, szabályok, technológiák alkalmazása a szoftverfejlesztési projektek során, és ez alól nem lehet kivétel a Magyar Honvédség sem.

A vízesés modell első ránézésre szimpatikus választás lehet katonai alkalmazási területeken. Követelmények meghatározása, funkciók implementálása, tesztelés és végeredményként létrejön az eredetileg kigondolt szoftver. Ha azonban alaposabban megvizsgáljuk a katonai alkalmazási területeket, legyen az béke vagy tábori rendszer, ha felhasználókkal, jogosultságokkal, adatbázisban tárolt adatokkal rendelkezik a kifejlesztendő szoftver, akkor az egy egyedi alkalmazás. A katonai alkalmazások esetében különös tekintettel oda kell figyelni a biztonsági, jogosultsági, megbízhatósági, fenntarthatósági, hibatűrési, katasztrófatűrési szempontoknak, azonban az informatika szemszögéből a katonai szoftverfejlesztés a nagyvállalati szoftverfejlesztés, ezen belül az egyedi alkalmazásfejlesztés egy speciális ágazata és ugyanazokkal a problémákkal kell megküzdenie, amelyekkel a civil szoftverfejlesztésben is találkozhatunk.

AZ AGILIS SZOFTVERFEJLESZTÉS BEMUTATÁSA

A specifikációs erőforrások végessége

Napjaink szoftverei iránt egyre több és több igény merül fel minden alkalmazási területen, nem kivétel ez alól a katonai célú alkalmazás sem. Az igények túlnyomó részben a megvalósítandó üzleti funkciókra vonatkoznak. Az igények megnövekedett száma egy statikus projektmódszertan számára elhúzódó specifikációs időszakot jelent, ráadásul nagyon speciális szaktudást igényel.

A vízesés modell esetében a fejlesztés megkezdéséhez a teljes specifikáció előállítása szükséges, mivel a fejlesztés és a tesztelés alapja ez a dokumentum lesz. A legnagyobb probléma a hagyományos statikus specifikációs módszerekkel, hogy kizárólag írott formában emberi nyelven állítanak elő egy szabályrendszert, nincs direkt kapcsolat a szoftver virtuális világával. A követelményrendszert értelmezni kell és az értelmezés folyamata hibákat rejthet, a szoftver bonyolultsága hatványozottan növekszik a funkciók számához viszonyítva.

A modern kor szoftverei esetében fontos szempont a felhasználói élmény, a letisztult grafikus felület, melyekhez előre elkészített képernyőtervekkel kell rendelkezni egy statikus fejlesztési módszertan esetében is. Sajnos a felületek működését nem lehet meghatározni a rendszer által kezelt adatok nélkül, teljes körű követelményrendszerre van szükség, amit megrendelői oldalon elvégezni nehézkes feladat, későn jelennek meg az első eredmények, csak a fejlesztési, tesztelési időszak végén lehet visszacsatolás. Sok éves szoftverfejlesztői,

rendszerszervezői tapasztalattal rendelkező szakemberek számára is nehézkes ez a fajta specifikációs módszer.

Az agilitás egy válasz az új igényekre

Az üzleti szférában, a civil szoftverfejlesztésben immár 15 éve elindult egy kezdeményezés, mely új alapokra helyezi az informatikai rendszerek, különösen az egyedi szoftverek előállítását. Az a felismerés, hogy a követelményrendszer alacsony szintű meghatározásában a fejlesztőcsapat is részt vehet, ha egy megfelelő üzleti tudással – lehet az hadászat is – rendelkező szakember részesévé válik a fejlesztési folyamatnak új dimenziókat nyitott a szoftverek előállításában. A Kiáltvány az agilis szoftverfejlesztésért [7] című dokumentumot 2001-ben fogalmazták meg korunk vezető szoftvermérnökei, melyhez 12 agilis alapelvet [8] azonosítottak. Az első négy alapelv üzenete az, hogy a megrendelői elégedettséget a folyamatosan jó minőségben előállított szoftverrel lehet elérni, mely megköveteli az aktív megrendelői és beszállítói együttműködést. Ezt követően a kiegyensúlyozott projektkörnyezet és az ütemezett előrehaladás fontossága kerül külön kiemelésre, ezekre azért van szükség, hogy a feladat végrehajtását végzők nyugodt és előre tervezhető munkakörnyezetben tudjanak dolgozni, így egyfajta plusz motivációt kapjanak. Az utolsó négy alapelv a kreatív gondolkodást és a mérnöki szabadság fontosságát hangsúlyozza, abból kiindulva, hogy egy motivált közegben, ahol a megvalósítást végzők magukénak érzik a szoftvert, ott lehetőséget kell biztosítani a saját ötleteik, koncepcióik megvalósítására, amennyiben azok a projekt célját szolgálják.

Az agilis kiáltvány nem definiál konkrét módszertant, csak az alapelveket fekteti le, több agilis módszertan is létezik, köztük az egyik legelterjedtebb a Scrum, amely az alábbi szerepköröket, szereplőket és feladatokat azonosítja:

Product owner / terméktulajdonos – „A projekt vezetésének központi szereplője, aki döntéshozási jogokkal is fel van ruházva, ezáltal egyszemélyes döntéshozóként felelős a kifejlesztendő funkciókért és azok elkészítésének sorrendjéért." [9; 15. o.] Ezen felül a terméktulajdonos további feladatai:

1. Kapcsolattartás az ügyféllel
2. Az üzleti és technológiai követelmények összefogása
3. A minőség ellenőrzése a fejlesztés során
4. Együttműködés a Scrum további szereplőivel (ScrumMaster, Fejlesztőcsapat)

ScrumMaster – „Mindenki számára igyekszik bemutatni és felölelni a Scrum értékeit, alapelveit és gyakorlatát. Egyfajta edzőként lép fel, vezetői szintű folyamatokat határoz meg és segíti a Scrum csapatot, valamint a szervezet fennmaradó részét saját teljesítményének javításában" [9; 16. o.] A ScrumMaster további feladatai:

1. Felmerülő belső és külső problémák megoldása
2. A Scrum hatékonyságának növelése
3. A Scrum-hoz szükséges környezet megszervezése

Fejlesztőcsapat – „A tradicionális szoftverfejlesztési megközelítés különböző szerepköröket tárgyal, például architekt, programozó, tesztelő, adatbázis adminisztrátor, grafikusfelület tervező és így tovább" [9; 16. o.] A fejlesztőcsapat feladatai a Scrum-on belül:

1. Tervezési feladatok ellátása
2. Fejlesztési feladatok ellátása
3. Tesztelés elvégzése
4. Önszerveződés (agilis megközelítés)

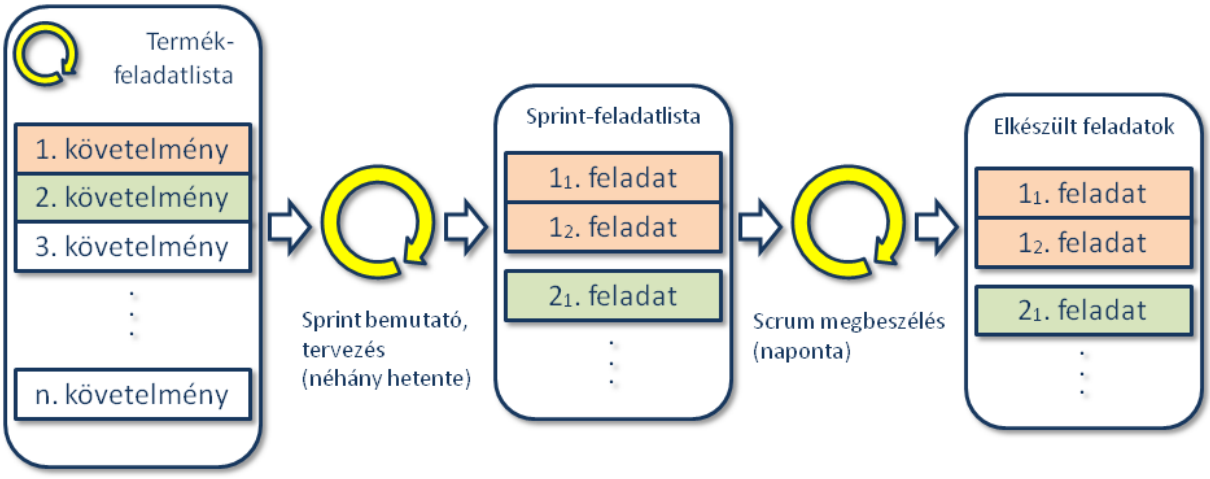
A szoftvertechnológiával foglalkozó irodalomban az ideális szoftverfejlesztő csapat résztvevőinek száma alapvetően eltérő, véleményem szerint egy erős technológia háttérrel rendelkező fejlesztés során ez a szám körülbelül 3-6 fő, tekintettel arra, hogy bizonyos feladatok automatizálhatóak, például: tesztelés, release management. Fontos megemlíteni a Scrum esetében, hogy nem javasolt az alkalmazása a 6-9 főnél nagyobb fejlesztőcsapatok esetében, ilyenkor újabb Scrum csapatok létrehozása ajánlott.

A Scrum egy iteratív szoftverfejlesztési módszertan [10], melynek kiindulási alapja a product backlog avagy a termék-feladatlista, mely a fejlesztés sorrendjében együttesen tartalmazza a termék előállításához szükséges még visszamaradt feladatokat. A fejlesztés elkezdésének pillanatában tartalmaznia kell az addig azonosított összes követelményt. A módszertan lehetővé teszi a követelmények és a sorrend megváltoztatását, feladatok összevonását vagy szétbontását.

Az agilis kiáltvány 3. pontja szerint minél gyakrabban kell működő szoftvert szállítani, a Scrum értelmezésében ez az alapelv a fejlesztések sprintekbe szervezésében nyilvánul meg. Egy fejlesztési sprint egy rövid fejlesztési időszak (általában 1-3 hét), melynek a végén működő szoftvert kell előállítania a fejlesztőcsapatnak, ahol a sprint feladatai a product backlog tetején szereplő legmagasabb prioritású feladatokból kerülnek ki.

Lényeges különbség egy hagyományos módszertanhoz viszonyítva, hogy a sprint-feladatlista előállítása a terméktulajdonos és a fejlesztőcsapat közös feladata, melyet egy erre a célra szánt körülbelül 4 órás közös tervezésen tesznek meg. A tervezések során, a termék-feladatlista tetején szereplő követelményeket a fejlesztőcsapat értelmezi, majd kisebb feladatokra bontja, egyfajta megvalósíthatósági tervet készítve. A fejlesztési sprint során ezeket a konkrét feladatokat kell megoldania a fejlesztőcsapatnak.

Az agilis kiáltvány 1. pontja szerint az ügyfél elégedettségét működő szoftverrel kell kivívni, ennek módja a Scrum módszertan szerint az előző fejlesztési sprint eredményeinek bemutatása az ügyfél képviselőjének sprintről-sprintre. Amennyiben az ügyfél nem tud részt venni a demonstráción, a bemutató megtartása akkor is hasznos, mert így a fejlesztőcsapat minden tagja láthatja az előző időszak eredményeit.



2. ábra A Scrum folyamata (saját szerkesztés)

Ez a módszer önmagában nem jobb, mint a vízesésmodell és félreértelmezhető, valamint komoly kérdéseket vet fel, hogy valóban az a szoftver fog-e elkészülni, amelyet a megrendelő szeretett volna, mert a követelmények és a megfogalmazott elvárások folyamat közben változhatnak. Ha a megrendelő nem veszi ki a részét a rá háruló feladatokból, nem bocsájt rendelkezésre egy terméktulajdonost a projekt időtartamára, akkor a Scrum csapat tehetetlenné válik megfelelő támogatás és követelményrendszer hiányában.

Természetesen a megfelelő szabályok betartásával a Scrum egy nagyon hatékony módszer is lehet, mellyel kimagasló eredmények érhetőek el. A módszer alkalmazásának alapfeltétele a korszerű technológiai háttér, valamint a fejlesztőcsapat, a terméktulajdonos és a megrendelő agilis hozzáállása, ami az addig megszokott szervezeti berendezkedést alárendeli a projekt sikerének.

Példának okáért, ha egy katonai bevetéseket szimuláló szoftvert szeretnénk kifejleszteni, akkor szükséges egy olyan katonai szakértő, aki az adott terület fogalomrendszerét, terminológiáját, szabályait átadja a fejlesztők számára és folyamatosan részt vesz a fejlesztésben. Így az adott szakértő a kifejlesztendő szoftver terméktulajdonosává válik, ezt követően már kapcsolattartóként képviselni tudja a projektet a megrendelő felé. Az sem okoz gondot, ha terméktulajdonos a megrendelő oldaláról kerül a projektbe, de nagyon fontos, hogy a fejlesztés időtartamára a fejlesztőcsapattal napi szinten együtt dolgozzon.

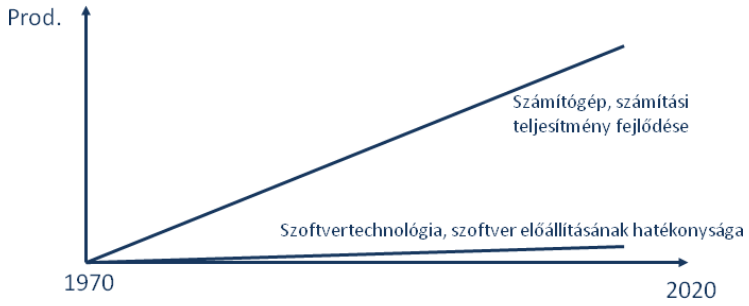
EVOLÚCIÓS SZOFTVERFEJLESZTÉS - A SZOFTVER, MINT ÉLŐ ENTITÁS

Az iteratív szoftverfejlesztési módszertanok lényege a rövid időszakonkénti release (kiadott verzió) előállítása. A helyesen alkalmazott Scrum esetében ez az időtartam néhány egymást követő fejlesztési sprintet jelent. A gyakorlatban egy teljes iteráció, melynek a végeredménye egy kiadható verzió egy néhány hetes, esetleg néhány hónapos időintervallumot ölelhet fel. A ciklikus demók következtében a fejlesztés során is közel stabil állapotú szoftvereket kapunk, melyek lehetőséget biztosítanak a folyamatos újratervezésre, a gyors változtatásra, ami nagyon hasznos lehet minden alkalmazási területen.

Az iteratív szoftverfejlesztés, mint projektmódszertan önmagában kevés, a jobb szoftvertechnológiai teljesítmény elérése érdekében a megfelelő szoftverfejlesztést támogató eszközök szabályozott alkalmazása is szükséges. A követelményeket és a sprint során azonosított fejlesztési feladatokat egységesen kell tudni kezelni, valamint megfelelő számú fejlesztési, tesztelési és fejlesztést támogató környezet szükséges, ahhoz hogy a Scrum a gyakorlatban is sikeresen megvalósítható legyen. A fejlesztést támogató rendszerek felhasználói szintű ismerete a terméktulajdonos, a ScrumMaster és a fejlesztőcsapat számára is kötelező.

A korszerű technológiák alkalmazása a fejlesztőket is új kihívások elé állítja. Ha a megrendelői oldalról plusz erőforrásnak tekinthető a szakértő beépítése a fejlesztő csapatba, akkor a fejlesztők oldaláról is extra energiaráfordítás az azonosított funkciókra, konkrét fejlesztési feladatokra vonatkozó automatizált tesztek elkészítése. Az speciális környezetek közül az egyik legfontosabb a tesztek automatizált futtatását lehetővé tévő tesztfuttató környezet.

A folyamatosan gyorsuló számítógépek, a növekedő számítási kapacitás lehetővé teszi, hogy a kifejlesztett szoftvert tesztesetek tízezreivel, tesztforgatókönyvek százaival fedjük le és folyamatosan teszteljük a magas minőség elérése érdekében. Egy teszt elkészítése ugyan sok időt igényel a fejlesztők részéről, de ezt követően az elkészített programkód szerves részévé válik a szoftver fejlesztői környezetének. A rendszer forráskódjában történő bármely változásról a tesztfuttató-környezetek azonnal visszajeleznek, így a szoftver a virtuális térben egy élő entitássá válik, melynek egyfajta mérhető minősége lesz. Az következő ábra szemléltetve hasonlítja össze az elmúlt évtizedekben végbement számítási teljesítménynövekedést és a szoftvertechnológia fejlődését.



3. ábra Hardverek sebességének fejlődése, szoftvertechnológia hatékonyságának fejlődése (saját szerkesztés)

Amíg a gépekről elfogadja a tudományos világ, hogy a Moore törvény [11] alapján az egységárra jutó számítási teljesítmény egyre gyorsabban növekedik, vagyis az integrált áramkörökben a tranzisztorok száma 18 havonta megduplázódik, addig az emberek, konkrétan az emberi agy ugyanolyan számítási képességekkel rendelkezik, mint 50 évvel ezelőtt.

A szoftvertechnológia hatékonyságán, ezáltal a költségeken, kizárólag az új projektmódszertanok és a fejlesztést támogató eszközök bevonása javíthat. Míg a korszerű számítógépek és a fejlesztést támogató eszközök beszerzése egyszeri költségként jelenik meg egy fejlesztési projektre levetítve, addig a fejlesztési projekt szereplőinek bérköltsége folyamatos kiadás. Ebből a megállapításból következik, hogy hosszú távon luxus a szoftverfejlesztési projektek egy kalap alá vétele az egyéb informatikai projektekhez, mert a leghatékonyabb módszerek alkalmazásához más projektstruktúrára és eszközparkra van szükség.

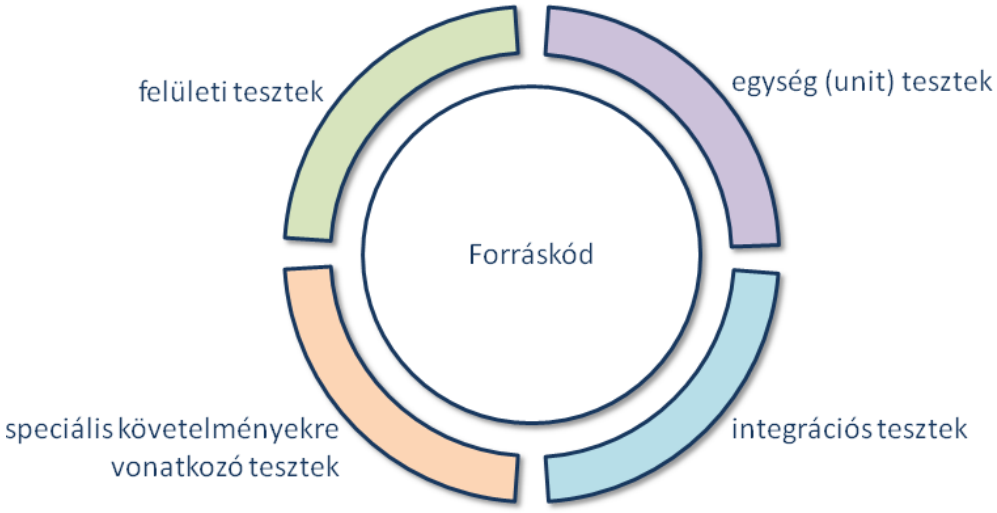
Ahogy a harcászatban újabb és újabb eszközök jelennek meg, úgy a szoftvernek is gyorsan le kell tudnia követni a változásokat. Ha van egy harcszimuláló szoftverünk, melyet hagyományos harcászati eszközökre fejlesztettünk ki a vízesés modellt alkalmazva, akkor komoly fejtörést okozhat a drónok bevezetése a rendszerbe, mert a követelményeket külön kell meghatározni. Célszerű, ha a specifikálást azok végzik, akik az eredeti specifikációt létrehozták, holott lehet, hogy már nem is dolgoznak ott, ahol az eredeti dokumentum elkészült. A fejlesztést követően is problémákba ütközünk, újra kell tesztelni manuálisan az egész rendszert.

Egy agilisan előállított szoftverben ezt a módosítást a terméktulajdonos és a fejlesztők együttesen oldják meg. Az új követelmény elemzését, apró feladatokra lebontását együttesen egyszerűen el tudják végezni, meghatározzák az új és megváltoztatandó használati eseteket, majd ezt követően változtathatnak a rendszer produkciós kódján. Az eddigi működést lefedő automatizált tesztek egyértelmű visszajelzést adnak a lefektetett követelmények és az új rendszer különbségeiről, melyekről már egyszerű eldönteni, hogy elvárt működésről vagy hibáról van szó. A hibák javítását követően a rendszer már képes kezelni akár a drónokat is egy szimulált csatában.

A módszertan tovább csiszolható, oly módon, ha a fejlesztők először a teszteket készítik el és utána futtatják meg a szoftvert. Ebben az esetben a virtuális téren belül a kódnak az a része éli túl a változtatást, mely a drónra is helyesen működött és a „rossz” kód kipusztul a rendszerből. A kipusztulást a fejlesztők „okozzák”, mikor kibővítik a rendszer működését és a drónra helytelenül működő kódot átalakítják. Ez a fejlesztési módszertan egy merőben új, szoftverekre értelmezett evolúciós folyamatot hoz létre.

Ha az automatizált tesztek kialakításakor a harcászati szempontokat is szem előtt tartjuk, akkor ezzel a módszerrel a katonai alkalmazás egy új szintje is elérhető. Kieső számítás kapacitás kezelése, alkalmazott algoritmusok közötti váltás, DoS támadás kiszűrése, stb…

Ellenség által használt támadási technikák beépítése a tesztkörnyezetbe, ezekre válaszoló védelmi funkciók kialakítása.



4. ábra Automatizált tesztekkel lefedett forráskód (saját szerkesztés)

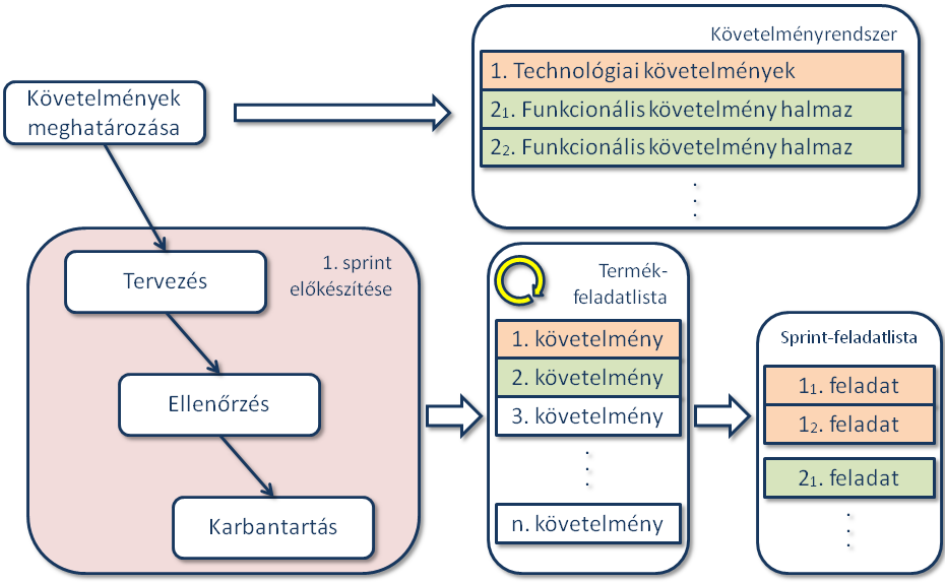
Az automatizált és a manuális tesztelés ugyanolyan fontos a sprint időtartama alatt, mert az egyik a konkrét funkciót fedi le, a másik a funkciót és a rendszer integritását vizsgálja együttesen. Utóbbi különösen akkor igaz, ha szakértő végzi a manuális tesztelést. Ez a fajta megközelítése a szoftverek előállításának aktív részvételt követel meg a megrendelőktől is, a probléma mély megértésének folyamata közös tevékenységgé válik. A módszer legnagyobb előnye az, hogy az elkészült termék a technológiai és az üzleti korlátok között is a lehetséges maximumot fogja produkálni, mert folyamatos tesztelés alatt áll üzleti és technológiai oldalról egyaránt. A módszer lehetőséget biztosít arra, hogy a széleskörű funkcionalitás ötvöződjön a megfelelő technológiai háttérrel.

A VÍZESÉS MODELL ALKALMAZÁSÁNAK LEHETŐSÉGEI A KÖVETELMÉNYEK MEGHATÁROZÁSÁRA

A modern kor szoftvere bonyolult és egyre bonyolultabbá válik, a régi és az újonnan létrejövő rendszereket össze kell kapcsolni, virtuális világ épül a szemünk láttára. Az implementáció során új tervezési módszerekre van szükség, erre lehet válasz az agilis szoftverfejlesztés. Mindezek ellenére, amikor egy újabb területet akarunk leképezni, kell egy kiindulási alap. A scrum szerint ez a product backlog, avagy termék-feladatlista, arról nem szól ez a módszertan, hogyan lehet ezt a feladatlistát elkészíteni.

A vízesés modell egy speciális alkalmazási területe lehet a termékre vonatkozó feladatlista előállítása. Ebben az esetben a cél egy olyan lista elkészítése, amely tartalmazza a leendő szoftver magas szintű áttekintését konkrét feladatokra lebontva. Az elsődleges feladat a teljes probléma feltérképezése, a kapcsolódó szakmai területek azonosítása, majd a probléma feladatcsoportokra és azon belül magas szintű feladatokra bontása. A verifikáció, a tesztelés arról szól, hogy a feladatok szétbontása valóban lefedi-e a teljes problémát. Az esetleges kérdések, problémák a tervezés magas szintjén is dokumentálhatók.

Az alábbi ábrán látható a Vízesés modell alkalmazásának lehetősége az agilis szoftverfejlesztési projektek előkészítéséhez. Ez egy speciális vízesés, mely mellőzi az implementálást, ennek oka, hogy ez a tevékenység teljes mértékben az agilis sprintekre tevődik át.



5. ábra A vízesés modell alkalmazásának lehetősége az agilis szoftverfejlesztési projektek előkészítése során (saját szerkesztés)

Ha egy katonai célú agilis projekt előkészítése zajlik, akkor ennek fényében a megfelelő dokumentációk és magas szintű tervezés megvalósítható a kapcsolódó katonai szakterületek bevonásával, ezt követően a kialakított követelményrendszerrel elindulhat a fejlesztés. Ha a fejlesztés folyamán újabb kérdések merülnek fel, akkor a karbantartás folyamatlépésben a feltárt kérdések tisztázhatóak és beépíthetőek a követelményrendszerbe, azaz a termék-feladatlistába. A megoldás nagy előnye, hogy egy szűkebb projektcsapat dolgozik a sprintek megvalósításán, ugyanakkor a fejlesztés teljes életciklusa alatt minden követelményhez a kapcsolódó szakterület megszólítható, a kérdések tisztázhatóak, így a fejlesztés végére a valódi követelményrendszer áll össze. Ha a dokumentációk karbantartása folyamatos, akkor az implementált szoftver és a hozzá kapcsolódó dokumentumok teljes mértékben lefedik egymást.

KATONAI CÉLÚ AGILIS SZOFTVERFEJLESZTÉS – MILITARY SCRUM

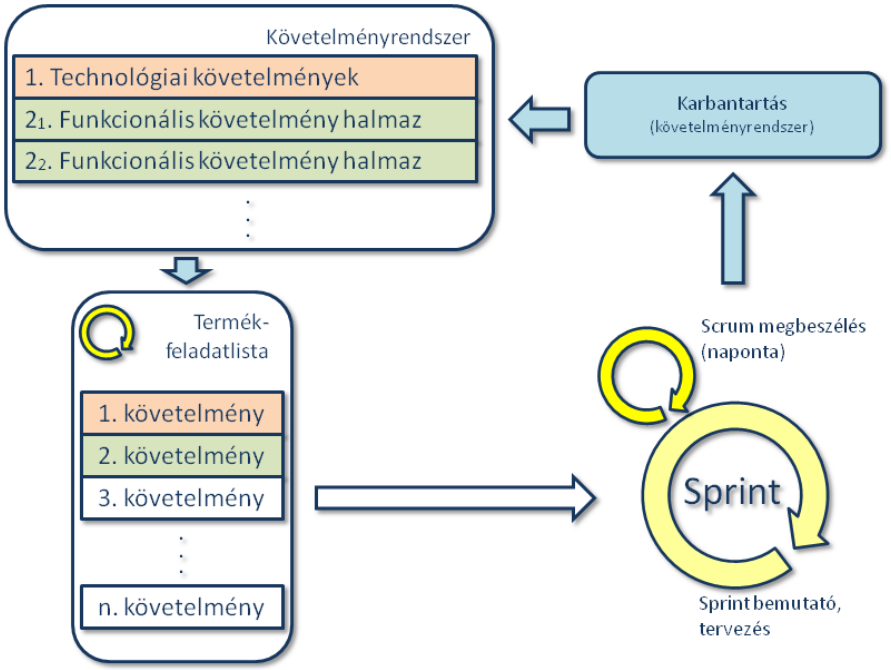
A termékre vonatkozó feladatlista megléte előfeltétele az agilis szoftverfejlesztési projekteknek. A Scrum módszertan összességében annyit mond a tervezésekről, hogy a követelményeket fejlesztési feladatokra kell szétbontani. Itt az a kérdés, hogy egy nagyobb projekt esetében elegendő-e csak a termék feladatlistájának karbantartása és frissítése. Minden alkalmazási területen lényeges minden részlet, ezek megértése, felkutatása már a tervezések előtt is szükséges. A statikus módszertanok válasza erre, hogy a fejlesztés előtt körbe kell járni minden területet. A gyakorlat azt mutatja, hogy az implementáció során is nagyon gyakoriak a rendszer teljes egészére vonatkozó, működést érintő kérdések, melyekre a válasz fogós lehet még a szakértők számára is.

Ha az implementálást agilis eszközökkel végezzük, akkor az evolúciós szoftverfejlesztés alkalmazása lehetővé teszi a gyors visszajelzést a specifikálást végzők számára, és ebből kifolyólag egyfajta kutatási tevékenységgé válik egy adott sprint-re vonatkozó részletes specifikáció előállítása, mert a témával kapcsolatos tárgyi tudás és az automatizált tesztkörnyezetben létező szoftver viselkedése, vizsgálata alapján konkrétan meghatározhatóak a követelmények a következő 1-2 sprintre.

Grafikus felülettel rendelkező szoftverek esetében a képernyőtervek és az azok működését leíró magyarázó szövegek megfelelő kiindulási pontnak tekinthetőek egy tervezéshez.

Felülettel nem rendelkező szoftverek esetében is a keletkező leírások elegendőek az implementálás megkezdéséhez.

A rendszeres sprinttervezések során a fejlesztők részéről nagyon hasznos és lényegi észrevételek hangozhatnak el az implementációval kapcsolatban, melyek a konkrét fejlesztési feladatok meghatározásánál fontosak lehetnek. Ebben a 4 órás tervezésben a cél az, hogy jól átgondolt alacsony szintű követelmények és megvalósítási tervek szülessenek, ám az itt feltárt összefüggések visszahathatnak a teljes követelményrendszerre is. Nem szabad elfelejteni, hogy a tervezést végző csapat része a terméktulajdonos is, aki szakmai szemmel vesz részt ezeken a megbeszéléseken. Az ő elsődleges feladata a feltárt problémák értelmezése, feltárása. A megoldások hatással lehetnek az egyes követelményekre vagy extrém esetben a teljes követelményrendszerre is. Az is elképzelhető, hogy egyéb szakmai területek bevonása szükséges az adott kérdés tisztázásához.



6. ábra Az agilis szoftverfejlesztési sprintek visszahatása az eredeti követelményrendszerre (saját szerkesztés)

A gyakorlatban ez a módszer azt jelenti, hogy a sprint időszaka alatt a következő sprint előkészítő tervezése is megvalósulhat, valamint a termékre vonatkozó feladatlista vezetése, karbantartása is ekkor zajlik. A fejlesztéssel párhuzamosan a felkészülés megvalósítható a soron következő sprintre.

Az előzőekben vázolt projektmódszertanban a fejlesztési feladatok előkészítéséhez 4 szintű tervezési tevékenység szükségeltetik. Példának okáért a megrendelő legyen a Magyar Honvédség. Az alábbi táblázat mutatja a felelősségi körök alakulását a tervezési tevékenységek során.

	Követelményrendszer meghatározása	**Termékre vonatkozó feladatlista kialakítása és karbantartása**	**Sprint-feladatlista meghatározása**	**Tesztforgatókönyvek elkészítése**
MH szakterület	x	x		
Terméktulajdonos	x	x	x	x
Fejlesztőcsapat			x	x

1. táblázat Tervezési szintek és felelősségi körök (saját szerkesztés)

1. *Követelményrendszer meghatározása* – a szoftverfejlesztési projekt első lépése, mely a megrendelő és terméktulajdonos közös feladata. Itt történik a magas szintű követelmények meghatározása, feltárása, strukturálása. Ennek a tervezési szakasznak a feladata a megfelelő dokumentációk előállítása, melyek tartalmazzák a követelmény-halmazokat és az azonosított felelősségi köröket.
2. *Termékre vonatkozó feladatlista kialakítása és karbantartása* – ez a tevékenység két részre bontható szét. Az első lépésben a megrendelő teljes körű részvétele is szükséges, mert ekkor határozzák meg a termék-feladatlista első változatát, ezt követően már a terméktulajdonos a megfelelő kommunikáció mellett egyedül is vezetheti azt. Ebben a dokumentumban jelennek meg a megrendelő által megfogalmazott új követelmények, valamint a sprint tervezések és fejlesztések során azonosított újabb feladatok.
3. *Sprint-feladatlista meghatározása* – ez a lépés a tervezések során azonosított megvalósítási tervek létrejöttét jelenti. A termékre vonatkozó egyes feladatok részfeladatokra történő szétbontása ezen a szinten valósul meg. Itt már nem szükséges a direkt megrendelői jelenlét.
4. *Tesztforgatókönyvek elkészítése* – ebben a lépésben a fejlesztők és a terméktulajdonos közösen teszteseteket dolgoznak ki, ez a legalacsonyabb szintű specifikálása a kifejlesztendő szoftvernek. A tesztesetek dokumentálását és jóváhagyását követően – ami visszavonatkoztatható a sprinttervezési dokumentációra vagy akár a teljes szoftver követelményrendszerére – az automatizált tesztek elkészíthetőek a fejlesztőcsapat által.

Az imént felvázolt módszertan nem a klasszikus Scrum, mert extra dokumentáció készítését írja elő és folytonos visszatekintő ellenőrzést vár el, azonban ezekkel a megkötésekkel egy nagyvállalati vagy államizgatási szerv is alkalmazhatja belső vagy külső fejlesztések során egyaránt.

KÖVETKEZTETÉSEK

A cikkben bemutatott lépcsőzetes tervezési és előkészítési tevékenység az államapparátus működésére jellemző munkamódszer, mely teljes mértékben alkalmazható az agilis szoftverfejlesztési projektek előkészítése során. Ha az állam a megrendelésen túl részesévé válik az agilis sprinteknek, megjelenik a rendszeres bemutatókon, akkor egy átlátható folyamat során kerül kialakításra a megrendelt szoftver, melynek végső formája még a fejlesztés során módosítható, változtatható.

Ha a megfelelő katonai szempontok is szerves részévé válnak a tervezésnek és a folyamatos visszatekintésnek, akkor tetszőleges katonai szakterület által is alkalmazható a Scrum módszertan speciális továbbfejlesztett változata a *Military Scrum*.

A Magyarországon jelenleg hatályos jogszabályokból kifolyólag a Magyar Honvédség kizárólag közbeszerzési eljárás során szerezhet be egyedi szoftvereket. Ez az eljárás a szoftverfejlesztést a műszaki projektekhez sorolja és ugyanabban a mederben kezeli. Ez azt eredményezi, hogy a pályázati kiírások során olyan követelményrendszer jelenik meg, mely nem adaptálható egy agilis szoftverfejlesztési projekt számára. Mégis mi lehet a megoldás arra, hogy az agilis szoftverfejlesztés, mint modern megközelítés meg tudjon jelenni az állami megrendelésekben?

Kézenfekvő válasz lehet a problémára a projektek kiírásának két részre bontása:

1. Prototípus előállítása
2. Teljes rendszer implementálása

A prototípus előállítása többlet ráfordítást igényel a pályázók részéről, azonban ebben az esetben a megrendelő kap egy limitált funkcionalitású működő szoftvert a pályázat első körének végére. A módszer előnye, hogy a terjedelmes sablon dokumentációk helyett a prototípus előállításának menetére vonatkozó dokumentumok és egy működő végtermék áll elő, ezzel valódi döntési alapot biztosítva a megrendelő számára.

A projekt időtartamára és költségvetésére mind megrendelői és pályázói oldalon is megalapozottabb becsléseket készíthetnek az elkészült prototípusok tapasztalatai alapján. Természetesen a győztes kihirdetésénél az ajánlati áron túl szempontnak kell lennie az adott cég addig végrehajtott agilis projektjeinek száma és az agilis projektek során használt dokumentációs, tervezési és fejlesztési módszertanok bemutatása.

A jogalkotók részéről érdemes lenne a szoftverfejlesztési projekteket külön kalap alá venni és egy korszerű szabályozást kidolgozni erre a területre vonatkozóan, mert kevesebb pénzből hatékonyabb és jobb minőségű szoftvereket vásárolhatna az állam a szoftvergyártóktól.

FELHASZNÁLT IRODALOM

[1] Waterfall model, Wikipédia
https://en.wikipedia.org/wiki/Waterfall_model (letöltve: 2016. 12. 20)

[2] ROYCE, W.: Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, 26 (1970) p. 329.

[3] .NET Framework
https://www.microsoft.com/net (letöltve: 2016. 12. 20)

[4] Java Enterprise Edition
http://www.oracle.com/technetwork/java/javaee/overview/index.html
(letöltve: 2016. 12. 20)

[5] Enterprise software, Wikipédia
https://en.wikipedia.org/wiki/Enterprise_software (letöltve: 2016. 12. 20)

[6] FÓTHI Á.: Bevezetés a programozáshoz, harmadik, javított kiadás (egyetemi jegyzet), Budapest, ELTE IK, (2012) 30-35. o.
http://people.inf.elte.hu/bzsr/progmod2/konyv.pdf (letöltve: 2016. 12. 20)

[7] Kiáltvány az agilis szoftverfejlesztésért
http://agilemanifesto.org/iso/hu/manifesto.html (letöltve: 2016. 11. 29)

[8] Az Agilis Kiáltványt alkotó elvek
http://agilemanifesto.org/iso/hu/principles.html (letöltve: 2016. 11. 29)

[9] RUBIN K. S.: Essential Scrum. Ann Arbor, Michigan, USA, Pearson Education, Inc., 2013. pp. 15-16.

[10] LARMAN, C., BASILI, V. R.: Iterative and Incremental Development: A Brief History. In: Computer, 36 (2003. 06) pp. 47-56.

[11] MOORE, G. E.: Cramming More Components onto Integrated Circuits. In: Electronics Magazine, 38 (1965. 04. 19) No. 8. pp. 114-117.