Spring 6-2016

DALLIMI MES AGILE DHE PLAN DRIVEN

Dea Macula

Universiteti për Biznes dhe Teknologji
Departamenti për Shkenca Kompjuterike

DALLIMI MES AGILE DHE PLAN DRIVEN

Shkalla Bachelor

Dea Macula

Qershor / 2016
Prishtinë

Universiteti për Biznes dhe Teknologji
Departamenti për Shkenca Kompjuterike

Punim Diplome
Viti akademik 2013 – 2014

Dea Macula

DALLIMI MES AGILE DHE PLAN DRIVEN

Mentori: Phd, Cand. Naim Preniqi

Qershor / 2016

Ky punim është përpiluar dhe dorëzuar në përmbushjen e kërkesave të pjesshme për Shkallën Bachelor

ABSTRAKTI

Softueri ka qenë pjesë e tregut modern për shumë vite. Kemi disa metodologji të zhvillimit që përdoren në ditët e sotme. Burimet e pakta, konkurenca e ashpër si dhe kërkesat gjithnjë e në ndryshim të biznesit kërkojnë modele të zhvillimit të softuerit që janë sa më efikase dhe fleksibile. Edhe pse disa prej kompanive kanë metodologjinë e tyre përshtatëse të zhvillimit, pjesa më e madhe përodrin dy prej llojeve më të njohur siq janë: Plan Drvien dhe Agile metodologjitë. Plan Driven metodologjitë konsiderohen si formë tradicionale e zhvillimit të softuerit që përkrahin planifikimin e plotë, dokumentacion të detajuar dhe dizjan gjithëpërfshirës. Metodologjitë Agile gjatë viteve të fundit përdoren shumë nga komuniteti i inxhinierisë softuerike. Këto ndryshe nga Plan Driven përdorin cikle të shkurtëra përsëritëse dhe mbështesin shkëmbimin e njohurive në grup në vend të dokumentacionit.

Në këtë punim diplome është paraqitur krahasimi mes këtyre dy metodologjive dhe janë prezentuar nga dy procese më të përdorura nga secila metodologjië. Proceset për metodologjinë Plan Driven janë Waterfall dhe V-Model ndërsa për metodologjinë Agile janë Extreme Programming dhe Scrum. Këto procese janë spjeguar në mënyrë më të detajuar duke përfshirë të gjithë hapat e zhvillimit nëpër të cilat kalojnë ato si dhe avantazhet dhe disavantazhet e tyre.

Mirënjohje dhe falenderime

Falenderoj familjen time të ngushtë dhe të gjerë për interesimin dhe mbështetjen e tyre të vazhdushme gjatë studimeve.

Falenderoj kolegët e fakultetit dhe shoqërinë për bashkëpunimin dhe ndihmën që më kanë ofruar.

Poashtu dua të falenderoj të gjithë personelin akademik të Universitetit për Biznes dhe Teknologji, veqanërishtë Phd, Cand. Naim Preniqi, për ligjeratat dhe këshillat gjatë studimeve dhe së fundi për ndihmën dhe përkushtimin gjatë përgatitjes së punimit të diplomës.

Përmbajtja

Lista e Figurave

Lista e Tabelave:

Fjalori i Termave:

UML - Unified Modeling Language

OO - Object Oriented

XP – Extreme Programming

CRC – Class Responsibility Collaborator

1. Hyrje

Në botën moderne shumë gjëra nuk mund të zhvillohen shpejtë dhe të jenë në hap me kohën pa përdorimin e softuerit. Pothuajse të gjitha institucionet kontrollin e tyre e kanë të bazuar në sistmet kompjuterike. Gjithashtu shumica e produkteve efektive përmbanjë kompjuter dhe softuer për kontrollimin e tyre. Madje edhe në industrinë muzikore, lojërat kompjuterike, filma dhe television softueri ka një rol të rëndësishëm. Prandaj inxhineria softuerike është esenciale për funksionimin sa më efikas të shoqërisë. Inxhinieria softuerike ka për qëllim të mbështes jo vetëm programimin individual, por në terësi zhvillimin profesional të softuerëve. Derisa programimi individual përfshin teknika më të thjeshta, inxhinieria softuerike ka një fushëveprimi më të gjerë ku aplikon teknika të cilat mbështesin specifikimin e programit, dizajnin dhe evoluimin. Shumë njerëz mendojnë se softueri është vetëm një term tjetër për programet kompjuterike, por kur ne flasim për inxhinerinë softuerike mund të themi se softueri nuk përfshinë vetëm programet por edhe dokumentacionet e ndërlidhura dhe konfigurimin e të dhënave që janë kusht për funksionimin korrekt të tij. Qasja sistematike e cila përdoret në inxhinerinë softuerike quhet proces softuerik. Procesi softuerik është një sekuencë e aktiviteteve të cilat përdoren për të zhvilluar një produkt softuerik. Ekzistojnë katër aktivitete themelore të cilat janë të ngjajshme për të gjitha proceset softuerike. Ato janë:

1. Specifikimi i softuerit, ku klientët dhe inxhinierët definojnë qartë softuerin që do të zhvillohet dhe kufizimet në operimin e tij.
2. Zhvillimi i softuerit, ku bëhet dizjanimi dhe programimi i tij.
3. Validimi i softuerit, ku bëhet kontrollimi i tij për tu siguruar se është konform kërkesave të klientit.
4. Evulimi i softuerit, ku bëhet modifikimi i tij në bazë të kërkesave të klientit dhe kërkesave të tregut. [1]

Nje metodë apo proces është mënyrë e zhvillimit të punës, kurdo që ju bëni diçka jeni duke ndjekur një proces. Proceset softuerike janë komplekse dhe si të gjitha proceset e tjera kreative edhe këto mbështeten në vendimarrjen dhe gjykimin e njerëzve. Nuk ka ndonjë proces softuerik ideal, prandaj shumica e organizatave kanë zhvilluar proceset e veta. Për disa sisteme më komplekse nevojitet një proces zhvillues mirë i strukturuar ndërsa për sistemet siq janë ato biznesore që kanë kërkesa të shpeshta nevojitet një proces zhvillues më fleksibil, formal dhe

efektiv. Natyra komplekse e zhvillimit të softuerit dhe shumëllojshmëria e metodave e bëjnë krahasimin mes Agile dhe Plan Driven të vështirë.

Plan Driven janë proceset ku të gjitha aktivitetet planifikohen paraprakisht dhe progresi matet kundrejtë këtij plani. Është metoda ku përfshinë planifikimin e gjerë, proceset e kodifikuara dhe ripërdorimin rigoroz duke e bërë zhvillimin e një aktiviteti të efektshëm dhe të parashikueshëm që gradualisht maturohet drejt përsosjes. Puna fillon me nxjerrjen dhe dokumentimin e kërkesave të klientit, arkitekturën dhe dizjanin e zhvillimit si dhe inspektimin. Plan Driven përdor planifikimin dhe arkitekturën e bazuar në dizajn për të përshtatur ndryshimet e parashikuara. Kjo u mundëson dizajnerëve të organizojnë sistemin për të përfituar nga ripërodrimi i softuerit si dhe mund të ketë efekt të madh në zhvillimin rapid. Disa nga llojet e modeleve të Plan Driven janë Waterfall dhe V-Model për të cilat do të diskutohet në vazhdim të punimit.

Agile janë proceset ku planifikimi i aktiviteteve është rritës dhe është më lehtë të ndryshohet procesi duke u bazuar në ndryshimin e kërkesave të klientit. Rritjet janë të vogla pasiqë ndryshimet e reja të sistemit prezentohen tek klienti çdo dy ose tri javë. Tek këto metoda klineti përfshihet në procesin e zhvillimit për të marrë komente të shpejta në ndryshimin e kërkesave. Dokumentacioni minimizohet duke përdorur më shumë komunikimet joformale sesa takimet zyrtare me dokumente të shkruara. Gjithashtu tek metodat agile përqëndrimi është në dhënien e produktit në kohë, që plotësisht përmbush një klient. Si të tilla, pjesa shqetësuese është e fokusuar tek produkti në fjalë dhe në përgjithësi injorohen problemet që mund të ndodhin më vonë. Disa nga llojet e modeleve të Agile janë Exteme Programming dhe Scrum për të cilat do të diskutohet në vazhdim të punimit.

2. Shqyrtimi i literaturës

Në librin “Software Engineering” të Sommerville (2011) thuhet se procesi softuerik është një grup i aktiviteteve të ndërlidhura që të dërgon drejt prodhimit të një produkti softuerik. Këto aktivitete mund të përfshijnë zhvillimin e një softueri nga fillimi me anë të një gjuhe standard programuese si Java, C etj. Agile metodologjitë konsiderojnë dizajnin dhe implementimin si aktivitete kryesore në procesin e softuerit. Në të kundërtën, Plan Driven identifikon faza të veçanta në procesin softuerik, rezultatet e të cilave lidhen me fazën tjetër. Këto rezultate përdoren si bazë për planifikimin e procesit në vijim. Tek Plan Driven përsëritja ndodh brenda aktiviteteve me dokumente zyrtare që përdoren për të komunikuar mes fazave të procesit. Ndërsa tek Agile përsëritja ndodh në të gjitha aktivitetet, prandaj kërkesat dhe dizajni zhvillohen së bashku e jo të ndara. [1]

Në librin “Agile Software Develeopment Succinctly” të Huants (2015) tregohet se Agile përqëndrohet shumë tek ekipi dhe ndërveprimet mes tyre sesa tek proceset dhe mjetet. Kjo është një vlerë thelbësore e Agile. Kjo është e rëndësishme pasiqë është input nga ekipi dhe klineti që në fund projekti do të ketë suksese, sesa çfarë mjete janë përdorur. Bashkëpunimi i vazhdueshëm gjatë gjithë ciklit të zhvillimit të projektit mundëson që të gjithë të përfshihen për të ndërtuar një marrëdhënie të mirë pune në bazë të besimit. Kjo marrëdhënie pune besim-me-bazë është kyçe kur softueri ndërtohet me rritje. Kostot mund të rriten kështuqe testerët janë të nevojshëm gjatë gjithë projektit e jo vetëm në fund. Testimi është një pjesë e rëndësishme e një projekti Agile gjatë sprintave dhe përsëritjeve. Kjo ndihmon për të siguruar cilësinë e gjithë projektit, pa pasur nevojë për një fazë të gjatë dhe të paparashikueshme prove në fund të projektit. Megjithatë, kjo do të thotë se testerët janë të nevojshëm gjatë gjithë ciklit të zhvillimit të produktit, dhe kjo mund të rritë koston në mënyrë drastike. Kjo kosto shtesë kursen të holla në afat të gjatë dhe kodi është duke u testuar vazhdimisht. Duke pasur një kombinim të testimit manual dhe testimit të automatizuar është mënyra më e mirë për të arritur deri në cilësinë e produktit. [2]

Në librin “The Art of Agile Development” të Shore dhe Warden (2007) përshkruhet se metodologjija Agile përbëhet nga elemente të veçanta të quajtura praktika. Praktikat përfshijnë përdorimin e versioneve të kontrolluara, vendosjen e standardeve të kodimit dhe duke i dërguar

klientit prototipe javore. Shumica e këtyre praktikave kanë qenë rreth e rrotull për vite. Metodologjija Agile i kombinon ato në mënyra unike, duke theksuar pjesët që e mbështesin filozofinë Agile, duke anashkaluar pjesët tjera dhe përfshinë ide të reja. Rezultati është një paketë punuese mbështetëse dhe e fuqishme. Një ndër përparësitë më të mëdha të XP është që mund të eliminosh fazat e kërkesave, dizajnit dhe testimit si dhe dokumentet formale që i përcjellin. Projekteve softuerike ju duhen më shumë kërkesa, dizajne dhe testime për këtë arsye ekipet e XP punojnë në këto aktivitete çdo ditë. Vlen të theksohet që XP i kushton rëndësi bashkëpunimit ballë për ballë. Kjo është shumë efektive për të hequr pengesat në komunikim dhe keqkuptimet në ekip. Gjithashtu kjo i lejon ekipet të punojnë në të gjitha aktivitetet çdo ditë.[3]

Në librin "Scrum and XP from the trenches" të Kniberg (2015) shpjegon se Scrum nuk është një metodologji, ai është një model. Scrum nuk tregon saktësisht se çfarë duhet të bëjmë. Product backlog është zemra e Scrum, kjo është pjesa kur gjithçka fillon. Product backlog në thelb është një listë me prioritete të kërkesave, ose ngjarje, ose karakteristika. [4]

Në librin "Extreme Programming Explained" të Beck (2000) tregohet se Extreme Programming (XP) është krijuar dhe zhvilluar për të adresuar nevojat specifike të zhvillimit të softuerit të kryer nga ekipe të vogla përball kërkesave të paqarta dhe të ndryshueshme. Kjo metodologji sfidon parime të shumta konvencionale, duke përfshirë edhe supozimin e gjatë, duke konstatuar se kostoja e ndryshimit të një pjese të softuerit domosdoshmërisht rritet në mënyrë drastike gjatë kalimit të kohës. Projekti fillon me një dizajn të thjeshtë që vazhdimisht transformohet për të shtuar fleksibilitetin e nevojshëm dhe për të hequr kompleksitetin e panevojshëm. Vendosja e nje sistemi minimal në përdorim të shpejtë dhe duke e rritur atë në çfarëdo drejtimi tregojnë më shumë vlefshmëri. Nuk shkruhet për të ruajtur dokumentacionet pasiqë takimet tek XP bëhen ballë për ballë, ose nëpërmjet testeve efikase dhe kodit të shkruar me kujdes. [5]

Në artikullin "Planned Methodologies vs Agile Methodologies under the Pressure of Dynamic Market" të Kamel, Bediwi dhe Al-Rashoud bëhet krahasimi i Plan Driven dhe Agile metodologjive. Tregohet se Plan Driven metodologjitë janë krijuar për të kontrolluar dhe për të zgjidhur problemet e kodit dhe për të rregulluar stilin e zhvillimit, ku softueri është shkruar pa ndonjë plan dhe dizjani është grumbulluar bashkë nga shumë vendime afatshkurtëra. Plan Driven metodologjitë përpiqen për të zgjidhur këto probleme duke imponuar një proces të disiplinuar

mbi zhvillimin e softuerit, me qëllim për të bërë zhvillimin e softuerit më të parashikueshëm dhe më efikas. Metodologjitë Agile janë krijuar për të qenë super metodologjitë që arrijnë fleksibilitetin e nevojshëm. Këto metodologji ofojnë prototipe të hershme dhe të vazhdueshme të programit dhe mirëpresin ndryshimin e kërkesave madje edhe në fund të zhvillimit. [6]

Në artikullin "Critical Analysys of the Scrum Project Management Methodology" të Ionel (2008) përshkruhet se Scrum është përdorur në kompani të mëdha në fushën e zhvillimit të softuerit, dhe ka një normë të rëndësishme suksesi. Analistët e fushës mendojnë se Scrum mund të jetë e pështatshme edhe për llojet e tjera të kompanive të zhvillimit të softuerit, për të përfituar nga përdorimi i mjeteve dhe teknikave të orientuara në objekte. Është një metodologji e menaxhimit, e zhvilluar për të përmirësuar dhe për të ruajtur një sistem ekzistues ose një prototip të prodhimit. Kjo metodologji supozon ekzistencën e një projekti dhe disa sekuenca të kodit burimor, të cilat pothuajse gjithmonë ekzistojnë në zhvillimin e programeve të orientuara në objekte për shkak të bibliotekave të klasëve. Metodologjija Scrum nga ana tjetër është projektuar të jetë fleksibile gjatë gjithë jetës së projektit. Ajo siguron mekanizmat e kontrollit për të planifikuar një realizim, dhe pastaj për menaxhimin e variablave të projektit se si ai zhvillohet. Kjo lejon kompanitë për të ndryshuar projektin dhe dërgimet e tij në çdo kohë. [7]

Në artikullin "A comparison between Agile and Traditional Software Development Methodologies" të Awad (2005) tregohet për dallimin mes Plan Driven dhe Agile ku thuhet se, dallimi kryesorë mes Plan Driven dhe Agile metodologjive është pranimi i ndryshimeve. Metodologjitë Plan Driven ngrijnë funksionalitetin e produktit dhe nuk lejojnë ndryshimet. Megjithatë procesi Agile është i sukseshëm në ditët e sotme ku si avantazh lejon ndryshimet në çdo fazë të projektit. Kjo e bën të vështirë për të zbatuar një proces parashikuese, ose për të siguruar një sërë kërkesash të qëndrueshme në këtë mjedis të paqëndrueshëm dhe vazhdimisht në ndryshim. Metodologjija Agile përqëndrohet në talent dhe aftësitë e individëve dhe proceseve për njerëz dhe ekipe të veçanta, jo si metodologjija Plan Driven ku të gjitha detyrat dhe rolet janë të caktuara për individët dhe pritet që individët do të kryejnë detyrat e tyre në përputhje me rrethanat. [8]

Në librin "Software Engineering – A PRACTITIONER'S APPROACH" të Pressman (2001) tregohet se modeli Waterfall, ndonjëherë i quajtur cikli klasik jetësor, sygjeron një qasje sistematike dhe sekuenciale të zhvillimit të softuerit që fillon me specifikim të kërkesave të

konsumatorëve dhe përparon përmes planifikimit, modelimit, ndërtimit dhe shtrirjen duke arritur në mbështetjen e vazhdueshme të softuerit të përfunduar. Agile mund të aplikohet për çdo proces softuerik. Megjithatë, për të arritur këtë është e domosdoshme që procesi të jetë projektuar në një mënyrë që lejon ekipi i projektit për të përshtatur detyrat dhe për t'i përmirësuar ato. [9]

3. Deklarimi i problemit

- Si funksionon modeli Waterfall dhe V-Model?
- Cilat janë avantazhet dhe disavantazhet e këtyre dy modeleve?
- Si funksionin modeli Scrum dhe ExtremeProgramming?
- Cilat janë avantazhet dhe disavantazhet e këtyre dy modeleve?
- Cilat janë dallimet mes Plan Driven dhe Agile?
- Krahasimi i definimit të kërkesave dhe dizjanit tek Plan Driven dhe Agile?

4. Analiza

4.1. Plan Driven

Metodologjija Plan Driven është krijuar për të kontrolluar dhe për të zgjidhur problemet e kodimit dhe rregullimit të stilit të zhvillimit, ku të gjithë softuerët janë koduar pa ndonjë plan themelorë dhe dizajnimi i sistemit është mbledhur së bashku nga vendime afatshkurtëra. Sa më shumë që rritet sistemi atherë aq më e vështirë është shtimi i funksioneve të reja dhe rregullimi i gabimeve. Plan Driven përpiqet për ta zgjidhur këtë problem duke imponuar përdorimin e një procesi të disiplinuar për zhvillimin e softuerit me qëllim për ta bërë atë më të parashikueshëm dhe më efikas. Për të qenë më të parashikueshëm kjo metodologji fokusohet në krijimin e një dizajni të plotë më të hershëm, nga i cili janë formuluar planet e detajuara për ndërtimin e softuerit. Metodologjija Plan Driven nuk shërben për projektet që kanë për qëllim të përfundojnë softuerin shpejtë për ta plasuar në treg siq janë ueb aplikacionet. Këto aplikacione kërkojnë një kohë të shkurtër prej ditësh ose javësh për tu plasuar në treg, duke plotësuar dhe ofruar shërbimet e specifikuara nga klienti. Për këtë arsye nevojiten procese të tjera të zhvillimit të sofuerit me më shumë fleksibilitet dhe përfshirje të klientit. Modelet Waterfall dhe V-Model janë dy modelet më të njohura të teknikës së planifikuar. [2]

4.1.1. Waterfall

Modeli i parë i publikuar i procesit të zhvillimit softuerik ka rrjedhur nga proceset e sistemeve inxhierike të përgjithshme (Royce, 1970). Përshkak të kalimit nga njëra fazë në tjetrën, ky model njihet si Waterfall modeli ose cikli jetësor i softuerit. Waterfall modeli është shembull i procesit Plan Driven, ku duhet planifikuar të gjitha aktivitetet e proceseve para se të fillojmë të punojmë në ato. Fazat kryesore të modelit Waterfall reflektojnë direkt në aktivitetet themelore të zhvillimit që janë:

- Analiza dhe definimi i kërkesave
- Dizajni i sistemit dhe softuerit
- Implementimi dhe testimi i njësive

- Integrimi dhe testimi i sistemit
- Operimi dhe mirëmbajtja.

Modeli Waterfall është konsistent me modelet e proceseve tjera inxhinierike dhe dokumentacioni shkruhet në çdo fazë. Kjo e bën procesin të monitorueshëm kështuqë menaxherët mund të vëzhgojnë progresin kundrejt planit zhvillues. Problemi kryesor është ndarja jofleksibile e projektit në faza të veçanta. Detyrat duhet të bëhen në një fazë më të hershme të procesit, gjë që e bën më të vështirë për t'iu përgjigjur ndryshimeve të kërkesave të klientëve. Në parim modeli Waterfall duhet të përdoret vetëm kur kërkesat janë të qarta që në fillim dhe nuk kanë tendencë të ndryshojnë gjatë zhvillimit të sistemit. Gjithsesi ky model reflekton llojin e procesit që përdoret në projekte tjera inxhinierike, edhe pse është më e lehtë të përdoret një model për tërë projektin. [1]

4.1.1.1. Funksionimi i Waterfall metodologjisë

Në parim, rezultatet e çdo faze janë një apo më shumë dokumente të aprovuara. Faza e ardhshme nuk duhet të fillojë pa përfunduar ajo paraprake. Në praktikë, këto faza përputhen dhe shkëmbejnë informacione me njëra-tjetrën. Gjatë dizajnimit identifikohen problemet me kërkesat, gjatë kodimit gjinden problemet e dizajnit e kështu me radhë. Procesi softuerik nuk është një model i thjeshtë linear, por përfshinë komente nga njëra fazë në tjetrën, prandaj dokumentet e shkruara në çdo fazë duhen të modifikohen për të reflektuar ndryshimet e bëra. Përshkak të kostos së prodhimit dhe aprovimit të dokumenteve, ripërsëritjet janë të kushtueshme dhe përfshijnë modifikime të dukshme. Për këtë arsye pas një numri të vogël të ripërsëritjeve është normale stagnimi i disa pjesëve të zhvillimit (siç është specifikimi) dhe vazhdimi në fazat më të vonshme të zhvillimit. Problemet mbesin për zgjidhje të mëvonshme, injorohen apo zgjidhen përmes programimit. Stagnimi i hershëm i kërkesave mund të thotë që sistemi nuk do të përformojë atë çfarë përdoruesi dëshiron si dhe mund të përfundojë në sisteme keq të strukturuara. [1]

4.1.1.2. Avantazhet dhe disavantazhet e Waterfall metodologjisë

Do të numërojmë disa nga avantazhet dhe disavantazhet e modelit Waterfall. Në realitet kemi parë realizim të planeve ku faza e implementimit është ngushtuar, që do të thotë se ekipi zhvillues ka më pak kohë për të dorëzuar një zgjidhje efektive. Kur bëhen shkurtesat bie kualiteti

i produktit dhe testimi i njësisë integruese është faza e parë që ndikohet nga kjo. Grupet testuese në fazën e tyre marrin zgjidhje që përmbajnë shumë defekte gjë që ju vështirëson punën, kështuqë edhe pse ndarja në ekipe shihet si avantazh, shumë lehtë mund të kthehet në disavantazh nëse një ekip vonohet me dorëzimin e pjesës së tyre të projektit.

Modeli Waterfall nuk lejon kohë për rishqyrtimet e dizajnit. Në momentin që kërkesat konfirmohen ato nuk supozohet të ndryshojnë, do të thotë ekipi zhvillues ka një dizjan fiks në të cilin duhet të punojnë. Në realitet kjo nuk ndodh dhe ndryshimet në kërkesa rezultojnë në kaos për shkak që dokumentacioni i dizajnit kërkon ndryshime dhe rikonfirmime nga ana e palëve të interest.

Disavantazhi më i madh i këtij modeli është që nuk mund të jep një prototip funksional deri në fazat e vonshme të procesit. Kjo do të thotë që përdoruesit final nuk mund të shohin vizionin e tyre të jetësuar derisa të jetë shumë vonë për të bërë ndonjë ndryshim. Është e vështirë për njerëzit që nuk i përkasin fushës teknologjike ta kenë të qartë se si duan që aplikacioni i tyre të operoj përderisa nuk kanë ndonjë vizualizim të tij të cilin mund ta komentojnë. Në këtë aspekt ndihmojnë prototipet në fazën e dizajnimit të sistemit, por asgjë nuk e zëvendëson dorëzimin e kodit funksional për tu provuar nga vetë klienti (gjë që mungon tek modeli Waterfall).

Avantazhi i parë është ndarja e dorëzimit final të projektit në faza të ndryshme që lehtëson mbajtjen e kontrollit mbi proceset e zhvillimit si dhe planifikimin e planprogramit që lehtëson punën e menaxherit të projektit. Me ndarjen e projektit në faza përkatëse ju caktohen role të ndryshme departamenteve dhe ju caktohet një interval kohor për të përfunduar detyrat e tyre. Nëse ndonjë prej departamenteve nuk mund të përfundojë detyrat brenda intervalit kohor të caktuar për menaxherin është më e lehtë të editoj planin e zhvillimit.

Waterfall është një proces i thjeshtë për tu kuptuar, dhe në letër duket një ide e mirë për të planifikuar një projekt, gjithashtu është më lehtë e menagjueshme për drejtuesin e projektit. Fazat kompletohen njëra pas tjetrës ku rezultatet nga njëra fazë shërbejnë si të dhëna hyrëse për fazën tjetër. Si proces funksionon më mirë në projektet ku rreziku për ndryshimin e kërkesave është më i ulët. Çdo fazë në Waterfall është qartë e definuar ku kjo e bën më të lehtë ndarjen e roleve në ekipe dhe departamente që janë pjesë e projektit. Pasiqë çdo fazë është mirë e definuar pikësynimi i vendosur nga menaxheri i projektit është shumë i qartë, për shembull nëse jemi

duke punuar në fazën e analizave të kërkesave duhet të jemi të kujdesshëm që të kuptojmë saktësisht qfarë duhet të dorëzojmë në fazën tjetër. [3]

4.1.1.3. Diagrami i rrjedhës së punës së Waterfall metodologjisë

Procesi Waterfall është i ndarë në faza, ku rezultati i një faze shërben si e dhënë në fazën tjetër.

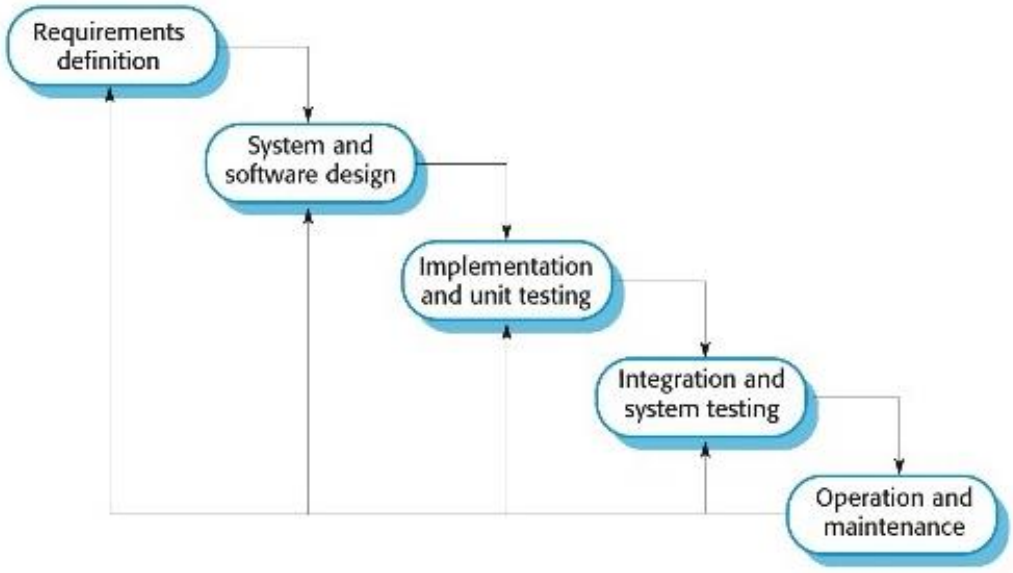

Figura 1 - Workflow Diagrami [1]

Në Figura 1 janë paraqitur hapat e zhvillimit të punës në modelin Waterfall. Hapat janë si vijojnë:

- Hapi 1: Specifikimi i kërkesave, të gjitha kërkesat e mundshme për zhvillimin e sistemit gjinden dhe dokumentohen në dokumentin e specifikimit të kërkesave. Ky dokument zakonisht kërkon aprovimin nga palët kryesore të interesit. Kjo pjesë e modelit zakonisht organizohet nga analistat e biznesit, por duke u bazuar në madhësinë e projektit apo ekipit, mund të përfshihen edhe pjesëtarë tjerë të ekipit zhvillues. Kjo fazë ka të bëjë me nxjerrjen e tërë kërkesave të sistemit nga palët e interesit, ku përfshihen funksionaliteti i kërkuar, dokumentimi i rregullave dhe proceseve të biznesit apo kërkesave tjera që mund të ndikojnë në sistemin e përgjithshëm.
- Hapi 2: Dizajni i sistemit, specifikimet e kërkesave nga faza e parë inspektohen dhe dizajni i sistemit është kompletuar. Ky dizajn ndihmon në specifikimin e kërkesave të dizajnimit dhe gjithashtu në dizajnimin e arkitekturës së përgjithshme të sistemit. Është

faza ku arkitektët, dizajnerët dhe zhvilluesit punojnë së bashku për të vendosur se si do të strukturohet i tërë sistemi. Përfshinë anën e kodimit, zgjedhjen e teknologjisë dhe perspektivën e infrastrukturës.

- Hapi 3: Implementimi, kjo është faza ku zhvilluesit marrin dizajnin dhe fillojnë të kodojnë për ta kthyer këtë dizajnin në realitet. Zhvilluesit gjithashtu shkruajnë njësi automatike dhe teste integruese.
- Hapi 4: Integrimi dhe testimi, këtu të gjitha rezultatet nga faza e implementimit bashkohen dhe testohen si tërësi. Ekipi testues punon sipas një plani testues të definuar. Kur sistemi testohet dhe konfirmohet nga ekipi testues, faza tjetër është dorëzimi i zgjidhjes tek përdoruesit final.
- Hapi 5: Operimi dhe mirëmbajtja, çdo problem që është raportuar prej fazës së implementimit duhet të rregullohet dhe ridorëzohet në fazën e mirëmbajtjes. Gjithashtu mund të bëhen rregullime të vogla në sistem. Nëse përmirësimet rriten në numër atëherë mund të rifillohet procesi Waterfall dhe fillon marrja e kërkesave shtesë. [3]

Kur të gjitha këto faza përfundohen, progresi rrjedh hap pas hapi siq parashikohet në Waterfall. E rëndësishme për këtë model është që fazat nuk tejkalohen asnjëherë. [4]

4.1.2. V-Model

V-Model është një version i modifikuar i Waterfall. Për dallim nga metoda Waterfall, V-Modeli nuk ishte dizajnuar për të funksionuar në një mënyrë lineare. Për këtë, procesi kthehet përpjetë pas fazës së implementimit ose kodimit, duke e formuar kështu shkronjën V, prandaj edhe e ka marrë emrin V-Model. Bazohet në iden e të pasurit një fazë të testimit për secilën fazë të zhvillimit. Kjo do të thotë që për secilën detyrë në ciklin e zhvillimit, ekziston një fazë e testimit. Ky është një model strikt, dhe faza tjetër fillon vetëm pas përfundimit të fazës së mëparshme. [3]

4.1.2.1. Funksionimi i V-modelit

Për t'a bërë V-Model sa më të suksesshëm, kërkesat e projektit duhet të jenë të definuara mirë, të dokumentuara, dhe të menduara mirë në mënyrë që ato të mos ndryshojnë me kohën. Poashtu duhet të jeni të sigurtë që definimi i një produkti të jetë stabil. Me V-Model, testimi relevant i fazës së zhvillimit planifikohet paralelisht, në mënyrë që të ketë faza verifikuese në njërën anë të V-së dhe faza të aprovimit në anën tjetër. Faza e kodimit i bashkon të dy anët e V-modelit.

4.1.2.2. Avantazhet dhe disavantazhet e V-modelit

V-Model ka përparësi dhe mangësi të ngjajshme me modelin tradicional Waterfall. Sikurse me modelin Waterfall, V-Model nuk është fleksibil ndaj ndryshimeve në kërkesa. Kjo do të thotë që duhet të përsëriten faza të ndryshme në model për t'u siguruar që të gjitha dokumentimet janë në vendin e duhur. Ndryshimet në kërkesa mund të shkaktojnë përçarje në projekt, kështu që ju duhet të siguroheni që kërkesat janë në rregull që nga fillimi.

Nëse jeni duke punuar në një projekt ku ndryshimet mund të parashikohen, atëherë V-Model nuk është modeli i duhur për ju. Pasi që të filloni të testoni zgjidhjet e krijuara, do të jetë shumë e vështirë dhe shumë e shtrenjtë për t'u kthyer mbrapa për të bërë ndryshime në kode. Disavantazhi më i madh i këtij modeli është që ju nuk do të keni një softuer që punon deri në fazat më të vonshme të procesit. Kjo do të thotë që përdoruesit tuaj nuk do të mund t'a shohin vizionin e tyrë të bëhet realitet deri në fazat e fundit, kur është shumë vonë për ndryshime.

V-Model mund të prezentoj një nivel të lartë të rrezikut dhe të paparashikueshmërisë për çdo gjë përveq projekteve të vogla. Vetëm pse një set i kërkesave për një dizajn është aprovuar nuk do të thotë që kërkesat nuk do të ndryshojnë. Fokusohet më së shumti që informatat për kërkesa, dizajni, dhe implementimi të jenë të qarta dhe të sakta që nga fillimi. Siq e cekëm më herët, ky është një ideal shumë i mirë, po fatkeqësisht në realitet nuk ndodh kështu dhe ky paraqet një rrezik shumë të madh për një projekt. U përmend më lartë se si V-Model është më i përshtatshëm për projekte të vogla, por ka projekte që janë të vogla por shumë të komplikuara.Sa më shumë kompleksitet aq më e lartë është mundësia që ndryshimet do të jenë të nevojshme. Kompleksiteti në një projekt është shumë i vështirë të testohet dhe të aplikohet, dhe zakonisht mund të shkaktojë vonesa në fazat më të vonshme të ciklit për zhvillimin e produktit me V-Model.

Përparësia e parë, është se V-Model është relativisht i lehtë për t'u kuptuar dhe aplikuar. Përshtatet mirë me kompanitë që kanë departamente të ndryshme që afektojnë procesin e zhvillimit. V-Model poashtu është shumë i lehtë për t'u menaxhuar, pasi që mund të kalohet në fazën tjetër vetëm atëherë kur faza paraprake është kompletuar.

Ky model funskionon më së miri në projekte të vogla ku rreziku i ndryshimit të kërkesave është më i vogël se sa me projekte të mëdha. Në përgjithësi, V-Model është i lehtë për t'u kuptuar dhe fazat e miratimit mund të jenë të mira për departamentet e testimeve më të avancuara.

Menaxherët e projekteve në të shumtën e rasteve preferojnë V-Model, pasi që është më i lehtë të menaxhohet. Intenziteti i modelit përshkruan shumë mirë pikëpamjen e menaxherëve të projektit dhe e bën punën e tyre më të lehtë. [3]

4.1.2.3. Diagrami i rrjedhës së punës së V-Model

Në Figura 2 janë paraqitur hapat e zhvillimit të punës në V-Model, ata janë:

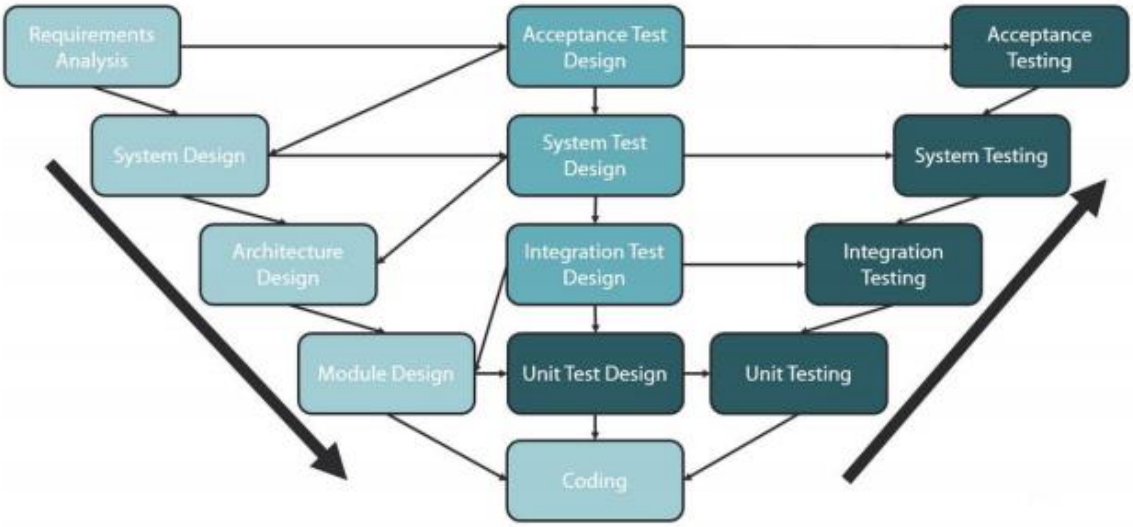

Figura 2- V-Model Diagrami [3]

- Hapi 1: Faza e analizimit të kërkesave është faza e parë në ciklin e zhvillimit, gjatë të cilës kuptohen nevojat e klientit, gjë që kërkon bashkëpunim me klientin që të kuptohen pritjet e tij ose të saj. Pasi që shumica e klientëve nuk janë të sigurtë se çfarë dëshirojnë, planifikimi i testimit për pranimin e dizajnit ndodh gjatë kësaj faze, dhe kërkesat e biznesit mund të përdoren si informacion për testin e pranimit.
- Hapi 2: Hapi tjetër është dizajni i sistemit. Pasi të kuptoni kërkesat, ju krijoni një dizajn më komplet të sistemit në tërësi. Ky dizajn përfshin dizajnimin e softuerit dhe të harduerit. Në të njejtën kohë që përgatitet dizajni i sistemit, duhet të krijohet edhe dizajni për testimin e sistemit në mënyrë që ekipet e testimit të mund të pri-planifikojnë aktivitetet e tyre të testimit.

- Hapi 3: Pastaj vie faza e dizajnit arkitektural. Ky dizajn fokusohet më shumë në dizajn se sa dizajni i sistemit. Kjo mund të rezultojë në shumë dizajne të tjera të propozuara të cilat balancojnë shitësit, shpenzimet, dhe faktorë të tjerë.
- Hapi 4: Testimi i njësive është një pjesë shumë e rëndësishmë e procesit të zhvillimit, dhe ndihmon në identifikimin e hershëm të problemeve në kodime, në mënyrë që ekipi ka një pamje të qartë të të gjitha thyerjeve të shkaktuara nga varësit tjera. Pasi që të kompletohen të gjitha këto faza të dizajnit, ju mund të vazhdoni me fazën e kodimit. Të gjithë duhet të jenë dakord me gjuhën e përdorur në fazën e kodimit dhe me arkitekturën, në mënyrë që krijuesit të fillojnë menjëherë me procesin.
- Hapi 5: Pastaj, hyjmë në fazën e miratimit të V-Model. Së pari, kemi Testimin e Njësive. Testet e njësive të dizajnuara në fazën e dizajnit të modulit krahasohen me kodin në fazën e miratimit. Testimi i njësive ndodh në nivel të kodimit dhe ndihmon në eliminimin e problemeve në faza të herëshme. Edhe pse është e pamunduar të gjinden të gjitha defektet përmes testimit të njëisive, këto testime japin indikacione të mira për të gjitha thyerjet që mund të ndodhin ose jo. Kjo do të thotë që krijuesit duhet të jenë shumë të diciplinuar që të shkruajnë testime të mira të njësive të cilat shtojnë vlera, dhe nuk testojnë vetëm karakteristikat e gjuhës.
- Hapi 6: Pastaj kemi fazën e Testimit të Integrimit. Testimet e integrimit përformohen që të testojnë bashkë-ekzistencën e moduleve dhe komponentëve të ndryshëm përbrenda sistemit. Këtu praktikisht sigurohemi që të gjitha integrimet në mes të komponentëve të ndryshëme përbrenda sistemit punojnë siq duhet.
- Hapi 7: Testimi i Sistemit. Testet e sistemit testojnë funksionalitetin e sistemit në tërësi dhe komunikimin në mes të gjithë sistemeve të jashtme. Për shembull, nëse kemi integruar me ofruesit e pagave të palës së tretë, ato do të testohen në këtë fazë. Shumica e problemeve me përshtatshmërinë e softuerit dhe harduerit që mund t'i ballafaqojmë, mund të gjinden gjatë kësaj faze.
- Hapi 8: Testimi i Pranueshmërisë. Testimi i pranueshmërisë është i asociuar me fazën e analizimit të kërkesave të biznesit, dhe përfshin testimin e produktit në mjedisin e përdoruesit. Testet e pranueshmërisë vënë në pah problemet e përputhshmërisë me sistemet tjera që mund të ndodhen në ambientin e përdoruesit. Testimi i pranueshmërisë

gjithashtu mund të zbuloj probleme të jo-funksionalitetit siq është defekte në performancë në mjedisin e përdoruesit.

4.2. Agile

Metodologjitë Agile janë krijuar për të qenë super metodologjitë që arrijnë fleksibilitetin e nevojshëm. Këto metodologji kanë për qëllim të përmbushin konsumatorin me anë të ofrimit të hershëm dhe të vazhdueshëm të pjesëve të programit funksional me ndryshime të mëdha të kërkesave, madje edhe në fund të zhvillimit. Proceset Agile shfrytëzojnë ndryshimet si avantazh tek klientët konkurues, dhe dorëzimin e shpeshtë të softuerit më një preferencë të shkallës së lartë në kohë të shkurtër. Njerëzit e biznesit dhe zhvilluesit e softuerit duhet të punojnë së bashku çdo ditë gjatë gjithë projektit. [2]

4.2.1. Extreme Programming

Extreme Programming (XP) është ndoshta opsioni më i mirë dhe më i përdoruri i metodave Agile. Emri ishte shpikur nga Beck (2000) sepse kjo mënyrë ishte zhvilluar duke shtyer praktikat e zakonshme, siq është zhvillimi përsëritës, në nivele “ekstreme” [1]. XP është një metodologji për zhvillimin e softuereve e cila ka për qëllim të përmirësojë kualitetin dhe nivelet e përgjigjjes së softuereve në bazë të kërkesave të kilentëve. Si një lloj Agile për zhvillimin e softuereve, përkrahë shkarkime të shpeshta dhe cikle më të shkurtëra të krijimit, të cilat kanë për qëllim të përmirësojnë produktivitetin dhe të njoftojnë pika të kontrollit përmes të cilave kërkesat e reja të klientëve mund të adoptohen. [3]

4.2.1.1. Funksionimi i Extreme Programming

Ekipet e XP performojnë gati secilin aktivitet të zhvillimit të softuereve në të njejtën kohë. Analizimi, dizajni, kodimi, testimi, madje edhe zhvillimi ndodhin me frekuenca të shpejta. Kjo është shumë për t’u bërë në të njejtën kohë. XP e bën këtë duke punuar me përsëritje, rritje javore të punës. Çdo javë, ekipi punon pak nga palinifikimi, dizajni, kodimi, testimit, e të tjera. Ata punojnë në storie, karakteristika shumë të vogla, ose pjesë të karakteristikave, të cilat kanë vlerë konsumatore. Gjatë javës, ekipi punon në të gjitha fazat e krijimit të secilës storie. Në fund

të javës, ata e shpërndajnë softuerin e tyre për rishikim. (Në disa raste, ata e shpërndajnë softuerin tek klientët aktual). [5]

4.2.1.2. Avantazhet dhe disavantzhet e Extreme Programming

Për shkak se konsumatori është anëtarë aktiv i ekipit në metodologjinë XP ndryshimet në kërkesa kërkohen në mënyrë joformale. Si pasojë e kësaj, qëllimi i projektit mund të ndryshojë dhe puna e herëshme mund të ndryshohet për të përshtatur nevojat aktuale.

Shumë projekte kanë klientë të shumtë, dhe secili prej tyre kanë kërkesat e veta. Në XP, vetë ekipi është i ngarkuar me asimilimin e kërkesave të konsumatorëve të ndryshëm, një punë që mund të jetë jashtë fushës së tyre të autoritetit.

User storiet dhe testet e pranimit janë vetëm manifestime të qarta të kërkesave në XP. Kritikët argumentojnë se një model më formal ose specifikimi më i detajuar është shpesh i nevojshëm për të siguruar që mangësitë, paqëndrushmëritë dhe gabimet të zbulohen para se sistemi të ndërtohet.

XP minimizon nevojën për dizajnin arkitukturorë dhe në shumë raste, tregon se të gjitha llojet e dizajnit duhet të jenë joformale. Kritikët argumentojnë se kur sistemet komplekse ndërtohen, dizajni duhet të theksohet për tu siguruar që struktura e përgjithshme e programeve do të shfaq cilësi dhe besushmëri. [6]

XP është përshkruese në fushë. Aplikohet mirë për ekipet e vogla që janë nën 10 zhvillues, dhe klienti duhet të jetë pjesë përbërse e ekipit ose lehtësisht i qasshëm.

Zhvillimi ndodh me ndërtime të shpeshta ose me ripërsëritje, ku secila prej të cilave është e lëshuar dhe jep funksionalitet në rritje. Kërkesat janë të specifikuara si user storie, ku secila jep një copë funksionalitet që përdoruesi e kërkon. Programerët punojnë në çifte, duke ndjekur standardet strikte të kodimit dhe bëjnë vetë testimin e njësive të tyre. Kërkesat, arkitektura dhe dizjani dalin gjatë rrjedhjes së projektit. [7]

4.2.1.3. Diagrami i rrjedhës së punës së Extreme Programming

Këtu në Figura 3 i kemi disa nga fazat e modelit XP që janë shpjeguar në mënyrë më të detajuar.

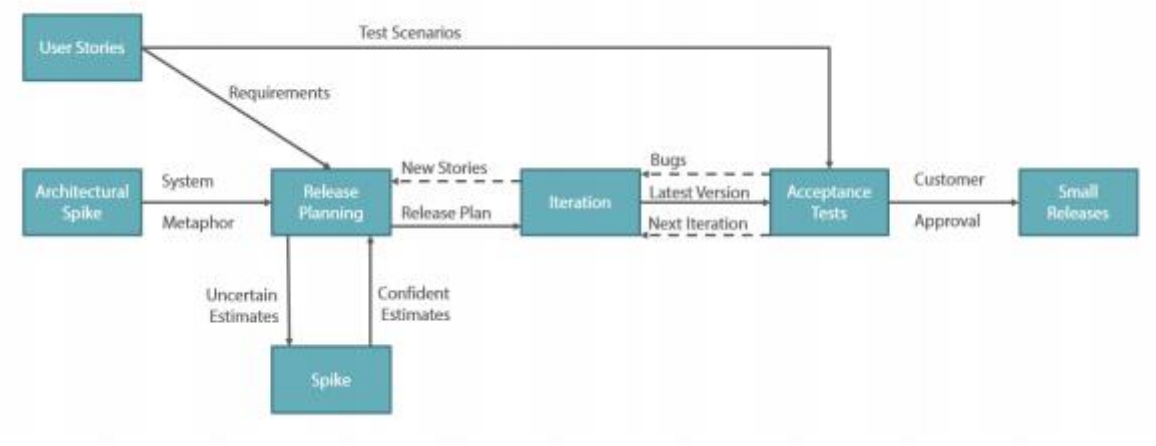

Figura 3 - Extreme Programming Diagrami [3]

- Hapi 1: Prej fazës "User Stories" ne përfundojmë me një listë të nevojave dhe një seri të testeve të mundshme të cilat formojnë testet e pranueshmërisë.
- Hapi 2: Prej "Architectural Spikes" ne përfundojmë me një metaforë të sistemit, e cila është një storie të cilën të gjithë klientët, programerët, dhe menaxherët mund t'a përdorin për të treguar se si punon sistemi.
- Hapi 3: Gjatë fazës së "Release Planning" ne përcaktojmë cilat kërkesa janë të përfshira në cilat shkarkime afat-shkurta, dhe kur duhet të dorëzohen. Klientët dhe zhvilluesit janë pjesë e këtij procesi.
- Hapi 4: Plani i shkarkuar pastaj pasohet tek "Development Iteration" ku zhvillohen teste për njësi dhe kode. Secila storie e re që vjen në pah gjatë kësaj faze pasohet mbrapa në "Release Planning". Nga kjo fazë, zakonisht krijohet një softuer që punon. Ky softuer duhet të kalojë testet e pranueshmërisë të cilat janë krijuar nga "user stories." Nëse ka ndonjë të metë, atëherë ato rregullohen nga zhvilluesit.

Zakonisht ndodhin përsëritje të shumta për një projekt. Pasi që testet e pranueshmërisë të kalojnë nga përsëritjet dhe klientët të aprovojnë sistemin që është krijuar në përsëritje, ndodh një shkarkim i vogël, i cili i'u jep përdoruesve qasje tek kodi punues ku ata mund të fillojnë t'i mbledhin benefitet më herët. [3]

4.2.2. Scrum

Scrum është një kornizë e shpejtë përsëritëse dhe rritje të zhvillimit agile, që përdoret për mehaxhimin e zhvillimit të produktit. Kjo kornizë definon një strategji për zhvillimin e produktit e cila është fleksibile dhe holistike, në të cilën ekipa e zhvillimit punon si një tërësi e vetme për të arritur një cak të dëshiruar. Scrum është një mënyrë e shpejtë për të menaxhuar një projekt, zakonisht zhvillimin e softuereve. Në botën e Scrum, në vend se të përshkruhet se si do të kryhet secila punë brenda projektit, shumica e përshkrimeve vendosen nga ana e ekipit zhvillues. Kjo ndodh sepse ekipa do t'a dijë më mirë se si t'a zgjidhin një problem me të cilin ata konfrontohen. Scrum u krijua nga Ken Schëaber dhe Jeff Sutherland në vitet 1990-ta, të cilët publikuan një ese për të përshkruar gjithë procesin. Termi "scrum" është huazuar nga sporti i rugbit për të përshkruar rëndësinë e ekipeve, dhe ilustron disa analogji në mes të sportit të rugbit dhe të qenit të suksesshëm në zhvillimin e një produkti të ri.

4.2.2.1. Funksionimi i Scrum

Është një kornizë për menaxhimin e projekteve e cila është e aplikueshme në secilin projekt me afate strikte, kërkesa të komplikuara, dhe një shkallë të veqantisë. Në Scrum, projektet përparojnë përmes disa përsëritjeve të cilat njihen me emrin "sprints". Secili sprint zgjat përafërsisht dy deri në katër javë. Nëse mundohemi t'a përshkruajmë kornizën Scrum është shumë e lehtë t'a ndajmë në tre fusha kryesore, të cilat janë:

- Rolet, të cilat përfshijnë Product Owner, Scrum Master, dhe Scrum Team
- Ceremonitë, të cilat përfshijnë sprint planning meeting, sprint review, dhe sprint retrospective meetings.
- Artifaktet e srcum, të cilat përfshijnë product backlog, sprint backlog dhe grafiku burn down.

Normalisht, product owner i ekipit të Scrum është aktori kryesor i projektit, por ky duhet të jetë edhe një analizues i biznesit që punon afër me biznesin dhe përdoruesit e sistemit. Pjesë e përgjegjësisë së tij është të ketë një vizion për atë që ai ose ajo dëshirojnë të ndërtojnë, dhe t'a përcjellë atë vizion tek ekipa e Scrum. [3]

4.2.2.2. Avantazhet dhe disavantazhet e Scrum

Një dobësi e Scrum është fakti se, kur projekti është për një klient të jashtëm, ky duhet të jetë i përfshirë shumë në këtë projekt. Klienti duhet të jetë në gjendje dhe në dispozicion për të testuar realizimet apo dërgimet mujore (ose periodike), dhe për të sygjeruar funksionalitete të reja ose të modifikuara.

Një tjetër dobësi e mundshme tek Scrum është nga periudha relativisht e gjatë në të cilën klienti (i brendshëm apo i jashtëm) nuk mund të ndërhyjë në projekt. Edhe pse në parim kjo është një avantazh, ka situatë në zhvillimin e softuerit, kur ndërhyrja e klientit duhet të bëhet brenda një sprinti. Nëqoftë se metodologjija nuk mund të akomodoj këto ndërhyrje atëherë rreziku mbi projektin është serioz.

Madhësia e vogël e ekipit të projektit mund të konsiderohet si dobësi. Mënyra në të cilën kjo metodologji qaset në projekte të mëdha me ekipe të mëdha është e mundshme, por jo shumë e lehtë për t'u implementuar.

Shikushmëri relativish të ulët mbi projektin jashtë sprintave, me fjalë të tjera është shumë e vështirë të vlerësohet se sa kohë do të marrë ose sa do të kushtojë një projekt me klient të jashtëm, ku ankandi përodoret për të përcaktuar një kontraktor për atë projekt. Kjo mund të jetë një pengesë e madhe gjatë përdorimit të Scrum. [8]

Metodologjija Scrum është projektuar të jetë fleksibile gjatë gjithë jetës së projektit. Siguron mekanizmat e kontrollit për të planifikuar një version, dhe pastaj për menaxhimin e variablave të projektit siç përparon. Kjo lejon organizatat për të ndryshuar projektin në çdo kohë, pra duke dhënë versionin më të përshtatshëm.

Një tjetër avantazh i përdorimit të Scrum është fakti se sprinti një herë vendoset dhe fillon, nuk ka më shumë ndryshime nga karakteristikat të cilat zhvillohen gjatë këtij sprinti. Ekipi mund të ketë një takim para se të filloj sprinti, dhe aty vendoset se cilat janë aktivitetet që do të realizohen për një periudhë 30 ditore (ose për gjatësinë e sprintit të ardhshëm). Duke mbajtur këtë takim, produktiviteti i ekipit fitohet, dhe ri-puna për shkak të ndryshimeve praktikisht zhduket.

Teknologjitë e orientuara në objekte sigurojnë bazën për metodologjinë Scrum. Objektet ose karakteristikat e produktit, ofrojnë një mjedis relativisht diskret dhe të lehtë për tu menaxhuar.

Kodi procedural nuk është zakonisht i pajisur për Scrum, për shkak të ndërfaqeve të shumta dhe komplekse. Scrum mund të aplikohet në mënyrë selektive për sistemet procedural, vetëm kur ndrëfaqet mes komponentëve të ndryshëm të softuerit janë relativisht të thjeshta, dhe produkti ka një orientim të fort të të dhënave. [8]

4.2.2.3. Diagrami i rrjedhës së punës së Scrum

Në Figura 4 janë paraqitur hapat e zhvillimit të punës në modelin Scrum:

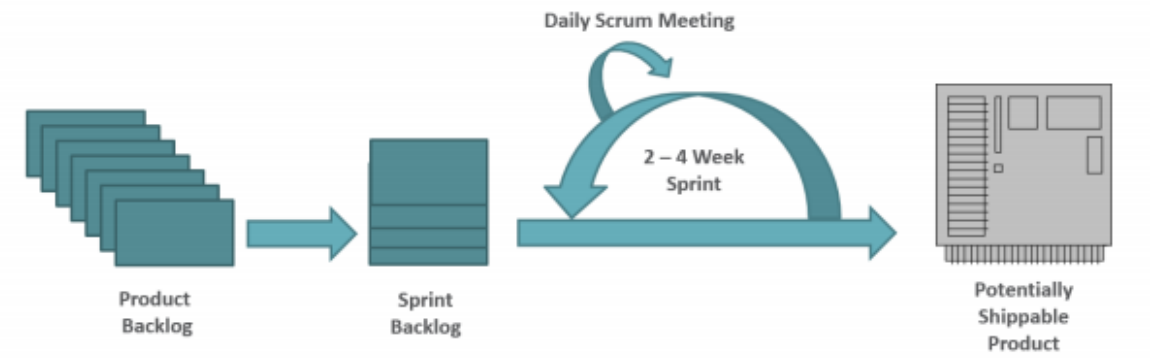

Figura 4 - Scrum Diagrami [3]

- Hapi 1: Product Backlog të projektit janë zakonisht pjesë e një liste me prioritete e cila përmban të gjitha tiparet ose ndryshimet e dëshiruara për projektin. Kur mbahet takimi për planifikimin e sprintave, pikat prej punëve të papërfunduara të projektit zgjidhen dhe identifikohen për t'u implementuar në sprinte të ardhshme dhe vendosen në punët e papërfunduara të sprinteve.
- Hapi 2: Pasi të identifikohen product backlogs të sprinteve, ekipi regjistron një sprint që zgjatë dy deri në katër javë, gjatë të cilit ata i implementojnë pikat prej punëve të papërfunduara të sprinteve.
- Hapi 3: Gjatë secilës ditë të sprintit, ekipi organizon një takim të shkurtër, i cili quhet Daily Scrum. Ky takim ndihmon në përcaktimin e natyrës së secilës ditë dhe ndihmon që të gjithë të jenë në drejtimin e duhur. Të gjithë pjesëtarët e ekipit janë të obliguar të marrin pjesë në këtë takim.

- Hapi 4: Në fund të sprintit, ekipi duhet të ketë një produkt potencial i cili mund të dërgohet që të mund të hyjë në prodhim, dhe që të mund të jep vlerë për klientin. Tani që kemi parë një përmbledhje të Scrum, mund t'a analizojmë secilin rol, artifakt, dhe ceremoni në më shumë detaje. [3]

4.2.2.4. Përmbledhje e Scrum

Product owner: Pronari i produktit është aktori kryesor i projektit i cili përfaqëson përdoruesit, klientët dhe të tjerët në këtë proces. Pronari i produktit është shpeshherë dikush nga ekipa e menaxhimit ose marketingut të produktit, palë e interesit, ose një përdorues kyç.

ScrumMaster: Eksperti i Scrum është përgjegjës për t'a bërë ekipin sa më produktiv të mundshëm. Eksperti i Scrum e bënë këtë të mundur duke e ndihmuar ekipin me përdorimin e procesit Scrum, përmes heqjes së pengesave për progres, mbrojtjes së ekipit nga efektet e jashtme, e kështu me radhë.

Scrum team: Një ekip tipik i Scrum ka përafërsisht pesë deri në nëntë persona, por projektet Scrum mund të përfshijnë, lehtësisht, me qindra persona. Sidoqoftë, Scrum mund të përdoret me shumë lehtësi nga ekipe të përbëra nga një person dhe në shumicën e rasteve ashtu ndodh. Ky lloj ekipi nuk i përfshin rolet tradicionale gjatë zhvillimit të softuereve si për shembull programer, dizajner, tester, ose arkitekt. Secili person i involvuar në projekt punon së bashku për të kompletuar punën të cilën ata së bashku janë zotuar për t'a kompletuar gjatë një sprinti. Ekipet e Scrum e krijojnë një atmosferë shoqërore dhe ndjenjën e "të qenit së bashku."

Sprint planning meeting: Në fillim të secilit sprint, mbahet një takim për planifikimin e sprintit, gjatë të cilit pronari i produktit prezenton tek ekipi pikat kryesore në punët e papërfunduara të projektit. Ekipi i Scrum zgjedh punët të cilat ata mund t'i përfundojnë gjatë sprintit të ardhshëm. Ajo punë zhvendoset prej punëve të papërfunduara të projektit tek punët e papërfunduara të sprintit, që është një listë e të gjitha detyrave që duhen për të kompletuar punët e papërfunduara të projektit të cilat ekipi është zotuar për t'i kompletuar gjatë sprintit.

Sprint review meeting: Në fund të secilit sprint, ekipi demonstron funksionalitetin e përfunduar gjatë takimit për rishikim të sprintit gjatë të cilit ekipi tregon se qfarë kanë arritur për të punuar. Zakonisht, kjo merr formën e demonstrimit të karakteristikave të reja, por në mënyrë joformale

për shembull, prezetimet PowerPoint nuk janë të lejuara. Takimi nuk duhet të merr natyrën e detyrës ose të jetë shpërqëndrim nga procesi.

Sprint retrospective: Gjithashtu në fund të secilit sprint, ekipi udhëheqë një retrospektivë të Scrum, që është një takim gjatë të cilit ekipi (përfshirë ekspertin e Scrum dhe pronarin e produktit) reflekton se sa mirë punon Scrum për ta dhe çfarë ndryshime mund të bëjnë që të bëhet edhe më i mirë.

Daily Scrum: Çdo ditë, gjatë sprintit, mbahet një takim i shkurtër që njihet me emrin Scrum Ditor. Ky takim ndihmon në përcaktimin e natyrës të secilës ditë dhe ndihmon ekipin të qëndrojë në drejtimin e duhur. Të gjithë anëtarët e ekipit janë të obliguar të marrin pjesë në këtë takim.

Product backlog: Punët e papërfunduara të produktit janë zakonisht pjesë e një liste me prioritet e cila përmban të gjitha tiparet ose ndryshimet e dëshiruara për projektin. VINI RE: termi "punë të papërfunduara" mund të jetë paksa konfuz sepse përdoret për dy gjëra të ndryshme. Për sqarim, punët e papërfunduara është një listë e karakteristikave të dëshiruara për produktin. Punët e papërfunduara të sprintit është një listë e detyrave që duhet perfunduar gjatë një sprinti. [9]

The sprint backlog: Është një listë e detyrave të identifikuara nga eksperti i Scrum të cilat duhet përfunduar gjatë një sprinti. Gjatë takimit për planifikimin e sprintit, ekipi zgjedh një numër të punëve të papërfunduara të produktit, zakonisht në formën e storieve të përdoruesit, dhe identifikon detyrat e nevojshme për të kompletuar secilën storie. Shumica e ekipeve llogarisin sa orë secila detyrë mund të marrë për t'u kompletuar nga një anëtarë i ekipit. Është e rëndësishme që ekipi të zgjedh pikat dhe madhësinë e punëve të papërfunduara të sprintit. Pasiqë ka njerëz tjerë që zotohen për të kompletuar detyrat, ata duhet të jenë njerëzit të cilët zgjedhin se për çfarë po zotohen për të përfunduar.

The burn down chart: Është pjesa esenciale e cilido projekt të shpejtë, dhe është një mënyrë për ekipin të shoh qartësisht se qfarë ndodh dhe sa progres është duke ndodhur gjatë secilit sprint Figura 5. Një problem që mund të vërehet në këtë tabelë është nëse vija e punës aktuale është mbi ose ndër vijën ideale të punës, dhe kjo varet në faktin se sa të sakta kanë qenë vlerësimet preliminare. Kjo do të thotë që nëse ekipi juaj vazhdimisht mbivlerëson kërkesat e kohës,

progresi gjithmonë do të shfaqet përpara afatit të paraparë. Nëse ekipi juaj vazhdimisht nënvlerëson kërkesat e kohës, progresi do të shfaqet mbrapa afatit të paraparë. [3]

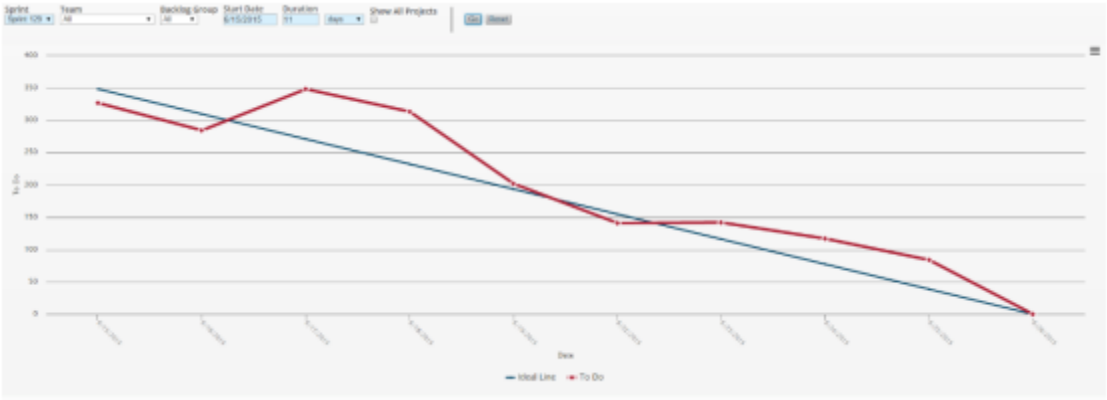

Figura 5 - Grafiku Burn Down [3]

4.3. Krahasimi mes Plan Driven dhe Agile

Qasjet tradicionale të zhvillimit kanë qenë rrotull për një kohë shumë të gjatë. Nga momenti kur është krijuar Waterfall modeli (Royce 1970) është përdorur gjerësisht në dy llojet e softuerëve të vogla edhe të mëdha si dhe është raportuar të jetë model i suksesshëm në shumë projekte. Pavarësisht suksesit që ka gjithashtu ka edhe shumë mangësi siq janë: lineariteti, fleksibiliteti në ndryshimin e kërkesave dhe proceset e larta fromale pavarësisht nga mashësia e projektit. Kent Beck mori në konsideratë këto mangësi dhe prezentoi Extreme Programming, metodologjija e parë Agile e krijuar. Metodologjija Agile merret me kërkesat e paqëndrueshme duke përdorur një numër të teknikave, duke u fokusuar në bashkëpunimin ndërmjet zhvilluesve dhe konstumatorit si dhe mbështet ofrimin e produktit në fillim. Një përmbledhje mes dallimit të Agile dhe Plan Driven metodologjive janë paraqitur në Tabel-ën 1 më poshtë.

Metodologjitë Agile dhe Plan Driven që të dyja kanë pikat e forta dhe të dobëta të tyre. Njerëzit zakonisht përdorin një nga këto metodologji ose ndjekin metodologjinë e tyre përshtatëse. Ka faktorë të ndryshëm që ndikojnë në vendimin e metedologjisë dhe përzgjedhjen se cila është më e përshtatshme për kushte të ndryshme. Këta faktorë mund të kategorizohen në madhësinë e projektit, njerëzit dhe rrezikun.

	Metoda Agile	Metoda Plan Driven
Teknika	Adaptuese	Parashikuese
Matja Suksesit	Vlera e biznesit	Skema të planifikuara
Madhësia e projektit	E vogël	E madhe
Stili menaxhimit	Decentralizuar	Autokratik
Prespektiva për ndryshim	Adaptimi me ndryshime	Qëndrushmëria e ndryshimeve
Kultura	Lidership-Bashkëpunim	Komand-Kontrol
Dokumentacioni	Pak i përdorur	Shumë i përdorur
Komunikimi	Orientuar në njerëz	Orientuar në procese
Ciklet	Të shumta	Te limituara
Fusha	Paparashikueshme/Kërkimore	Parashikueshme
Planifikimi me fillim	Minimal	Gjithëpërfshirës
Kthimi në investime	Në fillim të projektit	Në fund të projektit
Madhësia e ekipit	I vogël/Kreative	I madh

Table 1 - Dallimi në mes metodologjive "Agile" dhe "Plan Driven" [10]

- Madhësia e projektit- Një nga kufizimet e metodologjisë është madhësia e projektit. Elementet kryesore të madhësisë së projektit janë buxheti i projektit, kohëzgjatja dhe ekipi organizues i projektit. Sa më i madh të jetë ekipi dhe buxheti që nevoitet aq më i madh është projekti, kështu hartimi i shumë kërkesave kërkon shumë njerëz dhe më shumë koordinim. Plan Driven mbështet këtë duke ofruar planet, dokumentacionin dhe proceset për komunikim dhe koordinim më të mirë në grupe të mëdha. Siç shihet në Figura 6 më poshtë, Alistair Cockburn një nga themeluesit e bashkësisë Agile, deklaron se për një madhësi të caktuar të problemit, më pak njerëz janë të nevojshëm nëse është përdorur një metodologji më e lehtë dhe më shumë njerëz janë të nevojshëm nëse është përdorur një metodologji më e rëndë. Gjithashtu pohon se: “Ka një limit për madhësinë e problemit që mund të zgjidhet me një numër të caktuar të njerëzve”.

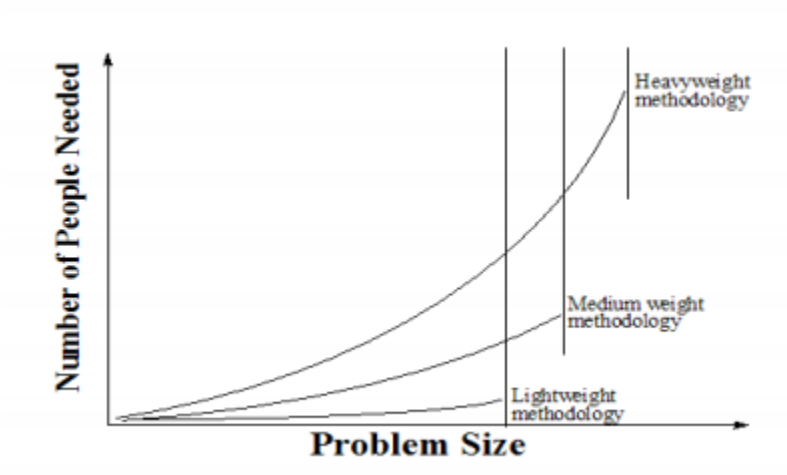

Figura 6 – Ndikimi i madhësis së projektit dhe metodologjisë në staf [10]

Sa më i madh ekipi gjithashtu ndikon në komunikimin në një projekt dhe efektivitetin për person. Në Figura 7 më poshtë tregohet ngarkesa e komunikimit ku me rritjen e numrit të njerëzve shkaktohet rënia e efektivitetit për person. Cockburn thotë se metodologjija është një çështje e koordinimit të njerëzve dhe menaxhimin e komunikimit, pra niveli i metodologjisë duhet të rritet sikurse rritja e numrit të njerëzve. Kjo e bën më të vështirë për të përdorur metodologjinë Agile me ekipe me më shumë se 40 persona duke e bërë metodologjinë Plan Driven alternativë të preferuar për ekipe të mëdha. Megjithatë, Ken Schwaber një zhvillues i Scrum, nuk është dakord me këtë duke deklaruar: "ekipet e mëdha mund të zbërthehen në ekipe të vogla duke përdorur scrum të scrum. Kohëzgjatja e projektit është një tjetër faktor i përdorur për zgjedhjen e një metodologjije. Plan Driven metodologjitë përfshijnë shumë "kohë të humbur" tek rezultatet si dokumentacioni, dokumentet e projektimit, shkruarja e analizave etj. Duke përfunduar se kur kemi kohë të kufizuar, përdorimi i metodologjisë Agile do të jetë më mirë. [11]

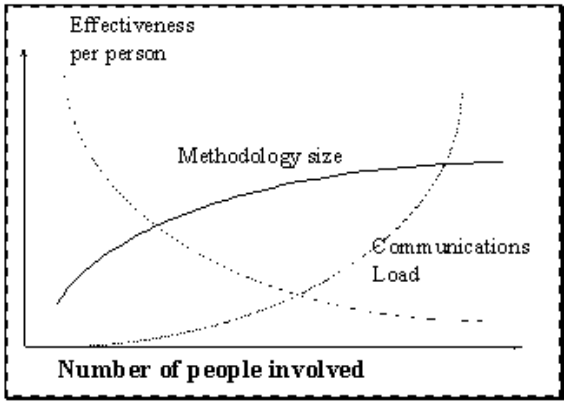

Figura 7 - Efekti i madhësis së projektit [10]

- Gjysma e vlerave të Agile mirren me faktorët njerëzorë "Individët dhe ndërveprimet" dhe "bashkëpunimi me klientët". Edhe NASA ka arritur në përfundimin se teknologjia dhe trajnimi nuk janë faktorë të mëdhejnë, "Praktika më efektive është përdorimi maksimal i potencialit njerëzor". Duke pasur njerëz të aftë dhe me përvojë në ekip është një faktor kyç për metodologjitë Agile. Inkujarimi i ekspertëve të fushës për të qenë pjesë e ekipit u jep zhvilluesve reagime të shpejta në implikimet e zgjedhjes së dizajnit për përdoruesit. Përshtatshmëria e konsumatorit është një tjetër faktor i madh, klienti merr fuqinë për të kontrolluar progresin dhe për të ndryshuar drejtimin e zhvillimit të softuerit gjatë çdo përsëritjeje. Fitimi në këtë nivel të angazhimit nga konsumatori bën metodologjinë Agile një proces më tërheqës se sa Plan Driven. Gjithashtu kultura e një organizate është një faktor i rëndësishëm kur zgjedhim një metodologjië. Nëse një organizatë është e fortë, e cila nuk është e përgjegjshme për ndryshimet dhe ka shumë rregulla dhe procedura nuk mund të jetë e susksesshme duke përdorur metodologjinë Agile. Përndryshe, nëse një organizatë është e përgjegjshme apo fleksibile, ata duhet të miratojnë përshtatshmërinë drejt ndryshimeve si kulturën e tyre nëse duan të aplikojnë metodat Agile. [10]
- Faktorët e rrezikut- Faktorët më të rëndësishëm të rrezikut në zhvillimin e një procesi softuerik janë: rëndësia e projektit dhe pergjigjja ndaj ndryshimeve. Metodat Agile janë përdorur në të gjitha kërkesat që mund të ndërtohen shpejtë dhe nuk kërkojnë siguri të gjerë të cilësisë. Sistemet kritike, të besueshme dhe të sigurta janë më të përshtashme për metodologjinë Plan Driven. Nëse një sistem është kritik, të gjitha kërkesat duhet të përcaktohen para zhvillimit të softuerit. Përkufizimi i varfër do të rezultojë në më shumë dëme nga defekte të pa zbuluara. Përgjigjja e ndryshimeve mund të zgjidhet duke

përdorur metodën Agile. Praktikat e përcaktuara në metodat Agile lejojnë trajtimin e ndryshimeve, të tilla si reagime të vazhdueshme nga klienti dhe zhvillim të shkurtër përsëritës. Duke përmbledhur atë që u tha më lartë, Tabela 2 tregon bazat për metodat Agile dhe Plan Driven, e cila përfshinë grupe të kushteve në të cilat ato kanë më shumë gjasa të kenë sukses.

Karakteristikat e projektit	Agile Diskriminuesi	Plan Driven Diskriminuesi
Objektivi primar	Rapid Value	Siguri të lartë
Kërkesat	Kryesisht emergjente, ndryshimi i shpejtë, i panjohur	E njohur më herët, kryesisht e qëndrueshme
Madhësia	Ekipe dhe projekte të vogla	Ekipe dhe projekte të mëdha
Arkitektura	E dizajnuar për kërkesa të tanishme.	E dizajnuar për kërkesat e tashme dhe të pritshme.
Planifikimi dhe kontrolli	Planet e internalizuara, kontroll cilësor.	Planet e dokumentuara, kontroll sasior.
Klientët	Dedicated, knowledgeable, collaborated, collocated onsite customers	As needed customer interactions, focused on contract provisions
Zhvilluesit	Agile, i ditur, bashkëvendosur, bashkëpunues.	Plani i orientuar, aftësi të mjaftueshme për qasjen në njohuri të jashtme.
Riprodhimi	I lirë	I shtrenjtë
Rreziqet	Rreziqe të panjohura, ndikim i madh.	Rreziqet kuptohen mire, ndikimi i vogël.

Table 2 - Karakteristikat e projekteve në "Agile" dhe "Plan Driven" [10]

4.3.1. Analiza e kërkesave tek Agile dhe Plan Driven metodologjive

Kërkesat në metodologjinë Plan Driven analizohen dhe përcaktohen në fillim, dhe pastaj dokumenti i specifikimit dhe plani i menaxhimit te projektit prodhohen. Pasi këto dokumente miratohen nga ana e klientit, ata bëhen bazë për fazën e dizajnimit e cila prodhon specifikimet arkitekturore dhe kërkesat e detajuara të dizajnit. Analiza e orientuar në objekte (OO) përdor shumë mjete të UML analizave. UML diagramet e stukturës përdoren për të identifikuar strukturën kryesore të sistemit. Funksionaliteti kryesor dhe sjellja e sistemit identifikohen duke përdorur diagramet UML të sjelljeve. UML diagramet e ndërveprimeve përdoren për të specifikuar rrjedhën e kontrollit dhe të dhënave midis komponentëve të sistemit. Si rrjedh e kësaj analiza e orientuar ne objetke (OO) përbëhet nga tri elemente të rëndësishme të modelimit që janë paraqitur në Figura 8.

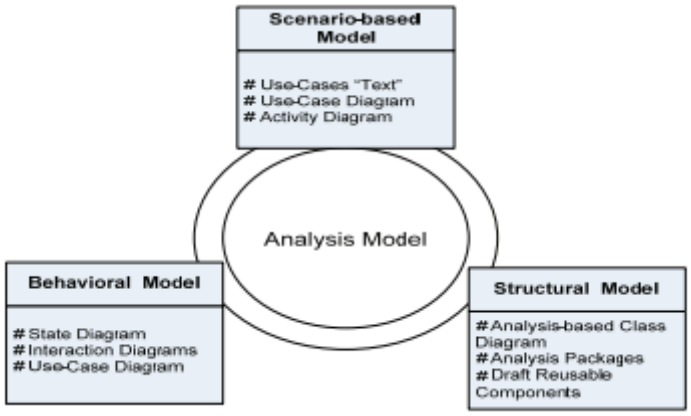

Figura 8 - Modeli i analizës [2]

Pasi kërkesat të analizohen dhe modelohen, ato janë të dokumentuara mirë në bazë të një standardi të dokumentacionit të zgjedhur (siq është Defense System Software Development Dod-Std-2167). Të gjitha kërkesat janë supozuar të jenë statike. Mundësia e ndryshimeve është shumë e vogël. Të gjitha kërkesave i'u ipet i njejti prioritet. Ato trajtohen në mënyrë të barabartë në përpjekje të analizave. Në Figura 10 tregohet pasi kërkesat janë mbledhur dhe analizuar, konsumatori pauzon lidhjen e tij me projektin, derisa produkti të përfundojë. Validimi nga klienti kryhet vetëm kur projekti arrinë në fazën e pranimit "duke përfshirë testet e paranimit, instalimin dhe dërgimin".

Ndryshe nga shumica e metodave Plan Driven, të cilat japin një sistem monolitik pas një kohe të gjatë të zhvillimit, metodat Agile fokusohen në gjenerimin e hershëm dhe njoftime të vogla për punën e përfunduar, duke përdorur teknika kryesishtë bashkëpunuese (XP përdorë programimin çift dhe riprodhimin). Kërkesat tek metodologjitë Agile zbatohen në mënyr përsëritëse. Kërkesat më të rëndësishme realizohen herën e parë. Gjatë zhvillimit klienti punon në vend, sikur antarë i ekipit, ku ata mbajnë përgjegjësinë e kontrollit të kërkesave. Për këtë arsye ata kanë të drejtë për të përcaktuar kërkesa të reja, për të ndryshuar ato ekzistues dhe të bëjnë ripririotizimin e kërkesave që ata i shohin të arsyeshme Figura 9. Ata gjithashtu duhet të jenë përgjegjës për marrjen e vendimeve dhe dhënien e infromacioneve në kohën e duhur. Ata ofrojnë përshtypje të vazhdueshme për zhvilluesit, të kuptuarit e kërkesave, ecurinë dhe cilësinë e produktit.

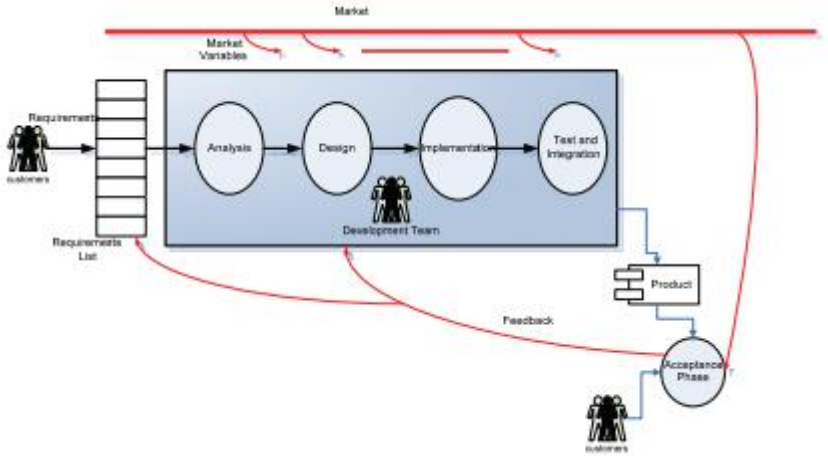

Figura 9 - Validimi nga konsumatori në metodologjinë Agile [2]

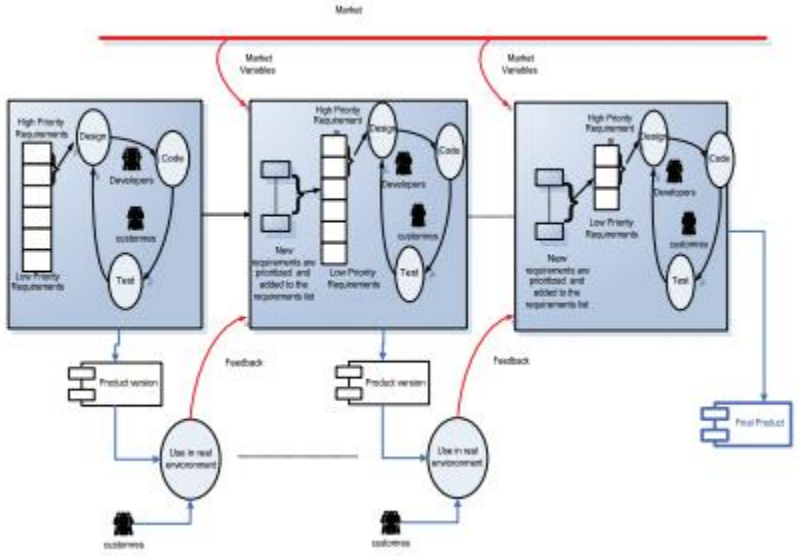

Figura 10 - Validimi nga konsumatori në metodologjinë Plan Driven [2]

4.3.2. Dizajni tek Agile dhe Plan Driven metodologjive

Dizajnimi tek metodologjija Plan Driven fillon pasi të jenë analizuar, modeluar dhe dokumentuar kërkesat. Tek Plan Driven, ekipi i dizajnit (dizajnerët) janë të ndarë nga ekipi i zhvillimit (programerët). Dizjanerët mendojnë për çështjet e mëdha paraprakisht. Ata nuk shkruajnë kod pasiqë ata nuk e ndërtojnë softuerin mirpo ata e dizajnojnë atë. Pra ata përdorin shumë teknika të UML dizajnit që është larg detajeve të programimit dhe i lejon dizajnerët të punojnë në një nivel

më abstrakt. Pasi dizjani përfundohet, dizajnerët mund ta dorëzojnë atë tek programerët për ta shkruar kodin. Që kur dizajnerët mendojnë në një shkallë më të madhe, ata mund të shmangin një sërë vendimesh taktike. Duke i'u referuar dokumenteve të elementeve të analizave të modelit Figura 8, dizajnerët krijojnë katër modele të dizajnit që janë të nevojshme për një specifikim të plotë të dizajnimit Figura 11. Modelet arkitekturore përdorin infromacionet e dala nga domeni i aplikimit, si dhe modelin e analizës (modeli strukturorë) për të nxjerrur një paraqitje të plotë strukturore të softuerit, nensistemet e tij dhe komponentët e tij. Ndërfaqet e modelit të dizajnit paraqesin ndërfaqet e brendshme dhe të jashtme të sistemeve tjera, pajisjet, rrjtet apo procedura të tjera ose konsumatorët e informacionit. Ato gjithashtu paraqesin ndërfaqet e brendshme midis komponentëve të ndryshme të dizajnit. Një aspekt tjetër i ndërfaqeve të modelit të dizajnit është modelimi i ndërfaqeve me përdoruesit e sistemit. Modelet e nivelit-komponent përcaktojnë secilën prej komponentëve të jashtme dhe të brendshme që popullojnë arkitekturën e sistemit. Modelet e dizajnit të nivelit-shpërndarje përcaktojnë elementet për ndarjen e arkitekturës, komponentët e tij dhe ndërfaqet për konfigurimin fizik ku do të ruhet softueri. [6]

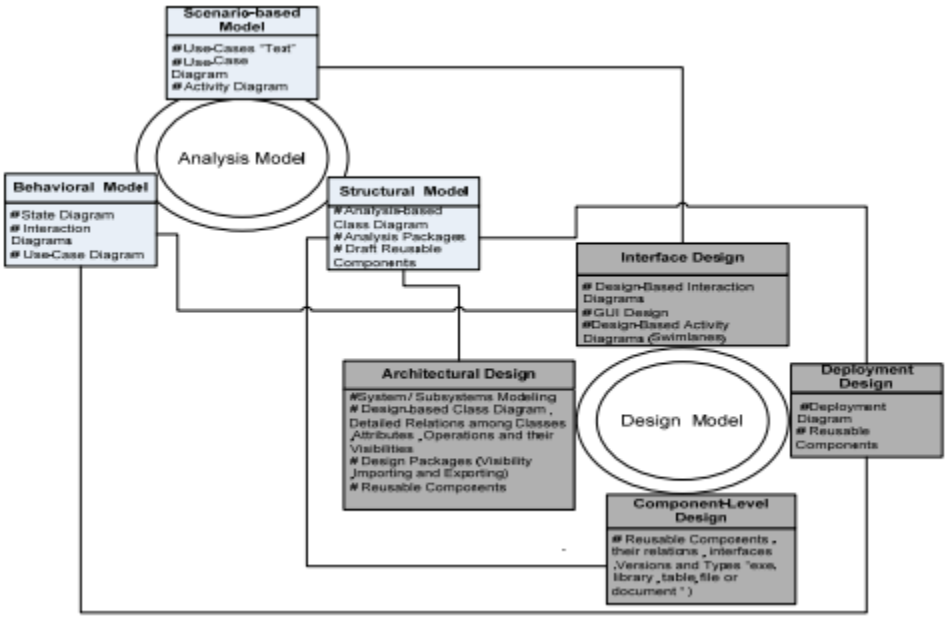

Figura 11 - Modelet e dizajnuara [2]

Të gjitha aktivitetet e dizajnit janë të dokumentuara mirë duke përdorur një standard të dokumentacionit që është zgjedhur në fazat e analizës. Këto dokumentacione do të jenë burimi kryesorë për programerët që të implementojnë sistemin. Dizajni tek Agile në mënyrë rigoroze ndjek parimin (e mban të thjesht/ dhe dizajnohet sot për sot). Tek metodologjija Agile supozohet se sa më shumë dizajn për të ardhmen, rezulton me dizajn më kompleks gjë që çon në kosto më të panevojshme siç tregon Figura 12 dhe Figura 13. Përveç kësaj, dizajni ofron udhëzime të implementimit për një njësi të kërkesave (XP përdor user stories), ashtu siç është shkruar asgjë më shumë apo më pak. Dizajni për funksionalitet shtesë (pasiqë nevojitet më vonë) mënjanohet. Agile përdor mjete të thjeshta për të ruajtur thjeshtësinë. Ajo nuk elaboron në përdorimin e mjeteve komplekse dhe të detajuara. Për shembull XP përdor Class-Responsibility-Collaborator (CRC) modelin. [12]

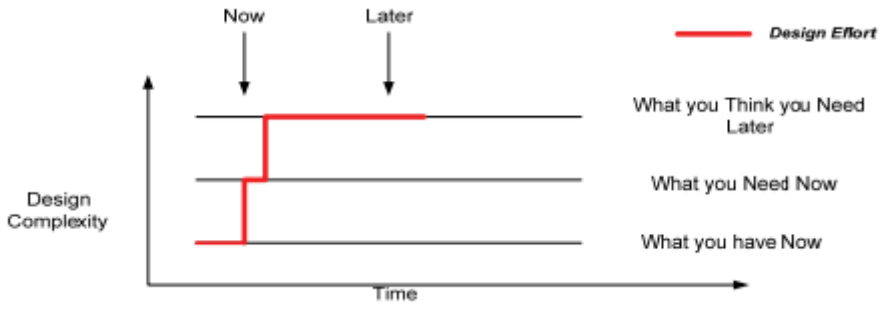

Figura 12 - Kompleksiteti i dizajnimit në të ardhmen [2]

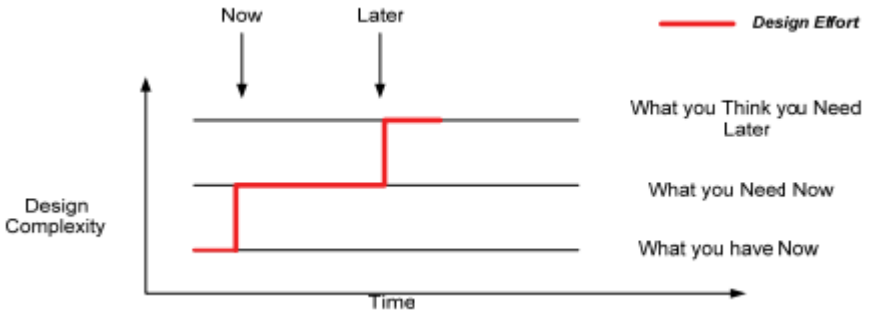

Figura 13 - Veshtirësia e dizajnit në të sotmen [2]

Nëse një problem i vështirë në dizajn është paraqitur, metodologjitë Agile rekomandojnë krijimin e menjëhershëm të një prototipi operues të asaj pjese të dizajnit (siç quhet në XP zgjidhje e lidhur). Qëllimi i kësaj është për të ulur rrezikun, kur fillon zbatimi i vërtet dhe për të vërtetuar njësitë origjinale të kërkesave. Agile inkujaron teknikën e riprodhimit Figura 14. Riprodhimi është një teknikë riorganizuese që përmirëson, thjeshton dhe maksimizon efikasitetin e dizajnit (ose kodi) i një komponente pa ndryshuar funksionin apo sjelljen e tij. Kur softueri riprodhohet, kodi ekzistues kontrollohet për elemente të papërdorura, përsëritje, algoritme jo efikase, strukturim jo efikase të të dhënave, apo për çfarëdo gabimi tjetër të dizajnit që mund të korigjohet për të pasur një dizajnë sa më të mirë. [6]

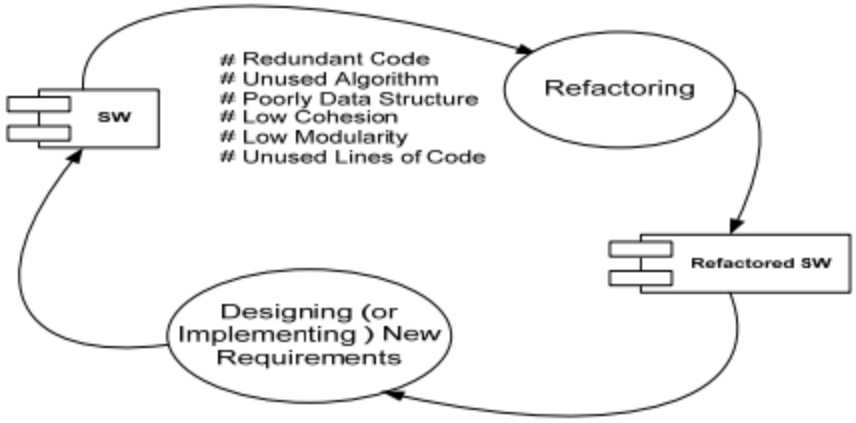

Figura 14 – Riprodhim [2]

5. Metodologjija

Për punimin e kësaj teme të diplomës kam përdorur teknika të ndryshime të hulumtimit. Duke filluar nga libri “Software Engineering”- Ian Sommerville nga i cili jam bazuar në shqyrtimin e secilës metodologji në mënyrë të detajuar. Gjithashtu kam përdorur edhe shumë arikujt shkencorë dhe dokumente shkencore të publikuara nga universitete dhe ekspertë të kësaj fushe. Për krahasimin në mes dy metodologjive kam përdorur “A Comparison between Agile and Traditional Software Development Methodologies”-Awad, artikull ky i publikuar për universitetin e Australis. Përshkrimi i grafeve dhe diagrameve të dizjanit janë bazuar në artikullin “Planned Methodologies vs. Agile Methodologies under the Pressure of Dynamic Market”- M. Kamel, I. Bediwi and M. Al-Rashoud, i publikuar për universitetin e Arabisë Saudite. Kam përdorur edhe shumë libra të tjerë të cilët janë të cekur në kapitullin e biblografisë. Në përgjithësi kërkimi në internet dhe shfletimi i kësaj literature ishin bazë për realizimin e punimit.

6. Diskutime dhe Konkluzione

Metodologjitë për zhvillimin e softuerit janë mënyra apo rruga e ndjekur nga ekipi zhvillues që na ndihmon për ta arritur një punë më efikase dhe më profitabile. Gjatë krahasimit të këtyre dy metodologjive Agile dhe Plan Driven kam arritur të gjejë një numër të dallimeve në projekte ku secila prej tyre kanë qenë të suksesshme. Një pikë e dallimit është përshtatshmëria e objektivave të secilës metodologjië për një projekt. Pikat tjera përfshijnë madhësinë e projektit në aspekt të numrit të njerëzve në ekip, kompleksitetin, volumin e softuerit dhe llojin e mjedisit të biznesit brenda të cilit është zhvilluar projekti.

Qëllimet kryesore të metodologjisë Agile janë vlerat e shpejta dhe përgjegjshmëria për ndryshim. Parimi kryesorë i kësaj metodologjije thotë: “Prioriteti më i lartë është arritja e knaqësisë së klientit me anë të ofrimit të hershëm dhe të vazhdueshëm të programit funksional”. Aktualisht, proceset Agile punojnë më së miri me ekipe të vogla-mesme që punojnë për aplikacione relativisht të vogla. Ka pasur projekte Agile të suksesshme me deri në 250 persona. Mirëpo menaxherët apo udhëheqësit e këtyre projekteve tregojnë se është një mënyrë shumë e madhe dhe e vështirë. Për këtë një projekt i madh Agile duhet të miratojë planet tradicionale dhe specifikimet në mënyrë që të merret gjithnjë e më shumë me ndërveprimet komplekse shumëdimensionale mes elementeve të projektit. Metodologjija Agile përqëndrohet në dhënien e një produkti të veçantë softuerik në kohë në mënyrë që të përmbush plotësisht klientin. Si i tillë, objekti shqetësues është i fokusar në produktin në fjalë dhe në përgjithësi injoron problemet që mund të ndodhin më vonë. Kjo metodë fuqimisht varet nga dedikimi për të mbajtur projketin të fokusuar në shtimin e vlerave të shpejta për organizatën. Kjo zakonisht punon shumë mirë në nivel të projektit.

Qëllimet kryesore të metodologjisë Plan Driven janë parashikushmëria, stabiliteti dhe siguria. Planet, produktet e punës, verifikimi dhe strategjitë e validimit të metodës Plan Driven i përkrahin këto qëllime. Metoda Plan Driven funksionon më mirë në projekte të mëdha. Planet, dokumentacionet dhe proceset sigurojnë komunikim dhe koordinim më të mirë nëpër grupe të mëdha. Në projekte të mëdha ku numri i personave në ekip është i madhë nevojitet specifikimi i detajuar i softuerit, pra nuk shihet asnjë mënyrë për të shmangur një aktivitet të kësaj natyre në këto sisteme, dhe absolutisht asnjë mënyrë për të trajtuar problemet që shfaqen përmes përdorimit të metodologjisë Agile. Plan Driven metodologjija punon më së miri kur kërkesat janë

kryesisht të përcaktueshme paraprakisht (përfshirë një prototip) dhe mbeten relativisht të qëndrueshme. Nëse ndryshimi i normave të kërkesave nuk është i madh atëherë këto ndryshime janë të pranueshme. Mirëpo fatkeqësisht nëse këto norma gjithnjë e më shpesh rriten atëherë metodat tradicionale të dizajnit të softuerëve stabil fillojnë të zgjidhen. Kjo metodologji në përgjithësi varet nga një formë e kontratës në mes të zhvilluesve dhe klientit si bazë për marrëdhënie me klientin. Ata përpiqen për t'u përballur me problemet e parashikueshme dhe formalizimin e zgjidhjeve në një marrëveshje të dokumentuar. Kjo ka disa avantazhe, veçanërisht në situata të qëndrueshme. Zhvilluesit e din se çfarë duhet të bëjnë dhe klineti e di se çfarë ata janë duke marrur dhe janë më të aftë për të vazhduar me përgjegjësitë e tyre operacionale. Nga kjo mund të konkludojmë se të dy metodologjitë janë të përshtatshme për projekte të ndryshme vetëm se duhet të shqyrtohen mirë parimet e metodologjive si madhësia projektit, volume i tij, kompleksiteti dhe të shikohet se cilës prej tyre i përshtatet projekti.

7. Biblografia

[1] I. Sommerville, Software Engineering, 9th ed., Boston, Massachusetts: Pearson, 2011.

[2] S. Haunts, Agile Software Development Succinctly, Morrisville, NC, USA: Syncfusion, Inc., 2015.

[3] James Shore, Shane Warden, The Art of Agile Development, 1st ed., Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc., 2007.

[4] H. Kniberg, Scrum and XP from the trenches - How we do Scrum, 2nd ed., USA: C4Media, 2015.

[5] K. Beck, Extreme Programming Explained – Embrace Change, Massachusetts: Addison Wesley Longman, Inc., 2000.

[6] M. Kamel, I. Bediwi, M. Al-Rashoud, "Planned Methodologies vs. Agile Methodologies under the Pressure of Dynamic Market," *Journal King Abdulaziz University,* vol. 21, pp. 19-35, 2010.

[7] N. Ionel, "Critical Analysys Of The Scrum Project Management Methodology," The Academy of Economic Studies Bucharest, Bucharest, 2008.

[8] M. A. Awad, "A Comparison between Agile and Traditional Software Development Methodologies," The University of Western Australia, Australia, 2005.

[9] P. R. S. Pressman, Software Engineering - A PRACTITIONER'S APPROACH, 5th ed., New York, USA: The McGraw-Hill Companies, Inc., 2001.

[10] D. W. W. Royce, "Managing the Development of large Software Systems," *IEEE WESCON,* pp. 1-9, 1970.

[11] D. Leffingwell, Scaling Software Agility - Best Practice for Large Enterprises, 1st ed., Boston: Addison-Wesley Professional, 2007.

[12] K. Schwaber, M. Beedle, Agile Software Development with Scrum, 1st ed., Upper Saddle River, NJ: Prentice – Hall, 2011.