ÇEVİK YAZILIM GELİŞTİRME METODLARI İÇİN MİMARİ DEĞERLENDİRME SÜRECİ ÖNERİSİ

2018

TEZSİZ YÜKSEK LİSANS

DÖNEM PROJESİ

BİLGİSAYAR MÜHENDİSLİĞİ

Fatih ÇETİN

**ÇEVİK YAZILIM GELİŞTİRME METODLARI İÇİN MİMARİ DEĞERLENDİRME SÜRECİ ÖNERİSİ**

**Fatih ÇETİN**

**Karabük Üniversitesi**

**Fen Bilimleri Enstitüsü**

**Bilgisayar Mühendisliği Anabilim Dalında**

**Tezsiz Yüksek Lisans Dönem Projesi**

**Olarak Hazırlanmıştır.**

**KARABÜK**

**Ocak 2018**

Fatih ÇETİN tarafından hazırlanan “ÇEVİK YAZILIM GELİŞTİRME METODLARI İÇİN MİMARİ DEĞERLENDİRME SÜRECİ ÖNERİSİ” başlıklı bu projenin Tezsiz Yüksek Lisans Dönem Projesi olarak uygun olduğunu onaylarım.

Doç. Dr. Oğuz FINDIK ..........................

Dönem Projesi Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

....../….../2018

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu Dönem Projesi ile, Tezsiz Yüksek Lisans derecesini onamıştır.

Prof. Dr. Filiz ERSÖZ ..........................

Fen Bilimleri Enstitüsü Müdür V.

*“Bu projedeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim.”*

Fatih ÇETİN

# ÖZET

**Tezsiz Yüksek Lisans Dönem Projesi**

**ÇEVİK YAZILIM GELİŞTİRME METODLARI İÇİN MİMARİ DEĞERLENDİRME SÜRECİ ÖNERİSİ**

**Fatih ÇETİN**

**Karabük Üniversitesi**

**Fen Bilimleri Enstitüsü**

**Bilgisayar Mühendisliği Anabilim Dalı**

**Proje Danışmanı:**

**Doç. Dr. Oğuz FINDIK**

**Ocak 2018, 19 sayfa**

Yazılım geliştirme süreci iki kategori halinde değerlendirilebilir : Geleneksel yöntemler ve çevik yöntemler. Geleneksel yöntemlerdeki ağır dökümantasyon işlemleri projede gecikmelere sebep olurken , daha az döküman ve hafif süreçlere sahip çevik yöntemler projelerdeki gecikmeleri azaltmaktadır.Çevik yöntemler müşteri memnuniyeti, değişimlere hızlı cevap verme ve daha kısa zamanda çıktı üretmek üzerine yoğunlaşır. Son yıllarda, yüksek kalitede ürünler elde etmek için, yazılım süreçlerinde mimari kullanımı önemli bir konu olmuştur. Çevik metodlarda yazılım mimarisi becerilerinin kullanımı, kalite niteliklerine odaklı, uygun yapıda bir yazılım çıkmasında önemli rol oynamaktadır. Bu çalışmada yazılım geliştirme sürecinin iyileştirilmesi için çevik yöntemlerle yazılım mimarisinin birleştiştirilmesi üzerine süreç geliştirmeleri önerilecektir.

**Anahtar Sözcükler :** Çevik yazılım geliştirme, yazılım mimarisi, ATAM, SAAM.

# ABSTRACT

**Term Project based M. Sc. Degree**

**(Without Thesis)**

**ARCHITECTURAL EVALUATION PROCESS RECOMMENDATION FOR AGILE SOFTWARE DEVELOPMENT METHODS**

**Fatih ÇETİN**

**Karabük University**

**Graduate School of Natural and Applied Sciences**

**Department of Computer Engineering**

**Project Supervisor:**

**Assoc. Prof. Dr. Oğuz FINDIK**

**January 2018, 19 pages**

We can categorize software development methods in to two groups which are Tradational and Agile methods. Beside of the heavy documentation on traditional methods that cause delays on release process, agile development process decrease the delays of release time by having less documentation and lightweight processes. Agile methods focuses on customer satisfaction, fast response to changes and release in short term. In last decades , using architecture skills has a key factor on software development process. Using software architecture on agile methods helps to develop structured , quality attributes focused application. This study make a proposal about adopting the architectural processes into the agile method as a process of agile method in order to improve the agile processes.

**Key Words :** Agile software development, software architecture, ATAM, SAAM.

# TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde, oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Doç. Dr. Oğuz FINDIK’a sonsuz teşekkürlerimi sunarım.

# İÇİNDEKİLER

**Sayfa**

KABUL ii

[ÖZET iv](#_Toc504655616)

[ABSTRACT v](#_Toc504655617)

[TEŞEKKÜR vi](#_Toc504655618)

[İÇİNDEKİLER vii](#_Toc504655619)

[ŞEKİLLER DİZİNİ ix](#_Toc504655620)

[BÖLÜM 1 1](#_Toc504655621)

[GİRİŞ 1](#_Toc504655622)

[1.1. LİTERATÜR ÖZETİ 1](#_Toc504655623)

[1.2. PROJENİN AMACI 4](#_Toc504655624)

[BÖLÜM 2 6](#_Toc504655625)

[ÇEVİK YAZILIM GELİŞTİRME 6](#_Toc504655626)

[2.1. SCRUM 8](#_Toc504655627)

[2.1.1. Scrum Ritüelleri 9](#_Toc504655628)

[2.1.1.1. Geliştirme Planlama (Sprint Planning) 9](#_Toc504655629)

[2.1.1.2. Günlük Toplantı (Daily Stand-up) 10](#_Toc504655630)

[2.1.1.3. Geliştirme Ön Sunumu (Sprint Review) 10](#_Toc504655631)

[2.1.1.4. Geliştirme Geçmiş Değerlendirmesi (Sprint Retrospective) 10](#_Toc504655632)

[BÖLÜM 3 12](#_Toc504655633)

[YAZILIM MİMARİSİ 12](#_Toc504655634)

[3.1. YAZILIM MİMARİSİ DEĞERLENDİRME 15](#_Toc504655635)

[BÖLÜM 4 17](#_Toc504655636)

[SONUÇ VE DEĞERLENDİRME 17](#_Toc504655637)

[KAYNAKLAR 18](#_Toc504655638)

**Sayfa**

[ÖZGEÇMİŞ 19](#_Toc504655639)

# ŞEKİLLER DİZİNİ

**Sayfa**

[Şekil 2.1. V model. 6](#_Toc504655808)

[Şekil 2.2. Waterfall. 7](#_Toc504655809)

[Şekil 2.3. Çevik yazılım süreçleri. 8](#_Toc504655810)

[Şekil 2.4. Scrum ritüelleri. 9](#_Toc504655811)

[Şekil 3.1. 4 + 1 mimari tasarım. 13](#_Toc504655812)

[Şekil 3.2. ISO 25010. 13](#_Toc504655813)

[Şekil 3.3. Kalite ağacı örneği. 14](#_Toc504655814)

# BÖLÜM 1

# GİRİŞ

## 1.1. LİTERATÜR ÖZETİ

Son yıllarda yazılım mimarisi konusu , kaliteli yazılım geliştirme için göz ardı edilemez bir konu olarak yazılım mühendisliğinde daha adından bahsedilir bir hal almıştır. Yazılım mimarisi yazılım kalitesi ne kadar gerekli olsada uygulama açısında sıkıntılı durumlar mevcuttur. Öncelikle yazılım mimarisi konusu soyut bir kavram olduğundan dolayı değişik tanımlamalara sahiptir. Bunun yanında mevcut bir sistemin mimari yapısının değerlendirmesi için belli bazı hazırlıkların yapılması gerekmektedir. Yazılımın mevcut mimari yapısının değerlendirilmesi için farklı metodlar geliştirilmiştir.

Ali Athar, Rao Muzamal Liaqat, Farooque Azam tarafıından yayınlanan “A Comparative Analysis of Software Architecture Evaluation Methods” isimli makalede mevcut metodların farkları , benzerlikleri ve kullanım alanları konularına değinilmiştir.

Harshavardhan Metla, Dharmendhra makineni, Manikanta Ellenki, Pranay Ratakonda

“A Metric Based Approach for Managing Architecture-Related Impediments” isimli çalışmalarında mimari ile alakalı engellerin çözümü için daha ölçümlenebirlir bir mthod önerisinde bulunmuşlardır.

Gelenesel yöntemlerdeki mimari değerlendirme metodları genellikle yazılımın kalite beklentilerine göre hazırlanmış mimari dökümanlarına göre yapılmaktadır. Colin J. Neill ve Nil H. Kilicay’ın “A Prescriptive Approach to Quality-Focused System Architecture” isimli çalışmalarında mimari değerlendirme metodlarında kalite odaklı yaklaşımları konu almışlardır.

Çevik yazılım geliştirme metodlarının kullanımının artması ile birlikte mimari yaklaşımlarında çevik bir yapıya uygun olması gerekliliği ortaya çıkmıştır. Farzaneh Akbari ve Sayed Mehran Sharafi’nin “A Review to the Usage of Concepts of Software Architecture in Agile methods” isimli çalışmasında yazılım mimarisi konseptinin Çevik metodlarda kullanımı ile ilgili bir literatür taraması mevcuttur.

Mimari değerlendirmenin temeli olarak kabul edilen mimari dizayn üzerine değerlendirme ile ilgili prensip ve uygulamalar David L. Parnas ve David M. Weiss tarafından yazılan “Active design Reviews : Principles and Practices ” isimli makalede konu alınmıştır.

Mimari değerlendirme yapmaya yardımcı olacak olan yazılımlarda, mimari yaklaşımların öneminin artması ile çoğalmıştır. Bu araçların bir örneği olan ARCAN’ın, Francesca Arcelli Fontana, Ilaria Pigazzini, Riccardo Roveda, Damian Tamburri tarafıından kaleme alınan “Arcan: a Tool for Architectural Smells Detection” isimli makalede mimari sıkıntıların belirlenmesinde nasıl kullanılacağı üzerine b,lg,lendirme mevcuttur.

Mimari değerlendirme konusunda , senaryo bazlı değerlendirme methodlarından biri olan ATAM (application tradeoff analysis method) üzerine bir vaka analizi, I Made Putrama, Kadek Teguh Dermawan, Gede Rasben Dantes ve Kadek Yota Ernanda Aryanto tarafından yazılan “Architectural Evaluation of Data Center System using

Architecture Tradeoff Analysis Method (ATAM): A Case Study” isimli çalışmadır.

ATAM uygulasının genellikle mimari tasarımın bitimide uygulanabilir olduğu gerçeğinden yola çıkan ve mimari tasarımın önceki adımlarında da değerlendirme gereksimini karşılayacak olan bir metod, Felix Bachmann tarafından “Give the Stakeholders What They Want: Design Peer Reviews the ATAM Style” isimli çalışmada konu alınmıştır.

Jens Knodel, Matthias Naab tarafından yazılan “How to Evaluate software Architecture” isimli makalede mimari tasarımın değerlendirmesi üzerine yapılan bir worksho çalışmasının detayları verilmiştir.

Çevik yöntemlerin bir özelliğide dağıtık takımlar üzerinde çalışma imkanı vermesidir. Bu şekildeki takımlarda roller de takım yapısana uygun olarak değişim göstermektedir. Ömer Uludağ, Martin Kleehaus, Xian Xu ve Florian Matthes’in “Investigating the Role of Architects in Scaling Agile Frameworks” isimli çalışmasında dağıtık olarak çalışan çevik takımlardaki yazılım mimarlarının görevleri tanımlanmaya çalışılmıştır.

Mimari tasarımın getirdiği dökümantasyon fazlalağının, çevik metodlarda nasıl uygulanması gerektiğine dair bir makale, Irit Hadar, Sofia Sherman, Ethan Hadar ve John J. Harrison Jr. tarafından “Less is More: Architecture Documentation for Agile Development” ismi ile yayınlanmıştır.

Çevik yazılım geliştirmeye mimari aktivitelerin entegre edilebilmesi konusuyla ilgili bir diğer makale Dan Sturtvant tarafından “Modular Architectures Make You Agile in the Long Run” ismiyle yayınlanmıştır.

Çevik metodlarda mimari bilginin yönetilmesi ve mimari tasarım uygulamalarının çeviş süreçlere katılması ile ilgili bir başka çalışma Gilberto Borrego, Alberto L. Morán, Ramón René Palacio Cinco, Oscar Mario Rodríguez-Elias ve Eloísa García-Canseco tarafından “Review of approaches to manage architectural knowledge in Agile Global Software Development” isimli makalede değerlendirilmiştir.

Mimari tasarımın değerlendirme süreci, sistemin kalite niteliklerinin belirlenmesi ile gerçekleştirebilir. Kalite kriterlerinin belirlenmesi, yazılım gereksinimlerinin öncelikledirilmesi ve önemlendirilmesi işlemlemleriyle doğrudan alakalıdır. Aneesa Rida Asghar, Atika Tabassum, Shahid Nazir Bhatti, Zainab Sultan ve Rabiya Abbas, “Role of Requirements Elicitation & Prioritization to Optimize Quality in Scrum Agile Development” isimli çalışmalarında gereksinimlerin önceliklendirme ve önemlendirlilme işlemlerin, yazılım kalitesi üzerine etkilerini araştırmışlardır.

Mimari tasarım değerlendirme methodlarından biri olan SAAM (software architectrue analysis method) , Rick Kazman, Len Bass, Gregory Abowd ve Mike Webb tarafından “SAAM: A Method for Analyzing the Properties of Software Architectures” isimli çalışmada detaylı olarak incelenmiştir.

Mimari tasarım bazı tanımlamalarda, sistemle ilgi yüksek seviye kararların bütünü olarak belirtilmiştir. Bu tanımdan anlaşılacağı üzere karar alam konusu mimari tasarımda önemli bir yer teşkil eder. Nitin Upadhyay, “SDMF: Systematic Decision-making Framework for Evaluation of Software Architecture” isimli çalışmasında mimari kararlar alınmasında yardımcı olacak bir yapıdan bahsetmiştir.

Mohsen Mohammadi, “SERVICE ORIENTED ARCHITECTURE: A REVIEW AND EVALUATION OF REFERENCE MODELS” isimli çalışmasında servis bazlı mimariler için temel alınan bazı yapıların gözden geçirme ve değerlendirmesini yapmıştır.

Bazı çevreler tarafından çevik geliştirme metodları ve klasik mimari tasarımın birlikte çalışmasının yapıları gereği zor olduğu konusunda yaklaşımlar mevcuttur. Philippe Kruchten, “Software Architecture and Agile Software Development—A Clash of Two Cultures?” isimli çalışmasında bu konu üzerine yoğunlaşmıştır.

## 1.2. PROJENİN AMACI

Klasik yazılım geliştirme süreci bazı alt süreçlerin birbiri ardına uygulanmas ve bir önceki sürecin çıktısının bir sonraki sürece girdi olarak etkilediği toplamda uzun bir süreçler topluluğu olarak görülebilir. Bundan dolayı klasik yazılım geliştirme süreçlerinde önceki adımlarda yapılacak hatalar sonraki adımlarıda olumsuz etkileceyecek ve süreçler arasındaki geçişler maliyet olarak çok fazla olduğu için bu hatalar ürün kalitesi, maliyet ve teslimat zamanında gecikme olarak kendini gösterecektir.

Yazılım mimarisi konusu tam bu noktada geliştirme sürecinin doğru yolda ilerlemesi için ve yukarıda belirtilen sıkıntıların önceden öngülebilmesi için yüksek seviyeden düşük seviyeye doğru sistemin soyut tasarımın oluşturlması ve geliştrme sürecinde bu tasarıma sadık kalınması konularını içerir.

Çevik geliştirmenin yaygınlaşması ile beraber gereksinimlerin belirlenmesi, kodlama ve çalışan ürünün ortaya çıkarılması gibi adımlar klasik yöntemlerdeki süreçlerin daha küçük süreçler halinde dönüştürülmesi ve bu süreçlerin uygulanması halini almıştır. Çevik geliştirmenin temel prensibi değişikliğe hızlı tepki verebilmektir.

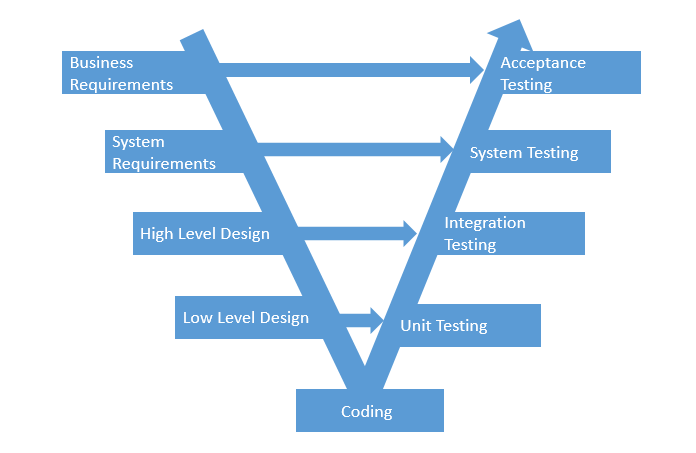
Yukarda belirtildiği üzere klasik yazılım geliştirmede yazılımın mimari tasarımı gereksinimler alındıktan sonra başlar, kodlama kısmından önce tamamlanır ve kodlama esnasında bu gereksinimlerin değişmeyeceği göz önüne alınarak kodlama tamamlanırdı. Çevik metodlarda gereksinim değişkliği her an olabileceği için kesin bir mimari tasarım yapılıp takp edilemez. Mimari tasarım gereksinimlerle doğrudan bağlantılı olduğu için gereksinmlerdeki değişiklik mimari tasarımda değişiklik yapılması gerekecektir. Bu durum mimari tasarımın değerlendirmesin, konusunda mimariye etki eden bir değişiklikten sonra geçerliliği kalmamaktır. Mimari tasarım değerlendirme metodları (ATAM, SAAM vb) klasik yöntemlerde uygulanabilir olmasına rağmen çevik yöntemlerde zaman ve değerlendirme yapacak uzman sıkıtısından dolayı çok uygun olamamaktadır.

Bu çalışmada mimari tasarımın çevik gelştirme süreçlerine ne şekilde ve nasıl eklenebileceğine dair bir öenride bulunulacaktır.

# BÖLÜM 2

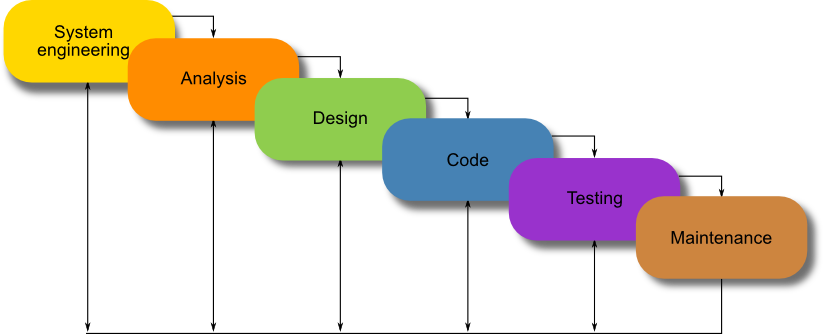
# ÇEVİK YAZILIM GELİŞTİRME

Yazılım dünyasında küreselleşme hemen hemen her sektördeki gibi bir fenomen halini almıştır. Dünyanın her yerindeki yetişmiş yazılım mühendisi küresel bir şekilde aynı aynı proje için mesafe problem olmadan çalışabilmektedir. Küreselleşme yazılım sektörüne yazılımcı maliyeti, uzak müşterilere ürün üretebilme, farklı kültürlerin getirilerinden faydalanma ve bunun gibi daha birçok konuda artı kazandırmıştır [1]. Yazılım sektörünün küreselleşme evriminin ilk zamanlarında klasik yöntemler (Şekil 2.1) tercih edildi. Klasik yöntemlerde öncelikle analiz ve dezayn daha sonra kodlama ve test yapılırdı. Bu işleyiş klasik yöntemlerde ürünün ortaya çıkma sürecini uzatıyordu.tüm bu süreç sonunda ortaya çıkan ürün tatmin edici olmayabiliyordu.



Şekil 2.1. V model.

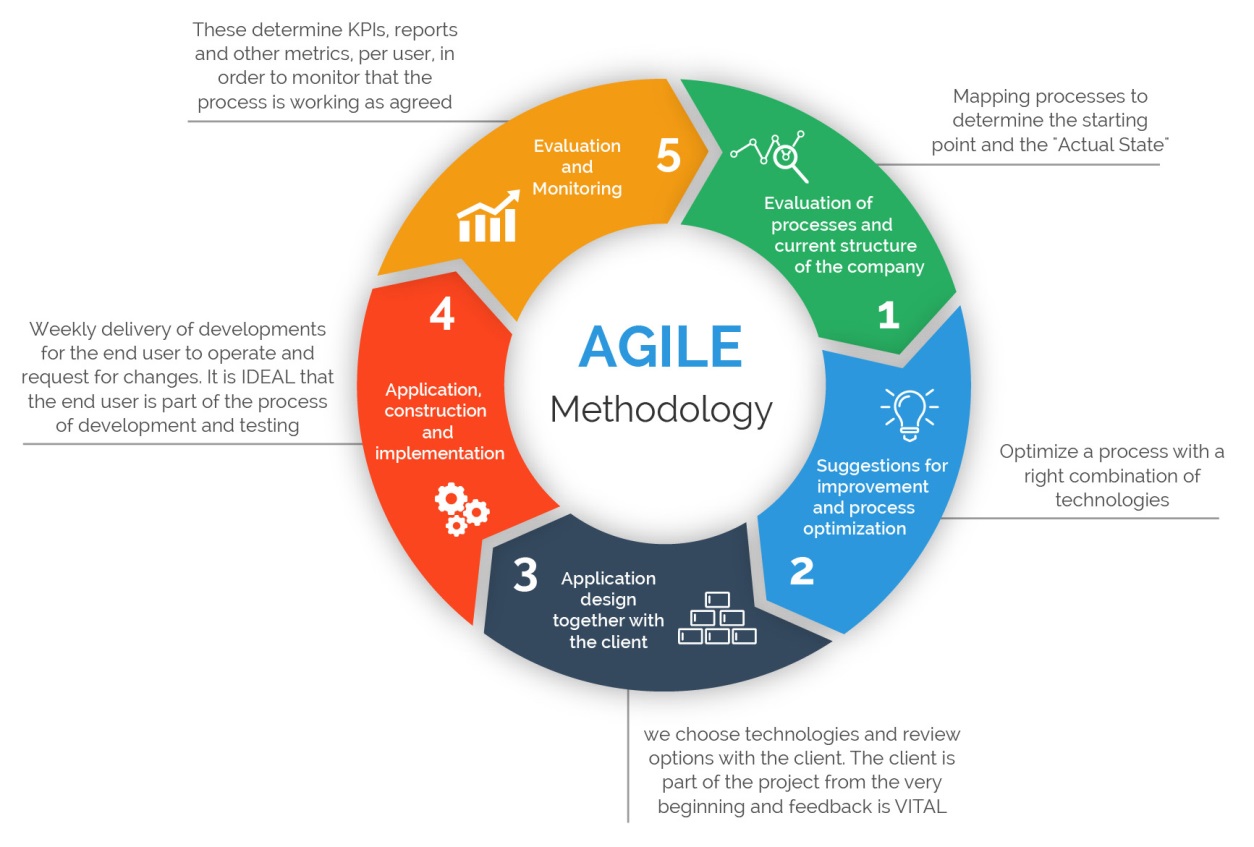
Bu sebeplerden dolayı , son yıllarda Çevik yazılım geliştirme metodları (Şekil 2.3) kürselleşen yazılım firmaları tarafından tercih edilir hale geldi[2]. Çevik yazılım geliştirmede esas olan tekrarlayan işlemlerle , herbir işlem sonucunda nihai ürene yaklaşan bir özellik eklemek ve hataları daha erken safhalarda yakalamaktır. Klasik uygulama geliştirme metodlarında ağır ve açık olmayan dökümantasyona karşı , çevik metodlarda daha az ve açık dökümanlar ve bunun yanında bilgi transferi için yüzyüze iletişim tercih edilir.



Şekil 2.2. Waterfall.

Çevik yazılım geliştirme metodları farklı altyapılar kullanalıbailir bunlardan bazıları extreme programlama (XP), Scrum, dinamik sistem geliştirme metodları (DSDM), adaptif yazılım geliştirme (ASD) and crystal’dir[3].

Çalışmamızda ki önerimizi Scrum metodu üzerine yapacağız ama yapılacak öneri alt yapı üzerine bir genişletme olacağı için diğer metodlara da uygulanmasında sıkıntı çıkmayacaktır. İlerki çalışmalarda önerilen metodun çıktılarının değerlendirilmesi sadece Scrum alt yapısı kullanan ekiplerden alınan geribildirimler üzerine yapılacaktır.



Şekil 2.3. Çevik yazılım süreçleri.

## 2.1. SCRUM

Scrum çevik yazılım geliştirme metodunu uygulamaya yardım eden bir altyapıdır[1-3]. Çevik yazılım geliştirmenin temeli olan sabit uzunluktaki tekrarlayan geliştirme zamanı (sprint) ve her tekrarda ürüne değer katan bir ekleme ortaya çıkartılması esasına dayanır[5].

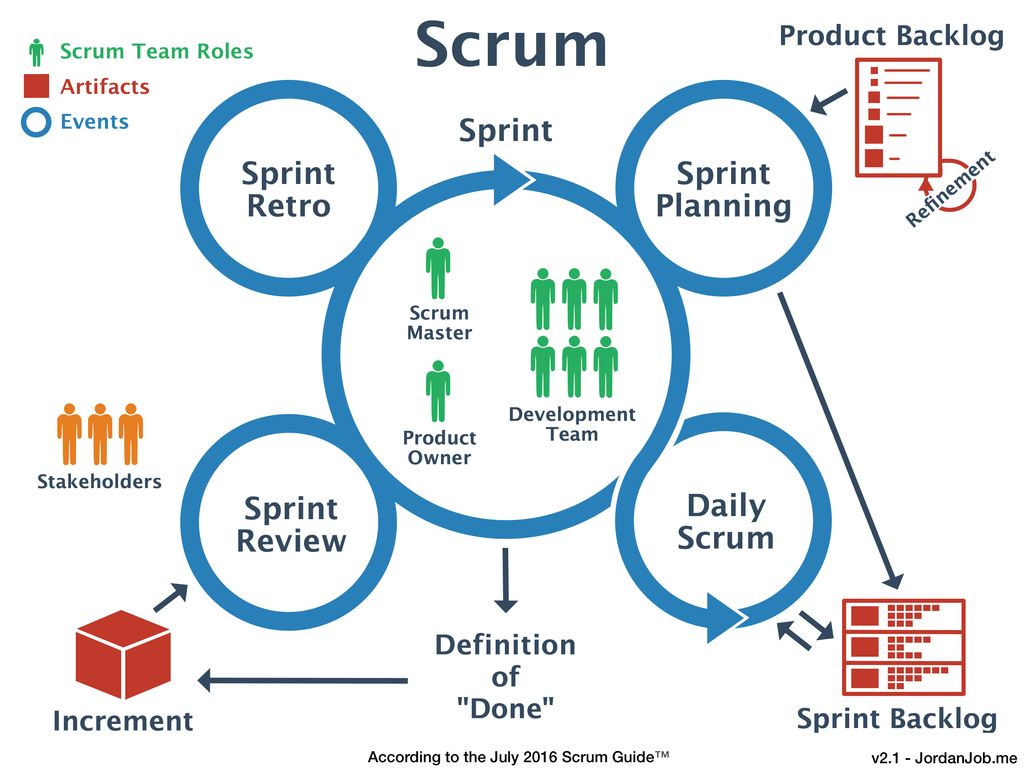
Scrum altyapısında bir geliştirme zamanı içerisinde 4 adet ritüel vardır. Bunlar :

* Geliştirme Planlama (sprint planing)
* Günlük toplantı (daily stand-up)
* Geliştirme Ön sunumu (sprint review)
* Geliştirme Geçmiş Değerlendirmesi (sprint retrospective)

Scrum altyapısı içerisinde tanımlı alt roller vardır. Bunlar :

* Ürün Sahibi (product Owner)
* Scrum Master (scrum master)
* Takım ( scrum team )

Şimdi scrum ritüelleri ve rollari biraz daha açalım (Şekil 2.4).



Şekil 2.4. Scrum ritüelleri.

### 2.1.1. Scrum Ritüelleri

#### 2.1.1.1. Geliştirme Planlama (Sprint Planning)

Her bir geliştirme zamnından önce o geliştirme zamanında yapılacak olan yazılım geliştirmelerinin planlandığı ve yapılacak olan işlerin takım tarafından kabul edildiği toplantıdır. Bu toplantıya Ürün Sahibi (PO), Scrum Master ve tüm takım katılır.

Bu toplantı her geliştirme zamanında (sprint) 1 kere yapılır. Bu toplantı esnasında Ürün sahibi (PO) yapılacak işlerin sırasını belirler ve takım bu sıraya bağlı olarak yapabilecekleri işleri alırlar.

#### 2.1.1.2. Günlük Toplantı (Daily Stand-up)

Bu toplantı hergün mevcut durumun kontrolü ve bilgilendirilmesi için yapılır. Bu toplantıya tüm takım, Ürün Sahibi (PO) ve Scrum Master katılır. Bu toplantını özelliği 15 dakikayı aşmamak kaydı ile herkesin sırasıyla

* Dün ne yaptım
* Bugün ne yapacağım
* Beni engelleyen birşey var mı

sorularına cevap vermesinin ardından sonlanmasıdır. Bu toplantının uzamaması için ayakta yapılması tavsiye edilir nitekim adıda burdan gelmektedir.

#### 2.1.1.3. Geliştirme Ön Sunumu (Sprint Review)

Geliştirme zamanın sonunda yapılan bu toplantıda Scrum Master, Ürün sahibi (PO) ve takım bulunur. Bu toplantının amacı mevcut geliştirme zamanndaki işlerin sunumunum yapılması ve PO ve diğer katılımcılardan istenilen özelliğin istenilen şekilde geliştirilmiş olduğunun onayının alınmasıdır. Bu toplantı her gelştirme zamanında bir kere yapılır.

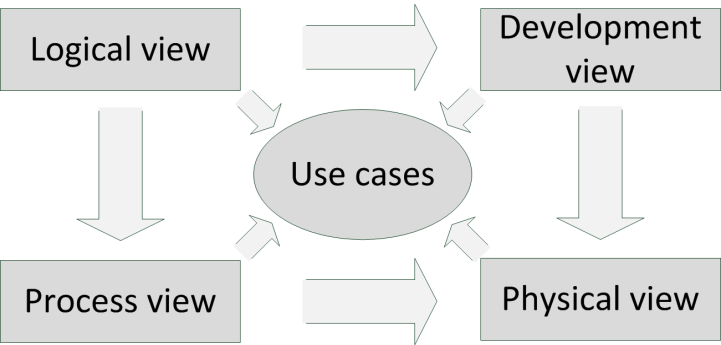
#### 2.1.1.4. Geliştirme Geçmiş Değerlendirmesi (Sprint Retrospective)

Çevik yazılım gelişitirme sürecinin bir özelliği olan, ürün ve geliştirme kültürünü geliştirecek olan hızlı geri bildirimlerin yapıldğı toplantıdır. Ürün sahibi opsiyonel olarak bu toplantıya katılır. Bunun dışındaki scrum Master ve diğer takım üyeleri bu toplantıya katılmak zorundadır. Bu toplantıda önce herkesin geri bildirim vermek istediği konular listelenir ve daha sonra toplantıya katılan her katılımcı bu maddeler üzerinden önceden belirlenen sayı kadar miktardaki geri bildirime puan verir. Tüm puanlama bittiğinde toplantıya katılan herkes bu maddeler üzerinde eğer belirlenebiliyor ise bir ölçü belirleyip, bu maddenin bir sonraki toplantıda takip edilip edilmeyeceğine karar verir. Bu toplantı her geliştirme zamanının (sprint) sonunda yapılır.

# BÖLÜM 3

# YAZILIM MİMARİSİ

Hayatımızın her alanında yazılıma bağımlılığımız artmaktadır. Buda yazılım üreticileri için bir fırsat ve aynı zamanda ürettikleri yazılımın artan ihtiyaçları çabuk ve doğru biçinde cevap verebilecek şekilde dizayn edilmesi konusunu zorunluluk haline getirmiştir. Yazılımda kalite konusu her ürün için aynı kritelere bağlı olmayıp, yazılımın uygulama alanı ve konusu ile bağlantılıdır. Doğru yazılım ürünü için gereksinimlerin doğru belirlenmiş ve planlı olması gerekmektedir(Şekil 6). Fakat günümüz şartlarında baştan tüm gereksinimleri elde etmek zordur ve gereksinimler çok kısa zaman içerisinde değişmektedir. Tüm bunlara rağmen eldeki gereksinimler ve bunların önemlendirilmesi sonucu temel kalite kriterleri oluşturulabilir ve yazılımın tasarımı gerçekleştirlebilir. Yazılımda mimari tasarımı yüksek seviye bir tanımlamadan uygulamayı oluşturacak daha küçük bileşenlere ayırma ve bu bileşenlerin arasındaki ilişkileri belirleme ve bunların mümkünse hali hazırda sektörde kabul görmüş tekniklerle dizayn edilmesidir. Bu dizayn geliştirme esnasında esas alınacak ve geliştirme bu tasarıma uygun şeklide yapılacaktır.

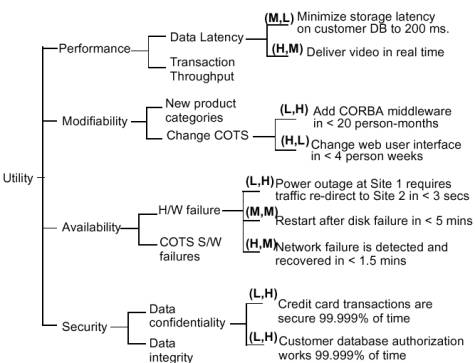


Şekil 3.1.5 4 + 1 mimari tasarım.



Şekil 3.2.6 ISO 25010.

Yazılımın tasarımın yapabilmek için öncelikle gereksinimlerden fonksiyonel ve fonksiyonel olmayan gereksinimsel çıkarılır. Fonsiyonel gereksinimler kodlama ile sağlanacağı için tasarım aşamasında kullanılamaz. Fonksiyonel olmayan gereksinimler tasarımın yapılabilmesi hazırlanmalı, önceliklendirilmeli ve önemlendirilmelidir. Daha sonra bunlar ISO 25010 (Şekil 3.2) kriterlerince belirlenmiş olan kalite niceliklerindeki karşılıklarına göre gruplanıp duruma göre her gruptan önceliği en fazla olanlar göz önüne alınarak bunlaradan bir kalite ağacı oluşturulur. Oluşturulan kalite ağacı üzerinde ürünün başka kalite nitelikleri ile nasıl etkileştiği yada tasarım üzerinde bu kalite kriteri ile yetersiz alanlar olup olmadığı kontrol edilebilir.



Şekil 3.3.7 Kalite ağacı örneği.

Yazılımın mimari tasarımı FR(functional requirement), NFR (non-functional requirement), Architectural Specification, Design decisions, Design Views gibi bir çok dökümanın oluşturulması gerekmektedir. Tüm bu dökümanlar eşliğinde bir sistemin tasarımın erken zamanlı değerlendirmesi yapılabilir.

3.1. YAZILIM MİMARİSİ DEĞERLENDİRME

Mimari değerlendirme yazılım geliştirme aşamasının çeşitli aşamalarında gerçekleştirlebilir. Erken zamanda yapılan değerlendirmelere farklı mimari yaklaşımların güçlü ve zayıf yanlarını erken aşamalarda karşılaştıma imkanı sunar. Daha sonraki aşamalarda yapılan değerlendirmeler sistemin bakımı ve mimarideki erozyonların farkedilmesine yardımcı olur [1].

Yazılım mimarisi değerlendirme metodları 4 gruba ayrılabilir :

##### Uzmanlık Tabanlı (Experience Based)

Bu tip değerlendirmelerde değerlendirme ekibinde daha önceden bu konuda tecrübe sahibi yazılımcılar yada domain ile ilgili uzman kişiler yer alır. Mevcut tecrübeleri ile ilgili mimari yapının gereksinimlere uygun olup olmadığı konusunda geri bildirimde bulunabilir ve öneriler yapabilir [1].

##### Simulasyon Tabanlı (Simulation-based)

Bu tür değerlendirmelerde sistemin yüksek seviye mimari tanımı üzerinden bir yada birkaç bileşenin, performance ve doğruluk gibi kalite niteliklerine uygun olup olmadığı simulasyon üzerinden değerlendirilir [1]. Simülasyon işlemi prototypelar üzerinde de yapılabilir. Bu metodlara örnek olarak Layered queuing Network (LQN) yaklaşımları ve olay bazlı metodlar (RAPIDE) verilebilir.

##### Matematiksel Modelleme (Mathematical Modelling)

Bu yöntemde matematiksel ispatlar ve modeller mimari tasarımın performance ve güvenilirliğini değerlendirmek için kullanılır. Matematiksel modeller simüasyon bazlı metodlarla kullanılarak mimari tasrımın performasını değerlendirmek amacı ile de kullanılabilir.

##### Senaryo Bazlı (Scenario-based)

Senaryo baslı değerlendirme metodunda belli kalite niteliklerinden çıkarılan senaryo profillerinin somut tanımlamalarından faydalanılır. Oluşturulan profil üzerinde oluşabilecek durumların adım adım analizi yapılır. Birçok senaryo bazlı değerlendirme metodu mevcuttur örneğin : Software Architecture analysis Method (SAAM), Application Tradeoff Analysis Method (ATAM), Architecture Level Modifiability Analysis (ALMA)[1].

Bu çalışmada karşılaştırmalarda genel olarak senaryo tabalı değerlendirme metodları kullanılacaltır.

# BÖLÜM 4

# SONUÇ VE DEĞERLENDİRME

Yukarıdaki tanımlamalarda da görüldüğü üzere klasik mimari değerlendirme metodları, gerek döküman yükü gerekse değerlendirme yapacak kişilerin zaman ve para maliyetlerinden dolayı, kısa sürede yalın bir yaklaşım uygulama geliştirme üzerine kurulu çevik yazılım gelştirme yapılarına uygun değil.

Bu konuda ki iyileştirmeler iki alanda yapılabilir.Birincisi her iki method da ağır döküman yükü getirdiği için çevik projelerde daha hafif içerikli ve mimari tasarımı ve kalite özelliklerini birkaç sayfada anlatabilecek bir döküman formatına geçilmesi, ikinci olarak her sprint sonunda takımın mevcut sprint içrisinde mimariye dokunan yada tehlike oluşturan bir kısım olup olmadığı konusunda bir saatlik bir toplantı ve tutanak tutulması.

Doküman konusunda Soyut Mimari Tanımlama Döküman [6] taslağı mevcut olup kullanılabilir haldedir.

Sprint içerisindeki toplantı, sprint gözden geçirme toplantısından sonra yapılabilir. Bu toplantıda geçmiş sprintteki mimari değişilşiler ve etkiler ile gelecek sprintteki muhtemel etkiler üzerine geri bildirimler sağlanıp gerekli değişilikleri yapmak için ajanda oluşturulabilir.

# KAYNAKLAR

1. Graham, R., McCabe, H. and Sheridan, S., “Pathfinding in computer games”, ***Institute of Technology Blanchardstown Journal***, Dublin, Ireland, 8: 56-80 (2003).
2. Kamphuis, A. and Overmars, M. H., “High quality navigation in computer games”, ***Science of Computer Programming***, Utrecht, The Netherlands, 67: 91–104 (2007).
3. Bakkes, C. J., Spronck, H. M. and van Lankveld, G., “Player behavioural modelling for video games”, ***Entertainment Computing***, Tilburg, The Netherlands, 3 (3): 71-79 (2012).
4. Wang, J. Y., “An effective method of pathfinding in a car racing game”, ***The 2nd International Conference on Computer and Automation Engineering (ICCAE)***, Tauyuan, Taiwan, 3: 13-18 (2010).
5. Waveren, J. M. P., “The quake III arena bot”, M. Sc. Thesis, ***Delft University of Technology***, Delft, Netherlands, 10-15 (2001).
6. İnternet: Grenier, M., “Pathfinding 103”, **http://mgrenier.me/2011/04/pathfin ding-103/** (2012).

# ÖZGEÇMİŞ

Fatih ÇETİN 1978 yılında İZMİR’de doğdu; ilk ve orta öğrenimini BALIKESİR’de tamamladı. 2002 yılında Eskişehir Osmangazi Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü’nde öğrenime başlayıp 2008 yılında mezun oldu. Özel sektröde ise çalışma hayatına ise ARGE mühendisi olarak devam etmektedir.

**ADRES BİLGİLERİ**

Adres : 100. Yıl Mah. Prof. Dr. Erdal İNÖNÜ Cad. Narlıken

Sitesi A Blok Daire : 3 Nilüfer / BURSA

Tel : (533) 3731480

E-posta : fatihcetin99@hotmail.com