İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK KRİTERLİ KARAR VERME YÖNTEMİ İLE YAZILIM GELİŞTİRME METODOLOJİSİ SEÇİMİ

YÜKSEK LİSANS TEZİ

Furkan ANARAL

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

OCAK 2012

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK KRİTERLİ KARAR VERME YÖNTEMİ İLE YAZILIM GELİŞTİRME METODOLOJİSİ SEÇİMİ

YÜKSEK LİSANS TEZİ

Furkan ANARAL
(507081112)

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Tufan Vehbi KOÇ

16 ARALIK 2011

İTÜ, Fen Bilimleri Enstitüsü'nün 507081112 numaralı Yüksek Lisans Öğrencisi **Furkan ANARAL**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"ÇOK KRİTERLİ KARAR VERME YÖNTEMİ İLE YAZILIM GELİŞTİRME METODOLOJİSİ SEÇİMİ"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Tufan Vehbi KOÇ** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Cengiz KAHRAMAN** İstanbul Teknik Üniversitesi	.....
	Doç. Dr. Ferhan ÇEBİ İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **16 Aralık 2011**
Savunma Tarihi : **24 Ocak 2012**

Aileme,

ÖNSÖZ

Günümüzde teknolojik gelişmelerin hızlanmasında ve geliştirilmesinde en büyük rol yazılımdadır. Yazılım ise son dönemde işlerin daha da sistematik hale gelmesini sağlayan yazılım geliştirme metodolojilerine ihtiyaç duymaktadır. Bu metodolojilerin en çok tercih edilenleri scrum ve şelale (waterfall) metodolojileridir. Bilişim sektöründeki şirketlerin karlılıklarını artırmak ve daha verimli çalışmak için bu metodolojilerden birini benimsemeleri ve kullanmaları gerekmektedir. Alternatifler arasından seçim yapmak insanların olduğu gibi kurumların da en zorlandığı eylemlerden birisidir. Karar verme işini yapmak için sık kullanılan yöntemlerden biri de Analitik Hiyerarşi Prosesidir. Bu tez çalışmasında bilişim sektöründe sıklıkla kullanılan iki yazılım metodolojisi, scrum ve şelale metodolojileri karşılaştırılacaktır. AHP yöntemiyle belirlenen kriterlerin ağırlıklandırmaları ve önceliklendirmesi yapılacak ve sonrasında alternatifler karşılaştırılacaktır. Sonuçlara ulaşılıp konu ile ilgili değerlendirmeler ve öneriler sunulacaktır.

Tez konusunun belirlenmesi, tezin oluşumu ve şekillenmesindeki yorumlarıyla yol gösteren danışmanım Prof. Dr. Tufan Vehbi KOÇ'a, çalışma esnasındaki desteklerinden dolayı çalıştığım şirketteki Uygulama Geliştirme Bölümü çalışanlarına, her zaman yanımda olan ve desteğini, güvenini eksik etmeyen annem Fatma ANARAL, babam Raşit ANARAL ve kardeşim Hilal ANARAL'a teşekkürlerimi sunarım.

Ocak 2012

Furkan ANARAL
(Endüstri Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

IT	**:** Information Technology
AAP	**:** Analitik Ağ Prosesi
TOPSIS	**:** Technic for Order Preference by Similarity to Ideal Solution
AHP	**:** Analitik Hiyerarşi Prosesi
CMMI	**:** Capability Maturity Model Integrated
YGYD	**:** Yazılım Geliştirme Yaşam Döngüsü
HUG	**:** Hızlı Uygulama Geliştirme
DSGM	**:** Dinamik Sistem Geliştirme Metodolojisi
ÖGP	**:** Özellik Güdümlü Programlama
UP	**:** Uç Programlama
KDS	**:** Karar Destek Sistemleri
ÇKKV	**:** Çok Kriterli Karar Verme
ÇNKV	**:** Çok Nitelikli Karar Verme
ÇAKV	**:** Çok Amaçlı Karar Verme

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÇOK KRİTERLİ KARAR VERME YÖNTEMİ İLE YAZILIM GELİŞTİRME METODOLOJİSİ SEÇİMİ

ÖZET

Bu çalışmada, öncelikle tezin ana konusu olan yazılım geliştirme metodolojileri ile ilgili teorik bilgiler verilmiştir. Sonrasında yazılım geliştirme metodolojileri arasından seçim yapılmasını sağlayacak karar verme teknikleri ile ilgili genel teorik bilgiler verilmiştir. Teorik kısımda ağırlıklı olarak kıyaslaması yapılacak alternatifler olan scrum ve waterfall metodolojileri üzerinde durulmuştur.

Tez çalışmasında genel olarak yazılım geliştirme metodolojileri anlatılmıştır. Yazılım geliştirme metodolojileri anlatıldıktan sonra tezin amacını oluşturan problem tanımı yapılmıştır. Bu problemin çözümü için çok kriterli karar verme yöntemleri kullanılacaktır. Bu nedenle, çok kriterli karar verme yöntemlerinden ayrıntılı şekilde bahsedilmiş ve karşılaştırma kriterlerinin tespiti, ağırlıklandırması ve önceliklendirmesinde kullanılacak olan Analitik Hiyararşi Prosesi yöntemi anlatılmıştır.

Teorik bilgilerden sonra, tezin uygulama bölümünde bilişim sektöründe kullanılan scrum ve şelale (waterfall) metodolojilerinin Analitik Hiyerarşi Prosesi yöntemi kullanılarak ağırlıkları belirlenmiş, önceliklendirmeler yapılmış ve alternatiflerden birinin seçilmesi amaçlanmıştır. Kriterlerin belirlenmesinde 2 adet anket çalışması yapılmıştır. Anketler yardımıyla bu karşılaştırmada kullanılacak faktörler ve alt faktörler belirlenmiştir. Anket çalışmasına, çalıştıkları şirketin uygulama geliştirme bölümünde çalışan ve her iki metodolojide de tecrübe sahibi kişiler katılmıştır. Sonuçta, 61 faktör belirlenmiştir. Daha sonra, bu faktörler 4 ana faktör ve 57 alt faktör olarak ayrılarak model oluşturulmuştur. Konusunda uzman 5 kişinin ortak görüşüyle, oluşturulan modeldeki faktörler ikili karşılaştırma metodu ile ağırlıklandırılmıştır. Bu ağırlıkların tutarsızlık oranları da incelenerek modelin faktör ağırlıkları belirlenmiştir. Sonraki aşamada, Scrum ve waterfall metodolojileri 1-5 skalasında uzman 5 kişi ile yapılan toplantılar sonucunda puanlandırılmıştır. Analitik Hiyerarşi Prosesi modelinin analizi ve faktörlerin karşılaştırılması için Super Decision paket programı kullanılmıştır. Programda ortaya çıkan raporlar ve sonuçlar analiz edilerek değerlendirilmiş ve öneriler sunulmuştur. Super decision programında model oluşturulduktan ve öncelikler belirlendikten sonra model excele aktarılmış ve yapılan puanlamalar sonucunda excelde gerekli hesaplamalar yapılarak alternatiflerin kıyaslamaları yapılmıştır. Ortaya çıkan bu sonuçlar üzerinden kurulan model ve karşılaştırması yapılan yazılım geliştirme metodolojileri ile ilgili analizler ve değerlendirmeler yapılmıştır. Metodolojilerin daha iyi hale getirilmesi için önerilerde bulunulmuştur. Bu şekilde tez çalışması sonlandırılmıştır.

SOFTWARE DEVELOPMENT METHODOLOGIES SELECTION WITH MULTI CRITERIA DECISION MAKING

SUMMARY

In this study, primarily theorical informations are given about software development methodologies that is the main subject in this thesis. After that, general theorical informations are given about decision-making techniques that will be used in the selection of software development methodologies. In theorical section, scrum and waterfall software methodologies that are the comparison alternatives in this thesis are elaborated mostly.

Generally, software development methodologies are explained in this thesis. Problem description that is generated the objective of thesis is made after the software development methodologies are explained. To solve this problem, the multi criteria decision making will be used. Because of this, the multi criteria decision making is described detailed and also analytical hierarchical process method is described that will use for fixing the comparison criterion, weighting and prioritization.

After the theorical informations, in the application section in this thesis scrum and waterfall methodologies comparison with using analytical hierarchical process method and select one of these alternatives is the main objective. Factors and lower factors in this comparison are determined with the help of surveys. The experts that work at application development section in the firm and have experience in these software methodologies join these surveys. Eventually 4 main factors and 57 lower factors are determined. To analyze weights of factors, pairwise compare method in analytical hierarchical process is used with the help of consensus of five experts. The basic inconsistency ratios are analysed and weighted factors are designed. In the next step, scrum and waterfall methodologies are given points in 1-5 scale. For analysis of analytical hierarchical process model and pairwise comparing of upper and lower factors, Super Decision software program is used. The reports and the results in this program are analysed and considered and new suggestions are offered. After the comparison model is fixed in super decision program, comparison of alternatives is made in excel program.

Software development methodologies include construction, planning and control of an information system's development process. Software development methodologies have an important place in especially information technology industry. Also, it is the most prefered scrum and waterfall methodologies by companies.

Scrum has been improved by Jeff Sutherland and Ken Schwaber during the middle of 1990s. It is a methodology, which is applied and leads to success in the projects that presents a high level of uncertainty.

Waterfall methodology that is mentioned above is the first software development methodology. In 1970s, Winston W. Royce wrote an article that includes the waterfall methodology. It has been used by many companies and corporation. The

information technology industry has improved day by day and it is a sector that companies spend lots of money.

The decision, which is about to choose the most efficient methodology, directly affects the growth rate of the companies. It is necessary that companies accomadate to technological development and work fast and efficiently. At the same time, they have to provide smooth and quality service. Therefore, they can live up to the expectations of users. To do this, they should choose a methodology which is appropriate to their own companies and sector. As a conclusion, the subject of thesis includes the problem is which software development methodology is chosen.

In the thesis study, the multi criteria decision-making methods are used to the comparison of these two methodologies. The theoretical information about Decision-making and multi-criteria decision making methods will be introduced. Analytical Hierarchy Process that is a multi criteria decision-making methods will be used for selection of the software development methodology that is the problem of this thesis.

The reason for choosing the method of Analytical Hierarchy Process is using both objective and subjective evaluation criteria. Another reason is testing consistency of evaluation, in particular through the substance to which a large number of alternatives by the decision maker to apply a very important decision. Users will also have to include information on the Analytic Hierarchy process are evaluated and measured data will be easily applied. In addition, the fact that the information are acquired from users are evaluated and measured data provides the Analytical Hierarchy Process will be easily applied. The other factor about why Analytical Hierarchy Process chosen is the written numerous articles and thesis about this method .This provides to vary from others. The main advantage of the method is understanding is easier than others. Also, it does not include unnecessary mathematical operations and it directs multiple criteria easily. Analytical Hierarchy Process is also explained detailed in this thesis.

In the application of Analytical Hierarchy Process criterias which are used to comparison of methodologies will be determined and also, classification criteria and priority levels will be determined. After establishing comparison model, methodologies will be compared in excel program by using these informations. The application will be done in a company, which is 75 rank in first 500 information technology companies. Also, the company has workers more than 500 and it applies both scrum and waterfall methodologies.

In the part of application, firstly the informations are given about scrum and waterfall from software development methodology. After the general structure, process, rules, roles and important points of scrum and waterfall methodologies are emphasised, applied questionnaire studies will be explained. The determination of factors which will be used for the comparison of these methodologies are enabled with questionnaire studies. After key factors and other factors are determined, by the method of Analytical Hierarchy process the comparison of software development methodology model will be established and it will be transferred to super decision program. levels of priority and weighting factors will be done by five expert person in super decision program. The priorities will be determined by comparing between 1-9 scale factor. After this decision, model home priorities list will be transferred to excel program. Scrum and waterfall methodologies are evaluated between 1-5 scale by five experts' opinion. According to consequences of evaluation, both methodologies will be compared and the analysis of results will determine which

methodology are chosen. After analysing these consequences, application study will be finished by giving suggestion.

Application study was applied in a company which is in information technology industry and it provides software to one of the most important bank of Turkey. Also, this company has ranked among the top 500 Information tchnology companies and studied not only scrum but also waterfall methodologies. The workers of the company support to make the comparison model by using questionnare studies.

In conclusion, necessary model has established for the comparison of software development methodology and in application study, the fact that scrum methodology is more productive than waterfall has reached. At the beginning of study, questionare study has applied and criterias has determined to compare scrum and waterfall methodologies. Afterwards, this criteria has arrenged as of main factors and other factor and the model has established. After the established model has transfered to super decision program, factors between 1-9 scale has compared each other with evaluation from five experienced people. Therefore, the final outcomes have reached by giving point between 1-5 scale on the model of scrum and waterfall methodologies.

The factor of the highest priority level is cost and the second highest priority level factor is quality. Also the factor of the lowest priority level is scope and the second lowest priority level factor is time. Process costs are the highest priority level factor in the subfactors of the cost factor. The position of the team is the lowest priority level factor in the subfactors of the scope factor.

When factors of second level is analysied, testing cost has the highest priority level in process cost. Software cost has the highest priority level in outside source. Test easiness has the highest priority level in software qualit factor and usability has the highest priority level in analysis-design quality. In delivery time factor, testing time has the highest priority level and in time planning factor, high level projects delivery time has the highest priority level. Change density of Project needs has the highest priority level in the external factors of the scope. Aim of the customer demands has the highest priority level in the internal factors of the scope. Also, the lowest priority level in the position of the team is the number of team members.

In conclusion, according to the outcomes of evaluation , scrum methodology has higher ratio than waterfall in many different areas. The ratio in the process cost factor is %77-%51, in the outsource costs %77-%62. In addition, in the software quality factor, the ratio is %83-%73. %96-%63 is the ratio of the analysis-design quality. Delivery time factor has %90-%50 ratio. The ratio of the time planning is %85-%55. External factors of th scope has %85-%58 ratio. The internal factors of the scope has %86-%59. The position of the team has %82-%68 ratio. In the main factors; the ratios are 92%-66% in quality main factor, 89%-51% in time main factor, 85%-60% in scope main factor, 77%-54% in cost main factor. According to the general evaluation, ratio is 85%-58%. Therefore, according to the multi criteria decision making methods, scrum is preferred software methodology.

The firm in the application has a %85 success rate with the scrum methodology. They are not experinced in scrum but they can raise this %85 ratio in time. The firm works with waterfall methodology last ten years and has some habituations becaus of this reason. They get used to scrum methodology. This thesis also shows them and the other firms in the information technology industry their deficiency. Suggestions in the application part will be usefull for the information technology industry.

1. GİRİŞ

Yazılım Geliştirme Metodolojileri bir bilgi sistemini geliştirme sürecinin yapım, planlama ve kontrolünü kapsar. Özellikle bilişim sektöründe çok önemli yere sahip olan yazılım geliştirme metodolojilerinin şirketler tarafından en çok tercih edilenleri scrum ve şelale (waterfall) metodolojileridir. Scrum Jeff Sutherland ve Ken Schwaber tarafından 1990'ların ortalarında geliştirilmiştir. Yüksek seviyede belirsizlik arz eden projelerde son yıllarda yoğun biçimde uygulanan ve başarılı sonuçlar üreten bir tür metodolojidir. Şelale metodolojisi ise ortaya atılan ilk yazılım geliştirme metodolojisidir. 1970'li yıllarda Winston W. Royce tarafından yazılan bir makalede oluşturulan şelale modeli yıllarca birçok firma ve kurum tarafından kullanıldı.

Tez çalışmasında bu iki metodolojinin karşılaştırılmasında kullanılmak üzere çok kriterli karar verme yöntemleri kullanılacaktır. AHP uygulamasında metodolojilerin karşılaştırılmasında kullanılacak kriterler belirlenecek, kriterlerin sınıflandırılması ve ağırlıklandırılmaları yapılarak öncelik seviyeleri belirlenecektir. Karşılaştırma modeli kurularak daha sonra excelde bu bilgiler kullanılarak metodolojilerin karşılaştırılması yapılacaktır. Uygulama bilişim sektöründe ilk 500 sıralamasında 75. Sırada bulunan ve 500'den fazla çalışanı olan hem scrum hem de şelale metodolojisinin kullanıldığı bir bilişim şirketinde yapılacaktır. Çalışmanın sonuçları da analiz edilerek sektörde hangi yazılım metodolojisinin kullanılmasının daha doğru bir seçim olduğunun belirlenmesi ve bu metodolojinin geliştirilmesinde neler yapılabileceğinin analiz edilmesi amaçlanmaktadır.

2. YAZILIM GELİŞTİRME METODOLOJİLERİ

2.1 Tanımı

Yazılım Mühendisliği kapsamındaki Yazılım Geliştirme Metodolojileri, bir bilgi sistemini geliştirme sürecinin yapımını, planlamasını ve kontrolünü sağlayan bir yapıdır. Her farklı iskelet (framework) yapısı güçlü ve zayıf yanlarıyla birlikte zaman içinde olgunlaşır.

Bir sistem geliştirme metodolojisi her proje için uygun olmayabilir. Her metodoloji, projenin teknik, organizasyonel ve takım unsurlarına göre farklı özellik gösterir.

Bir yazılım geliştirme metodolojisi aşağıdaki adımlardan meydana gelir;

- Yazılım geliştirme süreci yaklaşımıyla bir yazılım geliştirme felsefesi
- Yazılım geliştirme sürecine destek verecek araçlar, modeller ve yöntemler.

Bu iskeletler çoğunlukla metodolojiyi geliştiren, destek veren ve onu ilerleten organizasyon tiplerine bağımlıdır [1].

2.2 Tarihçe

En eski yazılım geliştirme araçlarından biri, kökleri 1920'lere dayanan akış şemalarıdır. Yazılım geliştirme metodolojisi 1960'lara kadar tam olarak ortaya çıkmamıştır. Yazılım geliştirme yaşam döngüsü (YGYD), bilgi sistemleri inşası için biçimlendirilmiş en eski metodoloji olarak nitelendirilebilir. YGYD'nin ana fikri bilgi sistemi geliştirme sürecinin planlanmış, yapısal ve metotlu bir şekilde olmasını kovalamaktır. Bu metodolojinin 1960'lardaki temel hedefi geniş ölçekli fonksiyonel business sistemleri, geniş ölçekli büyük şirketlerde geliştirmekti. Bilgi sistemleri aktiviteleri yüklü veri işlemleri ve yoğun hesaplama işlemleri etrafında döner. 1970'lerden itibaren ortaya çıkan yazılım geliştirme metodolojileri şu şekildedir [1].

1970'lerde

- Yapısal programlama

1980’lerde

- Yapısal sistem analizi ve tasarımı metodolojisi

1990’larda

- Nesne yönelimli programlama
- Hızlı Uygulama Geliştirme
- Scrum
- Takım Yazılım Süreci

2000’lerde

- Rasyonel Bütünleştirme Süreci
- Uç Programlama
- Çevik Tümleşik Süreç
- Entegre Metodoloji

2.3 Yazılım Geliştirme Yöntemleri

Her yazılım geliştirme metodolojisinin, yazılım geliştirme açısından az ya da çok kendi yaklaşımları vardır. Birkaç spesifik metodoloji tarafından geliştirilen genel yaklaşımlar şu şekildedir [2].

- Prototipleme: İteratif tip
- Artımlı: Lineer ve iteratif tip
- Hızlı Uygulama Geliştirme: İteratif tip
- Evrimsel Geliştirme
- Yinelemeli ve Artımlı
- Spiral: Lineer ve iteratif tip
- Şelale (Waterfall): Lineer tip
- Çevik Geliştirme (Agile)

2.3.1 Prototipleme

Yazılımda prototipleme, istenilen yazılım kodlanırken, yazılımın tamamlanmamış bir prototipinin oluşturulma sürecindeki aktiviteleri içeren bir iskelet yapısıdır (framework). Şekil 2.1'de gösterilen Prototipleme metodunun temel prensipleri;

• Tek başına bir şey ifade etmeyen, bütünsel bir geliştirme metodolojisidir. Fakat bütünün belirli parçalarını ele alma yaklaşımının aksine daha geleneksel bir geliştirme metodolojisidir.

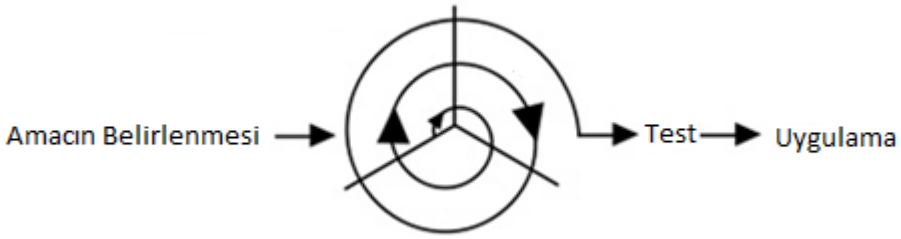
• Projeyi küçük segmentlere ayırarak ve geliştirme sürecinde projenin daha kolay değiştirilebilir olmasını sağlayarak öngörülen proje risklerini azaltmaya çalışır.

• Kullanıcılar sürece dâhil edilmişlerdir. Bu sayede nihai implementasyonda kullanıcıların onay verme oranı/olasılığı artmış olur.

• Prototip, kullanıcı gereksinimlerini karşılayana kadar olan süreçte iteratif şekilde sistemin küçük ölçekli modelleri geliştirilir.

• Reddedileceği düşüncesiyle geliştirilen prototiplerin bazıları prototip halinden gerçek çalışan sistem haline doğru evrimleşebilir.

• Gerçek problemi anlamak, yanlış problemi çözmekten kaçınmak için gereklidir [2].



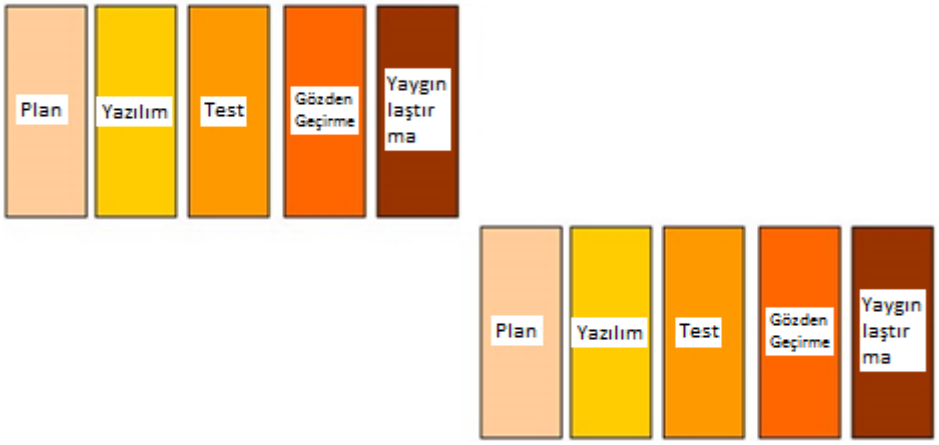
Şekil 2.1 : Prototipleme.

2.3.2 Artımlı yazılım geliştirme

Lineer ve iteratif sistem geliştirme metodolojilerini içermesi açısından muhtelif metotlar kullanılabilir. Projeyi küçük segmentlere ayırarak ve bu sayede kodlama sürecinde kolay değiştirilebilir olmasını sağlayarak projenin olası risklerinin azaltılması hedeflenir. Şekil 2.2'de gösterilen metodunun temel prensipleri;

• Şelale modelinin küçük parçalar halinde dizisi gibidir. Burada şelale geliştirme modelinin tüm fazları sistemin küçük bir parçası için tamamlanmış durumdadır.

• Şelale yaklaşımında ilk yazılımsal konsept, gereksinim analizi ve sistem mimarisi tasarımı belirlenir. Sonrasında da finalize edilmiş prototipin sisteme yüklenmesi ile sonuçlanacak olan prototipleme süreci ile devam edilir [2].



Şekil 2.2 : Artımlı yazılım geliştirme.

2.3.3 Hızlı uygulama geliştirme

Hızlı Uygulama Geliştirme (HUG), çevik prototipleme adına minimal planlamanın kullanıldığı bir yazılım geliştirme metodolojisidir. HUG kullanımında planlama, kod yazıldığı zaman aralıklarla yapılır. Kapsamlı bir ön planlamanın olmaması genellikle yazılımın daha hızlı yapılmasına ve gereksinimlerin daha kolay değiştirilebilir olmasına olanak tanır.

HUG'da yapısal teknikler ve prototiplemeler kullanıcı gereksinimlerini ve sonuç ürünü tasarlamak için kullanılır. Geliştirme süreci, yapısal teknikler kullanılarak öncelikli veri modellerinin ve iş süreç modelinin geliştirilmesiyle başlar. Bir sonraki

aşamada gereksinimler prototipleme çalışması ile doğrulanır. Bu süreçler iteratif biçimde ilerler, sonrasında geliştirme süreci yeni sistemin inşası için kullanılan birleştirilmiş iş gereksinimleri ve teknik tasarımları ile devam eder.

HUG, Çevik olmayan metotlara cevap olarak ortaya çıkmıştır. Diğer metotlardaki öncelikli sorun, sistem tamamlanmadan önce değişen gereksinimlerin uygulanmasının çok uzun zaman alması ve yetersiz/kullanışsız sistemlerin ortaya çıkmasıydı. Bir diğer problem de sistemli bir gereksinim analizi fazının tüm kritik gereksinimleri tek başına adresleyebileceği varsayımıydı.

Genellikle James Martin tarafından 1991'de ortaya atılan yazılım geliştirme sürecini tarif etmek için de HUG terimi kullanılır. Temel prensipleri;

• Anahtar hedef, az bir yatırım maliyeti ile hızlı kod geliştirerek yüksek kalitede sistemler üretmektir.

• Projeyi küçük segmentlere ayırıp geliştirme sürecince kolay değiştirilebilir olması sağlanarak proje riskinin azaltılmasına çalışılır.

• Hızlı bir şekilde yüksek kaliteli sistemleri iteratif prototipleme metodolojisini projenin herhangi bir safhasında kullanarak, kullanıcıları aktif şekilde dahil ederek ve otomatize edilmiş geliştirme araçları kullanarak sunmaya çalışır. Bu araçlar Grafik Kullanıcı Arayüzü geliştiricileri, Bilgisayar Destekli Yazılım Mühendisliği araçları, Veritabanı Yönetim Sistemleri, 4. Nesil programlama dilleri ve nesne yönelimli teknikleri kapsar.

• Anahtar vurgu, ihtiyaçları karşılamak üzerinedir. Bu durumda teknolojik ve teknik başarı daha az önem arz eder.

• Proje kontrolü, kod geliştirme ve teslim süresi tanımlama konularında önceliklendirme yapılmasını sağlar. Proje uzamaya başladıysa, yaklaşım gereksinimleri teslim tarihine göre revize etmek olmalı, teslim süresini geciktirecek şekilde değil.

• Genel olarak Bileşik Uygulama Geliştirme'yi kapsar. Kullanıcılar hem yapısal çalıştaylar hem de kolaylaştırılmış elektronik etkileşimler vasıtasıyla mutabakata varma çalışmalarında sistem tasarımına yoğun şekilde dâhil edilmiştir.

• Kullanıcıların aktif şekilde dâhil olması kaçınılmazdır.

• Prototiplerin atıl olmasına karşın yazılımsal ürün iteratif şekilde üretilir.

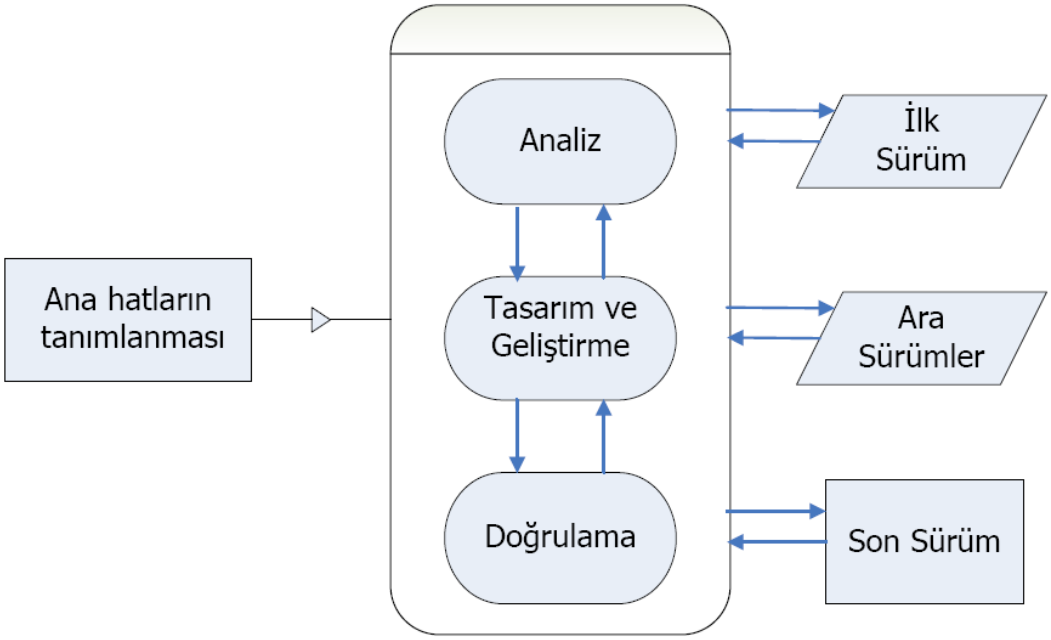
• İleride yapılacak yazılım ve destek faaliyetlerine kolaylık sağlayacak dokümanlar üretilir.

• Standart sistem analizi ve tasarımı teknikleri bu iskelet yapıya uydurulabilir.

2.3.4 Evrimsel geliştirme

Evrimsel geliştirme ise tanımlama, analiz, tasarım, geliştirme ve doğrulama faaliyetlerinin ilk olarak hızlı bir şekilde yapılması, doğrulama sağlanana kadar aşamaların tekrarlanması mantığını temel alan gelişmiş bir modeldir. Süreç şekil

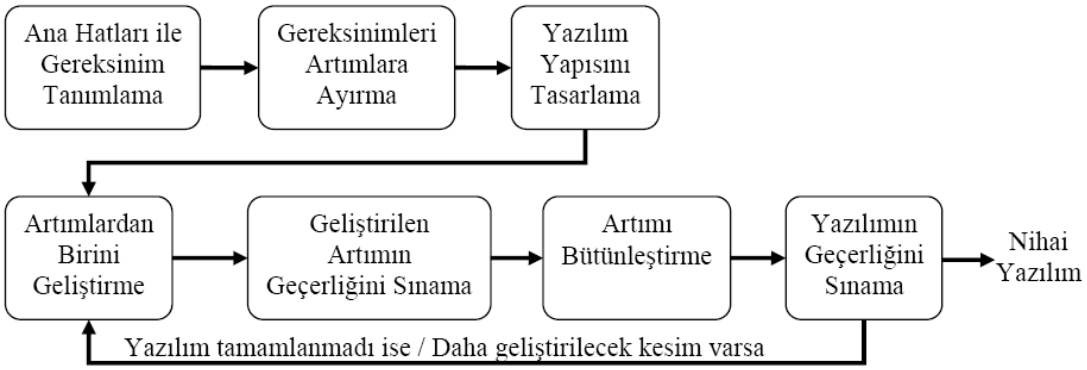
2.3'te gösterilmiştir. Karmaşık bir yönetim mekanizması gerektirse de, değişiklikleri çok daha az maliyetle karşılayabilmesi güçlü bir özelliğidir.



Şekil 2.3 : Evrimsel geliştirme.

2.3.5 Yinelemeli ve artımlı

Şelale modelindeki eksikliklerden yola çıkılarak geliştirilmiştir ve yazılımın geliştirilmesi sırasında bir tekrar ile yazılımın daha iyi hale getirilmesi hedeflenmiştir. Yazılım kalitesini arttıran, erken geribildirim sayesinde riskin azaltılmasını sağlayan, yeni ve değişen gereksinimleri gerçekleştirmek için esnek bir yapı sunan bir metottur. Şekil 2.4'te adımları gösterilmiştir.



Şekil 2.4 : Yinelemeli ve artımlı.

2.3.6 Spiral

Spiral modeli ilk olarak 1988 yılında Barry Boehm tarafından tanımlanmıştır. Spiral modeli, Şelale modelinden farklı olarak belli ve düz bir akıştan ziyade iteratif bir yazılım modelidir. Tabi ki Spiral ilk iteratif model değildir; ancak neden iteratif bir süreç yürütüldüğünü açıkça gösteren ilk modeldir. Spiral modelinde iteratiflikten

kasıt ise, her bir iterasyon bir tasarım hedefi ile başlar, standart yazılım süreçlerinin bazı aşamalarını geçip müşteri yorumu ve görüşü ile tamamlanır. Her bir iterasyonda amaç iş değerini yakalamak olup, bir önceki iterasyonun sonuçlarından faydalanarak gelişen bir yapıda devam eder [2].

Temel prensipler;

• Risk değerlendirmesi, projeyi küçük parçalara ayırarak riskin minimize edilmesi, geliştirme süresinde kolay değiştirilebilir olmasının sağlanması gibi noktalara odaklanılmıştır.

• Proje yaşam döngüsündeki her aşama, proje kapsamında hazırlanan genel bir operasyonel kavram dokümanından her uygulamanın kendi kodlarını içeren spesifik dokümanına kadarki her aşama ve ürünün ve detaylarının her bir adımı boyunca kendi içinde bir ilerleme sağlar.

• Her döngüye paydaşların tespiti ile başlanır ve her döngüyü son bir kez gözden geçirerek ve taahhütle sonlandırılır.

Basitçe Spiral modelinin aşamaları aşağıdaki gibidir.

1. Gereksinim analizi: Bu süreç müşteriden ihtiyaçların alınıp analizin yapılması sürecidir.

2. Taslak Tasarım: Bu süreçte alınan analize göre taslak tasarım yapılır.

3. İlk prototip: Oluşturulan taslak tasarıma göre yazılımın ilk prototipi oluşturulur. Tabi bu prototip sadece müşteriye yazılım karakteriği ve çözüme yaklaşımı konusunda fikir vermek için geliştirilen basit bir prototiptir.

4. İkinci prototip: İkinci prototip aşağıdaki dört adımın sonunda oluşturulur.

a) İlk prototipin güçlü ve zayıf yanları ve risklerinin değerlendirilmesi

b) İkinci prototip için tekrar gereksinim analizinin yürütülmesi

c) Tekrarlanan gereksinim analizi sonucuna göre tekrar teknik tasarım yapılması

d) Oluşan teknik tasarıma göre yazılımın geliştirilmesi ve test edilmesi

e) Müşteri ile oluşan prototipin değerlendirilmesi. Bu aşamada eğer müşteri gidişattan memnun değilse daha en başta proje iptal edilebilir, böylece boşa gitme ihtimali olan birçok maliyetten daha başta kurtulunur. Şayet müşteri gidişattan memnunsa sürece devam edilir.

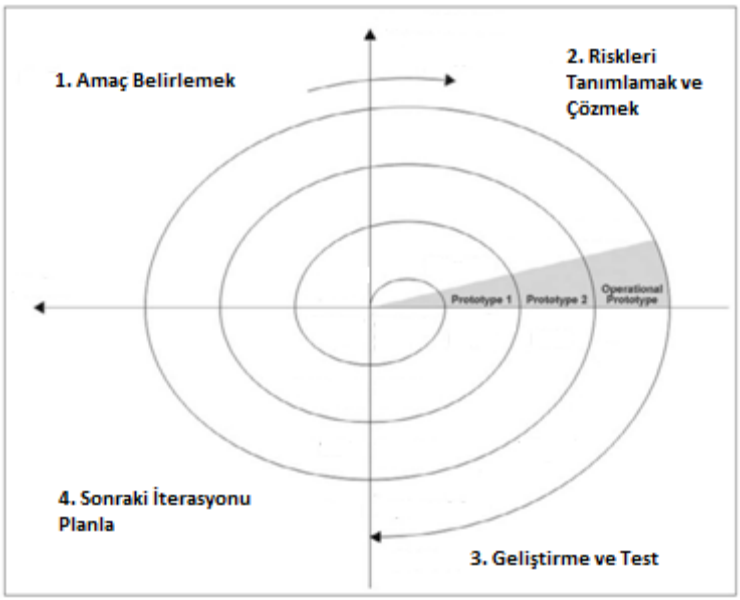
5. n. Prototip: Spiral modelde süreç bu şekilde iteratif adımlarla devam eder. Her bir adımda yukarıdaki beş adım uygulanır.

6. Her prototip sonunda yapılan müşteri değerlendirmeleri, müşteri oluşan yapıdan memnun olana kadar devam eder. Ne zaman müşteri ihtiyaçlarının karşılandığını düşünür ve sonuçtan tatmin olursa nihai ürün için çalışılmaya başlanır.

7. Nihai ürün, onaylanan son prototip üzerinde şekillenir ve canlı hayata alınır.

Görüldüğü gibi Spiral modelde açık bir değişiklik yönetimi, kapsamlı bir test süreci yoktur. Çünkü her iterasyonda müşteriden alınan analiz ile değişiklik istekleri toplanmış olur ve tekrar tekrar test edilmiş olur.

Spiral model, küçük kapsamlı projeler için uygun değildir ve büyük projelerde kullanılabilir. Şekil 2.5'te gösterilmiştir.



Şekil 2.5 : Spiral.

2.3.7 Şelale (Waterfall)

Şelale modeli şekil 2.6'da görüldüğü şekilde 7 ana süreçten oluşur.

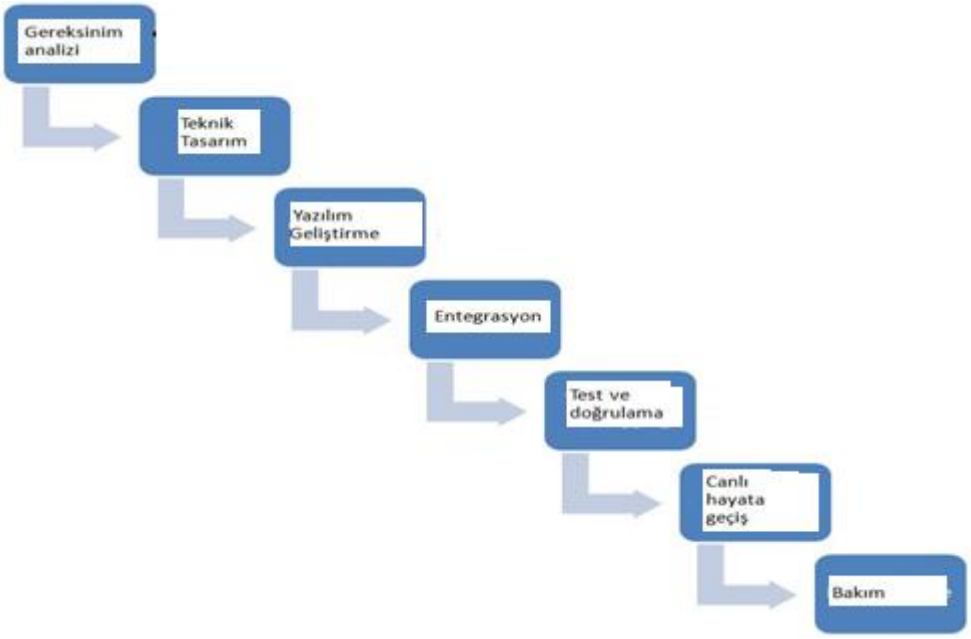
Royce yazdığı makalesinde 7 süreçten bahsediyordu,

- Gereksinim analizi
- Teknik Tasarım
- Yazılım Geliştirme
- Entegrasyon

• Test ve doğrulama

• Canlı hayata geçiş

• Bakım

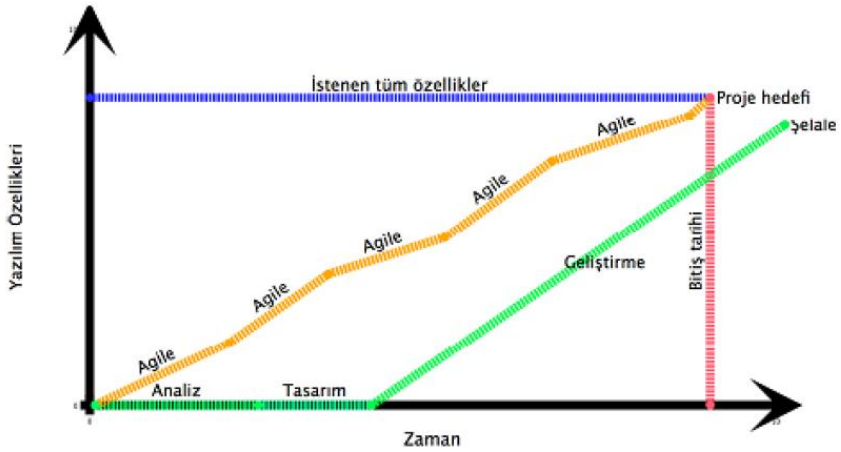


Şekil 2.6 : Şelale (Waterfall) metodolojisi.

2.3.8 Çevik geliştirme

2.3.8.1 Çevik geliştirme tanımı

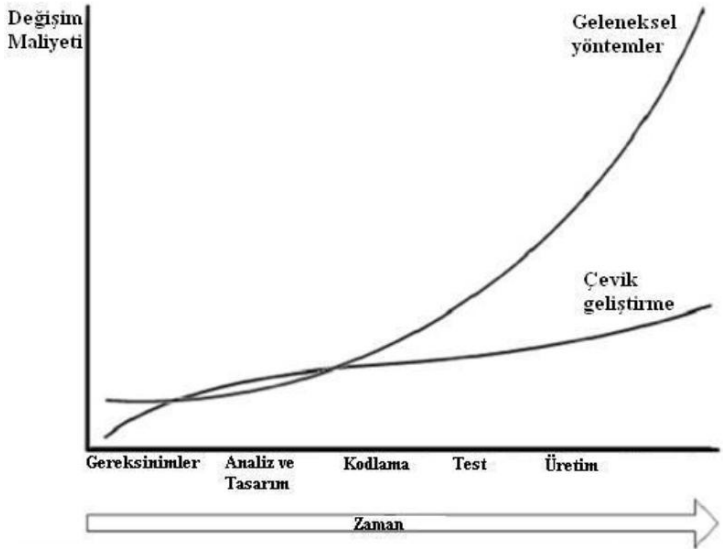
Çevik kelime olarak hızlı ve atik düşünmek, akıllı bir yolu tercih etmek demektir. Yazılım geliştirmede çevik kelimesi ise değişen ihtiyaçlara hızlı cevap veren pratikleri ifade eder. "Çevik", dünyada yazılım süreçlerini daha esnek ve güçlü kılmak için kullanılan aynı zamanda yazılım süreçlerini de kısaltan kavramsal bir yazılım geliştirme metodolojisidir [3]. Çevik geliştirmenin şelale sürecinden farkı şekil 2.7'de, değişim maliyeti ise şekil 2.8'de gösterilmiştir.



Şekil 2.7 : Çevik geliştirmenin şelale sürecinden farkı.

Çevik yazılım süreçleri, 1950'lerdeki üretim alanında verimliliğin artırılması için geliştirilen yalın yaklaşımların yazılım sektöründe bir uzantısı olarak ortaya çıkmıştır. Yazılım dünyasında çeşitli çevik yaklaşımlara 1970'lerden itibaren rastlanabilmekle birlikte, çevik yazılım geliştirme yöntemlerinin kullanımı 1990'larda hız kazanmış ve geçtiğimiz son 7–8 yıl içerisinde de tüm dünyada başarılarını kanıtlayarak popülaritesini arttırmıştır. Şu anda, dünyadaki birçok yazılım şirketinde ve birçok yazılım projesinde yazılımlar, çevik yaklaşımlarla geliştirilmektedir. Çevik yaklaşım, özellikle şu an bilişim sektörünün en güçlü olduğu Amerika ve İngiltere gibi ülkelerde altın zamanını yaşıyor. Çok büyük şirketler bile çevik yazılım geliştirme yöntemlerinden yarar sağlayarak çalışmalarını yürütüyorlar. şirketlerin ilgisindeki artışın en büyük nedeni artan rekabet koşulları, şirketlerin IT projelerine yaptıkları yatırımların sonucu daha hızlı görmek istemeleri, çevik çözümlerin bu ihtiyaçlara cevap verebilmesi ve müşteriye katma değer sağlamaya odaklı olmasından kaynaklanıyor.

Bilindiği gibi geleneksel yazılım geliştirme süreçleri, projelerde değişen gereksinimleri karşılamak ve beklenen kalite faktörlerine erişmek açısından çok katı ve ağırdır. Mesela Şelale modelinde hedefe doğru ilerlemeden önce uzun ve detaylı analiz, tasarım, gözden geçirmeler, onay süreçleri bulunur. Proje başlangıcından bir süre sonra çalışan yazılım özellikleri geliştirilmeye ve hedefe doğru ilerlemeye başlanır. Bu modelde proje başlangıcında detaylı çalışmalarda çok zaman kaybedilir ve hedefe ulaşılamayacağı projenin ancak sonuna doğru belli olur. Detaylı çalışmalar sonunda oluşturulan yapıda ilerleyen aşamalarda değişikliğe gidilmesi büyük maliyetler oluşturur. Çevik yazılım geliştirme ise süreçlerin katı kurallarla yönetilmesi yerine amaca yönelik verimli ve etkin pratiklerin yapılmasını esas alır.



Şekil 2.8 : Çevik geliştirmenin değişim maliyeti.

2.3.8.2 Çevik manifesto

Kent Beck ve dünyanın önde gelen çevik yazılım geliştiricisi 16 arkadaşı ortak bir zeminde buluşabilmek adına 13 şubat 2001'de Utah'da bir araya gelerek Şekil 2.9'da gösterilen çevik yazılım geliştirme manifestosunu yayınlamışlardır.

Çevik Yazılım Geliştirme Manifestosu

Yazılım geliştirmenin daha iyi yollarını yazılımı geliştirerek ve diğerlerinin yazılım geliştirmesine yardımcı olarak ortaya çıkartıyoruz. Bu çalışmamız sırasında aşağıdaki değerlere ulaştık:

- Bireyler ve aralarındaki etkileşimlerin, kullanılan araç ve süreçlerden;
- Çalışan yazılımın, detaylı dokümantasyondan;
- Müşteri ile işbirliğinin, sözleşmedeki kesin kurallardan;
- Değişikliklere uyum sağlayabilmenin, mevcut planı takip etmekten;
- Sağdaki konularda değer olmakla beraber; soldaki konulara daha fazla değer veriyoruz.

Şekil 2.9 : Çevik yazılım geliştirme manifestosu [4].

Çevik manifesto ile çevik sürecin bir değer sistemine sahip olması ve geleneksel yazılım metotlarından elde edilen tecrübeler doğrultusunda daha pratik ve çevik bir yapıda olması gerektiği dile getirilmiştir. Öncelikler şöyle sıralanmıştır.

• Kişiler ve iletişim, süreç ve araçlardan önce gelir.

İnsan başarının en önemli anahtarıdır. Güçlü oyunculardan oluşmayan takımların süreçlerin başarılı olması mümkün değildir. Aynı şekilde kötü bir süreç, mükemmel oyuncuları faydasız hale getirebilmektedir. Burada güçlü takım oyuncusundan kasıt çok iyi bir programcı değil, takımın diğer üyeleri ile başarılı bir şekilde iletişim kurabilen ve çalışabilen ortalama bir programcıdır. Kişi ve iletişim öncelikli olsa da süreçlerin yapılandırılması ve süreçlerde kullanılan araçların etkinliği de önemlidir. Çünkü geliştirme sürecinde kullanılan derleyiciler, geliştirme ortamları, kaynak kod kontrol sistemleri gibi araçlar geliştirme takımına uygun fonksiyonlar sağlayabilir.

• Çalışır durumda olan yazılım, detaylı dokümantasyondan daha önceliklidir.

Dokümantasyonu olmayan bir yazılım felakettir. Çünkü program kodları iletişim için ideal bir ortam değildir. Proje takımı sistemi tanımlayan veya tarif eden belgeleri oluşturmaya mutlaka ihtiyaç duyar. Fakat bu dokümantasyon faaliyetleri çoğaldıkça yazılım geliştirme süresi azalır. Bunun dengesinin sağlanması önemlidir. Ayrıca müşterinin asıl ihtiyaç duyduğu şey çalışan yazılım olduğuna göre dokümantasyon çalışmaları gerektikçe gerçekleştirilmeli ve asıl amaçtan uzaklaştıran bir duruma dönüşmemelidir.

• Müşteri ile işbirliği ve beraber çalışma, sözleşmelerden ve anlaşmalardan daha önceliklidir.

Başarılı projeler düzenli ve sık aralıklarla müşteri geri beslemesine ihtiyaç duyar. Müşterinin geliştirme takımına yakın çalışması bunun sağlanması açısından çok önemlidir.

• Değişikliklere uyum sağlayabilmek, mevcut planı takip etmekten daha önemlidir.

Değişikliklere cevap verebilme yeteneği, yazılım projesinin başarılı ya da başarısız olmasını belirleyen çok önemli bir özelliktir. Bu nedenle planların esnek olmasına, süreç boyunca teknolojik ve iş alanındaki değişikliklere uyum sağlayabileceğinden emin olmak gereklidir. Bunu sağlamak için planlamanın yakın zamanda gerçekleşecek planları detaylı, sonraki zamanlarda gerçekleşecek planları ise daha yüzeysel ve detaysız oluşturulmalıdır. Böylelikle zamanla belirsizliği azalıp netleşecek gereksinimlerin ortaya çıkaracağı yeni durumlara adaptasyon sağlanabilir.

2.3.8.3 Çevik geliştirme prensipleri

Çevik manifestoda yer alan ifadeler on iki prensip ile somut hale getirilmekte ve açıklanmaktadır [5]. Bunlar:

1. En önemli öncelik erken ve sürekli olarak kullanılabilir programlar oluşturarak, müşteriyi tatmin etmektir.

Bu prensip ile aslında programın müşteri tarafından talep edildiğini ve müşteriyi sadece istekleri doğrultusunda geliştirilmiş ve çalışır durumda olan bir programın tatmin edebileceği dile getirilmektedir. Peki, müşteri nasıl tatmin edilebilir? Bunun için proje ekibinin proje başlangıcından itibaren ve sürekli olarak çalışır durumda olan programlar oluşturarak, müşteriye sunması ve görüşlerini alması gerekmektedir. Bu sayede müşteri inceleyebileceği ve çalışır durumda olan bir program aracılığıyla gereksinimlerinin ne oranda gerçekleştirimle (kodlama) örtüştüğünü kontrol edebilir. Gereksinimlerin değişmesi ya da müşterinin istekleri dışında bir yazılım yapılması durumunda müşteri yazılım sürecine müdahale edebilir ve gerekli değişiklikleri talep edebilir. Bu durum proje sonunda müşteri gereksinimleri ile yüksek oranda örtüşen bir programın oluşmasını sağlar.

2.Yazılımın ilerleyen dönemlerinde gelse bile talep edilen değişiklikler hoş karşılanmalıdır. Çevik süreçler, değişiklikleri müşterinin rekabetteki avantajını korumak ve sağlamak için kullanılırlar.

Geleneksel yazılım metotlarının uygulandığı projelerde mimarinin ve planlamanın büyük bir bölümü gerçekleştirim öncesi oluşturulur. Geleneksel yazılım, hazırlanan planlardan sapmadan gerçekleştirilmeye çalışılır. Bu, müşteri tarafından talep edilen değişikliklerin göz ardı edilmesi anlamına gelmektedir. Böyle bir sürecin sonunda müşteri isteklerini tatmin etmeyen ve müşterinin piyasadaki rekabet kabiliyetini sınırlayan programlar oluşur. Bunu engellemek için her zaman müşteriden gelen değişiklik taleplerinin gerçekleştirim esnasında dikkate alınması gerekmektedir.

3. Kısa zaman aralıkları tercih edilerek iki haftadan iki aya kadar çalışır yazılım ortaya koyulmalıdır.

Çevik süreçlerde programlar hem iterasyonlu hem de artırımlı olarak geliştirilir. Bir iterasyon, yazılım için gerekli tüm safhaları ihtiva eden belirli bir zaman biriminde gerçekleştiriminin gerçekleştirildiği süreçtir. Program ayrıca artırımlı oluşturulur, çünkü ekip üyeleri her iterasyonda müşterinin seçtiği gereksinimlere yoğunlaşarak

programı yavaş yavaş oluşturur. Her iterasyon sonunda program müşteriye sunularak, görüşleri alınır.

4. Müşteri ve yazılımcılar proje süresince beraber çalışmalıdır.

Çevik projelerde müşteri ile proje ekibinin beraber çalışması doğaldır. Programdan olan beklentileri en iyi müşteri bilebileceği için, sürekli müşteriden geri dönüm alınması gerekmektedir. Bunun en kolay yolu müşteri ile proje ekibinin beraber çalışmasıdır.

Proje ekibi, özellikle yazılımcılar müşteri tarafından dile getirilen gereksinimlerin fizibilitesini (yapılabilirlik) araştırırken, her gereksinim için kodlama zamanını tahmin ederler. Bu doğrultuda müşteri; kendi tanımlamış olduğu gereksinimlere bir öncelik sırası vererek, hangi gereksinimlerin öncelikli olarak gerçekleştirilmesi gerektiğini belirler. Yazılımcılar bu konularda müşteriye yardımcı olarak, gereksinimlerin daha somut hale getirilmelerini sağlarlar. Bunlar genellikle birkaç cümleden oluşan kullanıcı hikayelerine dönüştürülür. Müşteri ve yazılımcılar arasındaki sürekli diyalog yanlış anlaşılmaları ortadan kaldırır. Bu yüzden müşterinin her gün yazılımcının erişebileceği bir mesafede olması gerekmektedir.

Geleneksel yazılım metotlarında bunun böyle olmadığı görülmektedir. Projenin ilerleyen safhalarında müşteri tarafından talep edilen değişiklikler olumlu karşılanmaz, çünkü bu proje başlangıcında yapılan anlaşmalara ters düşmektedir. Başlangıçta ne yapılmasına karar verildiyse, yazılımcılar rahatsız edilmeden mevcut gereksinimlerin gerçekleşmesine odaklanabilmelidirler. Böyle bir metotla oluşan programın, müşteri gereksinimlerini ne ölçüde tatmin edebileceği ortadadır.

5. Projelerin motivasyonu yüksek bireyler tarafından yapılmasını sağla, onlara ihtiyaç duydukları ortamı ve desteği ver ve işi bitirebileceklerine inan.

Çevik projeler bireylerden oluşur ve her yazılımcı kendi bireysel karakteri ile takımın bir parçasıdır. Ekip elemanlarının değişik karakterde olmaları doğaldır. Yazılımcıların, onlara duyulan güvenin hissettirilmesiyle motivasyonları arttırılır. Onların sorumluluk almaları sağlanarak, özgüvenlerinin pekişmesi sağlanır.

Yazılımcılar birbirlerine yardım ederler. Onlar arasında kıdem farkı yoktur. Bilen bilmeyene öğreterek, kısa bir zamanda aynı teknik seviyeye gelmeleri sağlanmış olur.

6. Bilgi alışverişinde en verimli ve efektif yöntem takım içinde yüz yüze konuşmaktır.

Çevik projelerde bilgi alışverişi yüz yüze gerçekleşir, çünkü bilginin transfer esnasında en az hasara uğradığı yöntemlerinden birisi budur. Yazılımcılar arasındaki kişisel konuşmalar güveni artırır ve yanlış anlaşılmaları ortadan kaldırır. Sorunlar hemen açıklığa kavuşturulabilir.

7. Çalışır durumda olan program ilerlemenin ana göstergesidir.

Müşteriye kısa aralıklarla çalışır durumda olan lakin henüz tamamlanmamış bir program sunulduğunda, müşteri mevcut programın kendi gereksinimlerini tatmin edecek durumda olup olmadığını kontrol edebilir. Bu yüzden müşteri çalışır durumda olan programı kullandıktan sonra gereksinimlerine uygunluğunu tespit edebilir.

8. Çevik süreçler etkili yazılım yöntemlerini destekler. Müşteri, yazılımcılar ve kullanıcılar sabit bir tempoda beraber çalışabilmelidirler.

Çevik projelerde sabit bir çalışma temposunun oluşturulması büyük önem taşımaktadır. Yazılımcılar arasında görev eşit bir şekilde paylaştırılır. Fazla mesai yapılması hoş karşılanmaz. Bu ayrıca çalışanların motivasyonunu olumlu etkiler. Proje ilerledikçe iş azalır. Sonradan yazılımcıları kötü sürprizler beklemez, çünkü ortada iyi test edilmiş ve çalışır durumda olan bir program vardır.

9. Teknik mükemmelliğe devamlı özen gösterilir ve iyi tasarım çevikliği kuvvetlendirir.

Çevik projelerde kalite beklentisi yüksektir. Yazılım temin edilen en iyi araç ve yetenekli yazılımcılarla gerçekleştirilir. Bu süreçte programın tasarımı sürekli optimize edilir. Yazılımcılar yeniden yapılandırma esnasında buldukları tasarım hatalarını hemen giderirler.

10. Sadelik (basitlik) esastır.

Bir programı mümkün olan en basit tarzda gerçekleştirmek, programın karmaşıklığını azaltır. Karmaşıklığı düşük olan bir programın bakımı ve geliştirilmesi kolaydır. Yazılımcılar, yazılım esnasında sadece kendilerinden o anda olan beklentiyi tatmin edecek kadar kod yazarlar.

11. En iyi mimariler, gereksinimler ve tasarımlar kendi kendine organize olabilen takımlardan çıkar.

Çevik takımlar kendi başlarına organize olabilme özelliğine sahiptir. Böyle takımlar görevleri kendi aralarında eşitçe paylaşırlar. Takımlar ve bireyler arasındaki görüşmeler ve bilgi alışverişi sonucunda mimari ve tasarım gözden geçirilir ve optimize edilir. Ayrıca bireyler arası yapılan konuşmalar müşteri tarafından dile getirilen gereksinimlerin açıklığa kavuşturulmalarını ve daha iyi anlaşılmalarını sağlarlar.

12. Belirli zaman dilimlerinde takım nasıl daha etkin olabileceği konusunda kendini sorgular ve edindiği bilgiler doğrultusunda çalışma tarzını adapte eder.

Çevik takımlarda bilgi alışverişi ve devamlı öğrenme çok önemlidir. Belirli zaman aralıklarıyla takım çalışanları bir araya gelerek, fikir alışverişinde bulunurlar ve çalışma yöntemlerini sorgularlar. Edinilen tecrübeler doğrultusunda yazılım sürecinde adaptasyonlar gerekebilir. Nihai amaç en verimli şekilde çalışabilmek ve müşteri tarafından kabul görmüş bir program oluşturabilmektir [4].

2.3.8.4 Çevik geliştirmenin faydaları

Çevik geliştirme ile elde edilen faydaların en önemlileri şunlardır:

• Çevik projelerde müşteri gereksinimleri ve bu gereksinimlerin karşılığı olan program paralel olarak gelişir. Müşteri projenin gidişatına her zaman müdahale etme yetkisine sahiptir. Bu uzun süren projelerde önem taşımaktadır, çünkü çoğu zaman proje başlangıcında müşteri kendi gereksinimlerini tam olarak bilmeyebilir. Zaman içinde oluşan ilk prototipler müşterinin kafasında nasıl bir sistem istediği hakkında daha net bir resmin oluşmasını sağlar. Projedeki geri besleme mekanizmalarıyla müşteri gereksinimleri her zaman değişikliğe uğrayabilir.

• Bu modelde geri beslenme merkezi bir rol oynamaktadır. Çeşitli safhalarda sağlanan geri beslenme ile projenin hangi durumda olduğu saptanır. Programcılar hazırladıkları testler aracılığıyla geri beslenme sağlayarak, geliştirdikleri bileşenlerin hangi durumda olduklarını tespit ederler.

• Sürekli entegrasyon yapılarak programın hangi durumda olduğu geri beslenme olarak elde edilir. Müşteriye kısa aralıklarla programın yeni sürümü sunularak, geri beslenme sağlanır.

• Proje başlangıcında detaylı dokümantasyon ve tasarım oluşturulmaz.

• Programcılar test güdümlü çalışır ve oluşan tasarımı her yeni testle gözden geçirirler. Eğer tasarım yetersiz kalırsa, gerekli tasarım değişikliklerini, ilerideki testlerin çıkmaza girmesini engellemek için uygularlar.

• Her iterasyon başlangıcında müşteri tarafından dile getirilen gereksinimler analiz edilir ve geliştirilecek olanlar seçilir. Müşteri her gereksinim için bir öncelik sırası belirler. Öncelik sırası yüksek olan gereksinimler öncelikli olarak gerçekleştirilir. Her iterasyon sonunda gerekli değerlendirmeler yapılarak, oluşan problemler tartışılır ve tekrar meydana gelmelerini engellemek için gerekli önlemler alınır.

• Test güdümlü çalışıldığı için kod kalitesi çok yüksek olur. Gün boyunca programcılar kendilerine değişik bir programcıyı takım arkadaşı olarak seçerek, implementasyonu gerçekleştirirler. Kısa bir zaman sonra programcılar arasındaki teknik bilgi aynı seviyeye gelir ve her programcı programın herhangi bir bölümünde çalışacak hale gelir. Bu şekilde sadece bir programcının kod hakkında bilgi sahip olması engellenir.

• Programcılar aktif olarak proje planlamasında yer alırlar. Onlar gereksinimlerin tespitinde müşteriye yardımcı olurlar ve zaman tahminlerinde bulunarak, proje planlaması için gerekli verilerin oluşturulmasını sağlarlar. Bu programcılara belirli bir sorumluluk yükler. Kendisine güvenildiğini bilen ve sorumluluk sahibi bir programcının öz güveni ve motivasyonu artar.

• Çevik projelerde iyi bir çalışma ortamının ve temposunun oluşturulması fazla mesai yapılmasını engeller. Fazla mesai yapılmayacak diye bir kural yoktur. Fakat fazla mesai bir kural haline gelmemelidir. Bu durum tüm ekibin motivasyonunu negatif etkiler. Hatalar genelde isteksizce yer alınan fazla mesailerde meydana gelir. Proje çalışanları sekiz saat olan ve fazla mesai yapılmayan iş günlerinde daha verimli olurlar.

• Müşteriye kısa aralıklarla çalışabileceği bir sürüm sunulur. Program tamamlanmamış olsa bile, müşteri hazır bölümleri kullanarak, yaptığı yatırımın hızlı şekilde geri dönmesi sağlanır. Programcılar ve müşteri arasında devamlı iletişim vardır.

• Programcılar soru ve sorunları müşteri ile paylaşarak, kısa sürede çözüm üretebilirler.

• Müşterinin piyasadaki değişikliklere ve bununla birlikte rekabet ortamına ayak uydurabilmesi önemlidir. Çevik süreç hızlı reaksiyon göstererek, bu değişikliklere ayak uydurulmasını sağlar. Rekabete ayak uydurabilmek için hızlı reaksiyon gösterebilmek hayatı bir önem taşımaktadır.

2.3.8.5 Çevik metotların kullanımı

Çevik metotların kullanılmasının en uygun olduğu durumlar şunlardır:

• Projenin yazılım evresinde müşteriden gelebilecek talep değişikliklerinin tahmin edilemez olması.

• Projenin parçalarının önce tasarlanıp ardından hemen geliştirilmesinin gerekmesi ve önceden ne yapılacağını, ayrıntılı yol haritasını ve tasarımını tahmin etmenin çok güç olması.

• Analiz, tasarım ve test etme süreçlerinin ne kadar zaman alacağının önceden bilinememesi.

• Yazılım ekibinin birlikte çalışmak, hiyerarşiye önem vermemek, sağlam iletişim kurmak gibi özelliklere sahip olması.

Elbette bazı durumlarda çevik programlamadan vazgeçilmesi gerekebilir. Örneğin çok kişinin dâhil olduğu projelerde çevik metotlar ile proje geliştirmek mümkün olmayabilir ya da aynı yerde bulunmayan takım arkadaşları ve hiyerarşik yapının her an hâkim olduğu şirket ortamlarında klasik yöntemler daha uygun olabilir [3].

2.3.8.6 Çevik metodoloji sürecinin farkı ve avantajları

Yazılım geliştirme süreci analiz, tasarım, kodlama, test, sürüm ve bakım gibi safhalardan oluşur. Geleneksel yazılım metotlarında bu safhalar Şelale modelinde olduğu gibi doğrusal olarak işler.

Şelale modelinin özelliklerini şu şekilde sıralayabiliriz:

1. Şelalenin her basamağında yer alan aktiviteler eksiksiz olarak yerine getirmeden bir sonraki basamağa geçilmez.

2. Her safhanın sonunda bir tamamlama dokümanı oluşturulur. Bu yüzden Şelale modeli doküman güdümlüdür.

3. Kullanıcı katılımı başlangıç safhasında mümkündür. Kullanıcı gereksinimleri bu

safhada tespit edilir ve ayrıntılandırılır. Daha sonra gelen tasarım ve kodlama safhalarında müşteri ve kullanıcılar ile diyaloğa girilmez.

Bu modelin beraberinde getirdiği problemleri şu şekilde sıralayabiliriz:

1. Safhaların birbirinden kesin olarak ayrı tutulmaları gerçekçi değildir. Gerçek projelerde safhalar arasındaki bu sınırlar yok olabilir.

2. Teoride safhalar birbirlerini takip ederler. Gerçek projelerde bunun bazen mümkün olmadığını ve önceki safhalara geri dönülmek zorunda kalındığını görebiliriz.

3. Safhalar arası geri dönüm yetersizdir. Model değişikliğe açık değildir.

4. Müşteri gereksinimlerinin proje öncesi ayrıntılı olarak kâğıt üzerinde oluşturulması ilerde sorun yaratabilir. Müşteri gereksinimleri değişikliğe uğrayabileceği için, yazılım sisteminin de yapısal değişikliğe uğraması kaçınılmaz olabilir. Böyle bir durum maliyeti artırır, çünkü yeni ve değişen gereksinimleri gerçekleştirimi için modelde yer alan safhaların birkaç kere uygulanması gerekebilir.

5. Sistemin tamamlanarak kullanılır hale gelmesi uzun zaman alabilir. Böylece eksikleri, hataları erkenden görmek mümkün olamaz.

6. Başlangıçta yapılan hataların tespiti çok uzun zaman alabilir. Bu hataların giderilmesi maliyeti yükseltir.

7. Modül gerçekleştirimleri için zaman tahminleri proje planlarını oluşturan yöneticiler tarafından yapılır. Teknik bilgiye sahip olmayan şahıslar tarafından yapılan bu tahminler çoğu zaman doğru değildir. Bu durum proje planlama sürecini negatif etkiler.

Proje başlangıcında her ayrıntıyı göz önünde bulundurmak mümkün olmadığı için, Şelale modeliyle geliştirilen yazılım sistemlerinin müşteri gereksinimlerini tam tatmin etmediğini görmek olağandır. Bunun önüne geçebilmek için projenin başlangıç safhasında analiz için çok zaman harcanır ve müşteri gereksinimleri en ince ayrıntıya kadar tespit edilir. Aslında proje başlangıcıyla oluşturulan dokümanlar ileride eskimiş olur. Çünkü müşteri gereksinimleri piyasa ve rekabet koşulları gereği değişikliğe uğramış olabilir. Ne yazık ki Şelale modeli bunları dikkate almaz ve müşterinin talep ettiği değişiklikleri aza indirmeye çalışır. Bunun bir sebebi de sonradan gelen değişiklik taleplerinin maliyeti yükseltmesidir, çünkü bu durumda Şelale modelinde yer alan safhaların birkaç kere uygulanması gerekebilir.

Bu çerçeveden bakıldığında proje sonunda oluşan program müşterinin aktüel gereksinimlerini tatmin etmez durumda kalır. Program daha çok müşterinin proje başlangıcında sahip olduğu gereksinimleri tatmin edecek şekilde tasarlanmıştır. Değişiklik talepleri alınır fakat işleme konmaz. Bir sonraki sürüme bırakılır. Baştaki plan ve analiz bozulmasın istenir. Ama bu da sürümlerin geç kalmasına, arkadan gelmesine neden olur.

Çevik süreçlerde durum farklıdır. Çevik süreç değişimi kabul eder ve onunla yaşamayı kolaylaştırmak için yeni yazılım metotları sunar. Çevik süreçlerde iterasyon bazında çalışmalar sürdürülür. Her iterasyon bir ila dört haftalık zaman dilimlerinden oluşur ve Şelale modelinde yer alan safhaları ihtiva eder. Aslında her iterasyon bir mini Şelale modeli ihtiva ediyor diyebiliriz.

Çevik süreç modelinin avantajları şöyledir:

1.Çevik projelerde müşteri gereksinimleri ve bu gereksinimlerin karşılığı olan program paralel olarak gelişir. Müşteri projenin gidişatına her zaman müdahale etme yetkisine sahiptir. Bu uzun süren projelerde önem taşımaktadır, çünkü çoğu zaman proje başlangıcında müşteri kendi gereksinimlerini tam olarak bilmeyebilir. Zaman içinde oluşan ilk prototipler müşterinin kafasında nasıl bir sistem istediği hakkında daha net bir resmin oluşmasını sağlar. Projedeki geri besleme mekanizmalarıyla müşteri gereksinimleri her zaman değişikliğe uğrayabilir.

2. Bu modelde geri besleme merkezi bir rol oynamaktadır. Çeşitli safhalarda sağlanan geri besleme ile projenin hangi durumda olduğu saptanır. Yazılımcılar hazırladıkları testler aracılığıyla geri besleme sağlayarak, gerçekleştirdikleri bileşenlerin hangi durumda olduklarını tespit ederler. Sürekli tümleştirme yapılarak programın hangi durumda olduğu geri besleme olarak elde edilir. Müşteriye kısa aralıklarla programın yeni sürümü sunularak, geri besleme sağlanır.

3. Her iterasyon başlangıcında müşteri tarafından dile getirilen gereksinimler analiz edilir ve uygulamaya alınacak olanlar seçilir. Müşteri her gereksinim için bir öncelik sırası belirler. Öncelik sırası yüksek olan gereksinimler öncelikli olarak uygulamaya alınır. Her iterasyon sonunda gerekli değerlendirmeler yapılarak, oluşan problemler tartışılır ve tekrar meydana gelmelerini engellemek için gerekli önlemler alınır.

4. Test güdümlü çalışıldığı için kod kalitesi çok yüksek olur. Gün boyunca yazılımcılar kendilerine değişik bir yazılımcıyı takım arkadaşı olarak seçerek (pair

programming), gerçekleştirimi gerçekleştirirler. Kısa bir zaman sonra yazılımcılar arasındaki teknik bilgi aynı seviyeye gelir ve her yazılımcı programın herhangi bir bölümünde çalışacak hale gelir. Bu şekilde bir yazılımcının kod hakkında bilgi monopolüne sahip olması engellenir.

5. Yazılımcılar aktif olarak proje planlamasında yer alırlar. Onlar gereksinimlerin tespitinde müşteriye yardımcı olurlar ve zaman tahminlerinde bulunarak, proje planlaması için gerekli verilerin oluşturulmasını sağlarlar. Bu yazılımcılara belirli bir sorumluluk yükler. Kendisine güvenildiğini bilen ve sorumluluk sahibi bir yazılımcının özgüveni ve motivasyonu artar.

6. Çevik projelerde iyi bir çalışma ortamının ve temposunun oluşturulması fazla mesai yapılmasını engeller. Fazla mesai yapılmayacak diye bir kural yoktur. Lakin fazla mesai bir kural haline gelmemelidir. Bu durum tüm ekibin motivasyonunu negatif etkiler. Hatalar genelde isteksizce yer alınan fazla mesailerde meydana gelir. Proje çalışanları sekiz saat olan ve fazla mesai yapılmayan iş günlerinde daha verimli olurlar.

7. Müşteriye kısa aralıklarla çalışabileceği bir sürüm sunulur. Program tamamlanmamış olsa bile, müşteri hazır bölümleri kullanarak, yaptığı yatırımın hızlı şekilde geri dönmesini sağlanır.

8. Yazılımcılar ve müşteri arasında devamlı iletişim vardır. Yazılımcılar soru ve sorunları müşteri ile paylaşarak, kısa sürede çözüm üretebilirler. Müşterinin piyasadaki değişikliklere ve bununla birlikte rekabet ortamına ayak uydurabilmesi önemlidir. Çevik süreç hızlı reaksiyon göstererek, bu değişikliklere ayak uydurulmasını sağlar. Rekabete ayak uydurabilmek için hızlı reaksiyon gösterebilmek hayatı bir önem taşımaktadır [6].

2.3.8.7 Çevik metodolojiler

Çevik metotlar, kendi içerisinde özü aynı fakat süreçlerinde farklılaşan çeşitli alt kollara ayrılmaktadır. Zaman içinde, bir metamodel olarak kabul edebileceğimiz çevik süreci gerçekleştiren değişik çevik süreç türleri oluşmuştur. Yaygın olarak kullanılan Uç programlama, Scrum, Çevik Tümleşik Süreç, Özellik Güdümlü Programlama, Yalın Geliştirme, Dinamik Sistem Geliştirme Metodolojisi (DSGM) ve Microsoft Çözüm Yapısı çevik metodolojileri vardır [7]. Bu metodolojiler arasında en yaygın uygulananları Uç programlama ve Scrum'dır.

Uç programlama

Kent Beck tarafından 1999 yılında bir yazılım geliştirme disiplini olarak ortaya çıkarılmıştır. Uç programlama kolay, grup içi iletişime önem veren, geri beslemelerin daha fazla olmasına imkân sağlayan bir yazılım geliştirme yöntemidir [8]. Yazılım geliştirmede kolaylığı ve esnekliği sağlamak için Uç programlama 12 farklı uygulamayı öngörür. Bunlar;

- Planlama oyunu
- Küçük ve kısa aralıklı sürümler
- Metafor Basit tasarım
- Önce test
- Yeniden yapılandırma
- İkili programlama
- Ortak kod sahiplenme
- Sürekli tümleştirme
- Haftada 40 saat çalışma
- Ekipte müşteri
- Kodlama standartları

Scrum

Scrum seksenli yıllarda Jeff Sutherland tarafından uygulanmış ve Kent Schwaber ve tarafından da detaylandırılmış bir çevik süreçtir [9]. Sutherland ve Schwaber daha sonra birlikte çalışmış, Microsoft, Borland ve Hewlett-Packard'da oluşan yöntemsel deneyimlerle birlikte endüstriyel kontrol yöntemlerini uygulayarak Scrum'ı yeniden şekillendirmişlerdir. Aslında Scrum, Rugby oyununda kullanılan bir terimdir. Oyuncular kısa bir süre için bir araya gelerek, bir sonraki oyun hamlesi hakkında fikir alış verişinde bulunurlar, yani kısa bir toplantı yaparlar. Scrum daha çok proje yönetim ve planlama ile ilgili değerlere ve pratiklere konsantre olmaktadır. Yazılımın nasıl yapılması gerektiği hakkında detay ihtiva etmez. Bağımsız bir çevik yöntem olarak algılanmamalıdır. Birçok projede Scrum, Uç programlamaya da Bütünleşik Süreç ile desteklenir [10].

Scrum'ın 9 tane pratiği vardır. Bunlar;

• Genellikle 10-30 günlük iterasyon süreleri

• 5-10 kişilik takımlar

• Kısa süreli (3-12 ay) projeler

• İterasyon içerisinde takım dokunulmazdır

• Kendi-kendini düzenleyen, yönlendiren takım

• Her iterasyon müşteri güdümlüdür ve adapte olan bir planlama içerir.

• Kararlar kısa sürede alınır ve hemen uygulanır.

• Ayakta yapılan ve takım üyelerinin bir çember oluşturacak şekilde dizildiği ve belirli soruların sorulduğu günlük Scrum toplantıları.

• Her iterasyonun sonunda dış paydaşlara yapılan demo

Çevik tümleşik süreç

Çevik Tümleşik Süreç, Rasyonel Bütünleştirme Süreci'nin basitleştirilmiş bir versiyonudur. Çevik Tümleşik Süreç, Rasyonel Bütünleştirme Süreci'ne sadık kalarak çevik teknikleri ve kavramları kullanan yazılım geliştirme yaklaşımının anlaşılmasını kolaylaştıran bir metodolojidir.

Çevik Tümleşik Süreç aşağıdaki ilkelere dayanmaktadır:

• Personel ne yapacağını bilir

• Basitlik Çeviklik

• Yüksek değerli faaliyetlere odaklanma

• Araç bağımsızlığı

• İhtiyaçları karşılamak için, yazılım ürününün uyarlanması

Özellik güdümlü programlama

Özellik Güdümlü Programlama (ÖGP), Jeff De Luca tarafından doksanlı yılların sonunda geliştirilmiş bir çevik metodolojidir. ÖGP, özellik güdümlü çalışır. Sisteme yeni bir özellik kazandırılmadan önce, ayrıntılı bir tasarım çalışması yapılarak bu özelliği kapsayan mimari yapı oluşturulur. Bu yüzden ÖGP daha çok tasarım odaklı işleyen bir çevik süreçtir. 5 temel süreç vardır;

• Genel bir mimari geliştirme

• Özellik listesini oluşturma

• Özellikleri planlama

• Özellikleri tasarlama

• Özellikleri gerçekleştirme

Yalın programlama

Bob Charette tarafından geliştiren "Yalın Programlama" 1980'de Japon otomobil endüstrisindeki yeniden yapılanmada kullanılan yalın üretimin prensipleri baz alınarak, 2000'li yılların başında yazılıma uyarlanmış bir çevik metodolojidir. Bob, geleneksel metodolojilerin risk olarak gördüğü değişimi, kısıtlayıcı yönetim uygulamalarıyla kontrol ederek bu değişimin yeni fırsatları üretilebilmesini sağlamıştır. 12 prensibi vardır;

• Müşteri memnuniyeti en önemli önceliktir

• Bütçeyi en iyi şekilde kullanır

• Müşteriyle sürekli iletişim başarıyı arttırır

• Her proje takım çalışmasıdır

• Her şey değişebilir

• Yama değil, kapsamlı çözüm

• Tamamla, yapılandırma

• Minimalism temeldir.

• Bugün %80 tamamlamış olmak, yarın % 100 tamamlayacak olmaktan iyidir.

• Teknolojiyi gereksinimler belirlesin

• Ürünün büyümesi demek özelliklerin büyümesi demektir

• Lean geliştirmeyi asla sınırlarının ötesine götürmeyin.

Dinamik sistem geliştirme metodolojisi

Dinamik Sistem Geliştirme Metodolojisi (DSGM), hızlı yazılım geliştirme adımlarının kontrolünü sağlayan bir metodolojidir. Büyük Britanya' da bir konsorsiyum tarafından 1990'lı yılların ortalarında geliştirilmiştir [7]. Bu yaklaşımla

fonksiyonel gereksinimlere öncelik verilmesi gereklidir. Aynı zamanda yüksek kalite ve değişen gereksinimlere adapte olunması önemlidir. Sekiz tane prensibi vardır. Bunlar;

- İş ihtiyaçlarına odaklanma
- Zamanında teslim
- İş birliği yapmak
- Asla kaliteden ödün verme
- İteratif geliştirme
- Sağlam temellerden artırımlarla büyür.
- Sürekli iletişimde olunmalı
- Kontrolde olduğunu gözle

Microsoft çevik çözüm yapısı

Microsoft Çevik Çözüm Yapısı, ilk olarak Microsoft tarafından 1993 yılında versiyon 1.0 olarak yayınlanmıştır. Bunu sonraki yıllarda sırasıyla; 1997'de versiyon 2.0, 1999'da versiyon 2.5, 2002'de versiyon 3.0 ve en son 2005 yılında ise versiyon 4.0 yayınlanması izlemiştir [11]. Başarılı bilişim çözümleri için geliştirme ekibinin nasıl organize edileceği, projelerin nasıl planlanacağı, süreç yapısının nasıl gerçekleştirileceği, risklerin değerlendirilip kurulumun nasıl tamamlanacağıyla ilgili bir süreç yaklaşımıdır. Takım ve süreç olmak üzere iki modelden oluşmaktadır [12].

Aşağıdaki prensipleri benimsemektedir:

- Müşteri ile birlikte geliştirme
- Açık iletişim kurmak
- Ortak bir vizyonda hareket etmek
- Kalitenin herkese ait ortak bir değer olduğuna inanmak
- Sürekli gelişim sağlamak

Çevik modelleme

Çevik Modelleme, yazılım sistemlerinin modelleme ve etkili dokümantasyon işlerini kolaylaştırmak üzere bu işleri iş ürünü olarak fazla dikkate almayan ve

önemsemeyen diğer çevik yöntemleri tamamlayan bir yapıdır. Yazılım mühendisleri tarafından uygulanacak bazı prensip ve değerlerin belirlediği pratikler kümesidir. Tamamen detaylandırılmış özel bir süreç modelleme yöntemi tanımlamaz. Sadece modellemenin daha etkin olması için tavsiyelerde bulunur. Çevik Modelleme gereksinimler, mimari ve tasarım modelleme işlerinde kullanılabilir [13].

IXP

Uç Programlama'dan doğan IXP'nin amacı UP'yi geliştirmek ve UP'de yer alan metot ve tekniklerin daha büyük organizasyonlar için adapte etmektir.

2.4 Yazılım Geliştirme Metodolojileri Problem Tanımı

Bu bölümde yazılım geliştirme metodolojileri genel olarak tanıtıldı. Görüldüğü gibi çok çeşitli ve birbirinden farklı özelliklere sahip metodolojiler kullanılmaktadır. Bilişim sektörü gün geçtikçe büyüyen ve yüksek miktarda paranın döndüğü bir sektördür. Bu metodolojilerden hangisinin seçileceği kararı şirketlerin büyüme hızına doğrudan etki etmektedir. Şirketler teknolojik gelişmelere hızlı uyum sağlamak zorunda, hızlı ve verimli çalışmak durumundadır, aynı zamanda sorunsuz ve kaliteli üretim yaparak kullanıcı isteklerine karşılık vermeleri gerekmektedir. Bu doğrultuda yapmaları gereken sektöre ve kendi şirket yapılarına en uygun yazılım geliştirme metodolojisini seçmektir. Bu noktada tezin konusunu oluşturan problem hangi yazılım geliştirme metodolojisinin seçileceğidir. Problemin çözümü için çok kriterli karar verme yöntemleri kullanılacaktır. Sonraki bölümlerde karar verme ve çok kriterli karar verme yöntemleri ile ilgili teorik bilgiler verilerek bu yöntemler tanıtılacaktır.

3. KARAR VERME

3.1 Tanımı

İnsanlar yaşamlarını sürdürebilmeleri ve gelecekte başarılı olabilmeleri için karar vermek durumundadırlar. Bunu yaparken de bilimsel ölçütleri dikkate almaları daha iyi kararlar vermelerini sağlar [14].

Karar verme, en basit tanımı ile hareket tarzları içinden en uygun seçeneği belirlemektir. Her bir karar beraberinde başka karar ihtiyaçlarını ve problem çözmeye dönük faaliyetleri getirir. Bu özellikleri itibariyle problem çözme ve karar verme yöneticinin faaliyetlerinde anahtar rolü oynar. Yönetim faaliyetinde hangi amaçların ön plana alınacağı, hangi fırsatların yaratılacağı, hangi kaynakların hangi ilkeler çerçevesinde tahsis edileceği ve alınan kararların icrasını kimlerin yürüteceği gibi konulardaki tercihlerin tümü birer karar niteliğindedir. Bu nedenle de, yönetici kendisini her an karar veren bir yönetsel mekanizma olarak algılamaktadır [15].

Bir iş ya da sorun hakkında düşünülerek verilen kesin yargıya karar denir. Bir karar meselelerin özel parçaları için tatmin edici durumları geliştirmeye niyetlenen bir davranışı üstlenmedir [5]. Karar verme ise, karar organının değişik seçeneklerle karşı karşıya bulunduğu durumlarda bunlar arasından kendi amaçlarına en uygun olanını seçme işlemi olup, karar süreci de bu işlemlerin sırasıyla yapılmasıyla gerçekleşebilecektir [16]. Tüm karar problemlerinin bir amacı olmalıdır. Amaç seçimi, karar verme sürecindeki en zor noktalardan biridir.

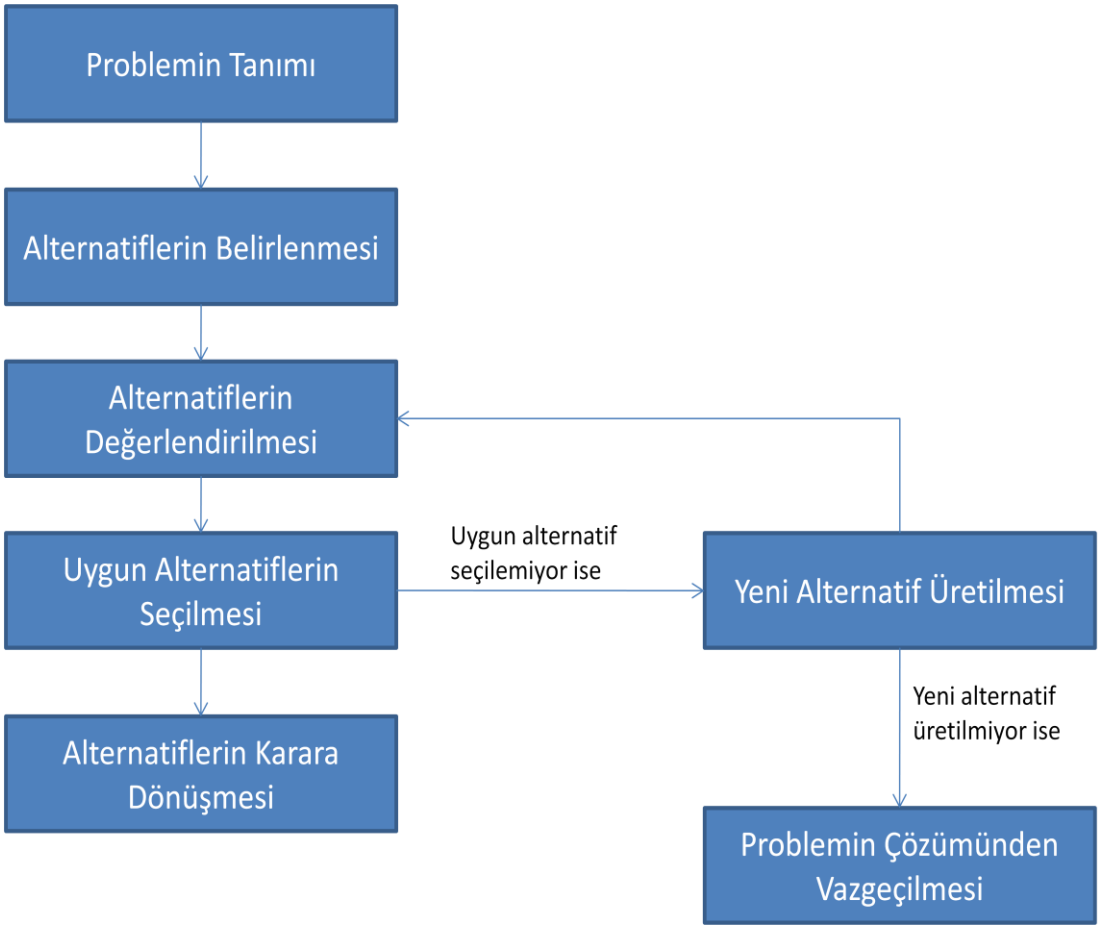
Karar verme, çeşitli amaçlar, bunlara ulaştıracak yollar, araçlar ve imkânlar arasından seçim ve tercih yapmakla ilgili zihinsel, bedensel ve duygusal süreçlerin toplamıdır. Yöneticiler mümkün olduğu sürece işlerini yürüttükleri kuruluşların amaçları ve bunlara ulaştıracak yol, yöntem, araç ve imkânlarının neler olduğunu bilmeli ve birer alternatifler dizisi ortaya koyarak sağlıklı bir seçim yapmalıdırlar.

Değerlendirmeye alınan birçok alternatif arasından seçilen alternatif, firmanın içinde bulunduğu koşullar ve yöneticinin bilgi, yetenek, kişilik ve eğilimi bakımından en uygun olanıdır [17].

3.2 Karar Verme Süreci

Karar verme, mevcut tüm alternatifler arasından, amaç veya amaçlara en uygun, mümkün alternatifler arasından bir veya birkaçını seçme süreci olarak tanımlanmaktadır [18].

Karar verme süreci Şekil 3.1'de gösterilmiştir.



Şekil 3.1 : Karar verme süreci.

Karar bir sonucu ifade eder. Dolayısıyla yönetici karar vermekle bir sürecin sonucunu açıklamış olur ve karar konusunu incelemek için sadece sonucu ifade eden "seçim" veya "tercih"in incelenmesi yetmez. Bunun gerisine giderek, seçim yapmaya gelinceye kadar nelerin olup bittiğine bakmak gerekir. Bu açıdan ele alındığında, karar verme işini bir süreç olarak görmek mümkündür. Dolayısıyla karar verme, belirli bir başlangıç noktası olan; değişik iş, faaliyet veya düşüncelerin birbirini izlediği ve sonunda bir tercihin yapılması ile sonuçlanan bir süreçtir [8].

Karar verme süreci bilgilenme, tasarlama ve seçim olmak üzere üç aşamadan oluşmaktadır [6] :

Bilgilenme: Bu aşama, problemi tanıma ve anlama aşamasıdır. İnsanları dinleme; çeşitli veri tabanlarını sorgulayarak ortamı inceleme, mevcut durum ile gelecekteki durum arasındaki farklılıkları belirlemek için beyin fırtınası yapma; şirketin güçlü ve zayıf yönlerini, önündeki fırsatları ve tehlikeleri inceleme gibi etkinliklerden oluşur ve sorunları ya da fırsatları ortaya koyar.

Tasarlama: Sorunun nedenleri ve alternatif çözümler bu aşamada ortaya konulur. Değişik çözümler bulmanın birkaç yolu vardır. Beyin fırtınası, literatür inceleme, araştırmalar yapma, ilgili değişik kişi ve kurumların farklı alternatifler önermelerini sağlamak için bir ilan verme sayılabilir.

Seçim: Alternatifler içinden en iyi olanının seçilmesi, karar verme sürecinin zihinsel olarak en zorlayıcı kısmıdır. Daha önceleri seçme işlemi, ilgilenilen seçeneklerin fayda ve sakıncalarının sezgisel olarak bir araya getirilmesi ile yapılırken, günümüzde bu süreçte alternatifler içinden en iyi alternatifin seçilmesini sağlayan analitik yöntemler kullanılmaktadır.

Daha ayrıntılı bir perspektifte karar verme süreci aşağıda belirtilen 6 aşamada incelenebilir.

3.2.1 Amaç veya problemin saptanması

Pek çok yönetim bilimci karar vermede birinci safhanın problemin saptanması, bir sonraki safhanın ise problemi işaret etmekte kullanılacak uygun amaçların ortaya çıkarılması olduğunu belirtmektedir. Bazıları ise bunun tersini ifade etmektedir. İkinci gruba göre bireyler, öncelikle amaçları anlamak için çok fazla çaba harcamakta daha sonra bu amaçlara ulaşmada kullanılacak yolları ya da karar seçeneklerini aramaktadır. Bu tartışmaya rağmen kararlar pek çok şekilde verilebilir. Bazen ikinci grubun düşündüğü gibi karar verme geleceğini şekillendirerek şanslı olunmakta, bazen de bireyler kendilerini beklenmedik zor koşulların içinde bulmaktadırlar. Her iki durumda da karar durumunun doğasını doğru kurmak, amaçları doğru tanımlamak ve anlamaktan geçmektedir [19].

Bilgi teknolojileri karar alma işlevini bir bütün olarak; daraltarak, genişleterek ya da diğer sistemlere bağlayarak ve bu fonksiyonu oluşturan alt sistem unsurlarına yönelik olarak da, karar almada yapılması gereken araştırmaların hızını artırmak, seçim işini programlara bırakmak ve değerlendirmede kullanılan teknikleri değiştirmek suretiyle etkilemektedir. Bu iki yoldan hangisi olursa olsun, bilgi teknolojileri karar almayı

genel olarak insan gücünün tekelinden alarak Bilgi Teknolojileri destekli karar sistemlerine aktarmaktadır [20]. Tanımlanmak istendiğinde sorunlarla değişik biçimde karşılaşılabilir [21].

• Gerçeğe Dayalı

• İşlevsel

• Taktiksel

• Stratejik

• İnsanlar

3.2.2 Amaç veya problemin irdelenmesi

Amacın belirlenmiş olması veya sorunun tanımlanmış olması "karar"ı ifade eden "seçim" için yeterli değildir. Bu amaç veya sorunların nedenlerinin, özelliklerinin, çözülmemesi halinde karşılaşılacak durumların, niteliklerinin vs. incelenmesi ve analiz edilmesi gerekir. Böyle bir irdeleme ile amaç belirleme ve sorun tanımlamada daha sıhhatli olunması sağlanacaktır [8].

İvedi bir sorun olduğunda ilk yapılması gereken sorunun gruplara ayrılmasıdır. Sorunu gruplandırma, sorunun kendisi hakkında karar vericiye genel bilgi verir. Bu yolla çözmenin ne kadar zaman alacağı kestirilebilir ve sorunun büyüklüğü görülebilir [21].

Bu ve benzeri sorularla tanımı yapılan sorun irdelenmiş ve böylece tanımın daha sıhhatli olması sağlanmış olacaktır. Sorunların daha sistematik bir şekilde tanımı ve irdelenmesi ile ilgili çeşitli teknikler geliştirilmiştir. Bunların içinde en bilinenleri Pareto Analizi, Balık Kılçığı Tekniği, Sebep-Sonuç Analizi ve Akış Diyagramlarıdır [8].

3.2.3 Çözüm alternatiflerinin belirlenmesi

Karar durumu ile uygun amaçlar kurulduktan sonra alternatiflerin oluşturulması ve yaratılması aşamasına geçilmektedir. Amaçların sıkı bir inceleme ve analizi ile başlangıçta görülemeyen alternatifler ortaya çıkarılabilir. Bu karar verme

yaklaşımının önemli bir faydasıdır. Ayrıca yaratıcılık konusunda yapılan araştırmalar yeni tekniklerin ortaya çıkmasına sebep olmuş bu da yeni alternatifler bulmada kullanılabilmelerine şans tanımıştır [22].

Alternatifler karar vermenin ham maddeleridir. Amaçların izlenmesinde sahip olunacak seçimler dizisini temsil ederler. Bu merkezi önemleri nedeniyle çok yüksek bir alternatif üretme standardı sağlanması ve kurulması gerekmektedir. Düşünülemeyen alternatif asla seçilemeyeceğinden yeni ve yaratıcı alternatifler araştırmanın ödülü çok yüksek olabilir [23].

3.2.4 Alternatiflerin irdelenmesi

Karar verici her bir alternatifi ayrı ayrı değerlendirir. Bir karar, mevcut bilgiye bağlı olarak icra edilebilir ya da hangi alternatifin seçileceğinin belirlenmesinde Yönetim Bilgi Sistemleri yeteneklerinin kullanılması arzu edilebilir. Örneğin, karar verici, belli bir alternatifin seçilmesi durumunda, elde edilecek muhtemel sonuçları tahmin etmek için bilgisayara dayalı bir model kullanabilir [24].

Fikir üretmede geleneksel bir teknik olarak yararlar ve zararların listesinin yapılması kullanılabilir. Özellikle bir seçeneğin diğeriyle karşılaştırılmasının gerektiği durumlarda kullanılacak en basit araçlardan biridir. Bazı alternatiflerin basit incelenmesi gerektiğinde bir liste yeterli olabilir. Bu liste, hem hangi yoldan gidilebileceğine karar vermede yardımcı olabilir, hem de bütün her şeyi perspektife koyar [21].

3.2.5 Karar kriterlerinin belirlenmesi

Geliştirilmiş ve irdelenerek bir sıraya konulmuş bulunan alternatifler arasından seçim neye göre yapılacaktır? Seçimi yapabilmek için bir seçim kriterine ihtiyaç vardır. Seçim kriteri, alternatif veya seçeneklerin özelliklerinden hangilerinin, bunları karar olarak seçerken kullanılacağını ifade etmektedir. Örneğin ‘maliyet’ bir seçim kriteri olabilir. Bu durumda en düşük maliyeti olan alternatif seçilecektir. ‘Gerektirdiği ek kaynak’ seçim kriteri olarak alınırsa, en az kaynak gerektiren alternatif seçilecektir: Seçilecek kriter ile problem tarifi ve sahip olunan kaynaklar arasında yakın bir ilişki vardır [8].

3.2.6 Karar verme

Karar verme sürecinin son safhası alternatifler arasından seçim yapmaktır. Seçilen alternatif "karar"ı temsil eder ve böylece işletmenin kaynaklarının nasıl kullanılacağı, hangi işlerin yapılacağı belirlenmiş olur. Sıra bu kararı uygulamaya gelir [8].

Bütün bu süreçte sistematik bir yaklaşım kullanmak gerekir. Bu sistematikliği de Karar Destek Sistemleri (KDS) sağlayabilir. Bir KDS'nin en çok kullanıldığı yer ve karar verme sürecinin özünde, olası alternatiflerin bir grubundan seçilen uygun çözümü bulma işi bulunmaktadır. KDS olası alternatifler grubunun analizinde niceliksel yaklaşımlar sağlamak ve yöneticiye problemin mümkün olan en iyi çözümünü seçmede yardım etmek için kullanılabilir [25].

Temelde bilgisayar destekli bir bilgi sistemi olan KDS gerekli veri ve bilgileri veri tabanından alarak, bunları sayısal yöntemlerle analiz eder ve yöneticinin daha doğru karar vermesine yardımcı olur. Bu nedenle, KDS ne yalnızca insan tabanlı ne de yalnızca bilgisayar destekli bir sistemdir. KDS işletmelerde yöneticilerin karar sürecinde ihtiyaç duydukları bilgi ve karar alternatiflerini üreten bilgisayar destekli bir insan makine sistemidir [26].

Gerek kişisel anlamda, gerekse toplumsal ölçekte ele alındığında, hemen her alanda karar verme ile karşı karşıya kalındığı görülmektedir. Kararlar bazen basit ve yalınken, bazen de çok yönlü ve karmaşık olabilmektedir. İster basit, ister karmaşık olsun, tüm karar verme süreçlerine analitik olarak yaklaşmak için, öncelikle karar verme sürecini içine alan faktörleri belirtmek gerekir [27].

3.3 Karar Verme Sürecindeki Faktörler

• Karar verici veya vericiler: mevcut alternatifler arasında, optimum alternatifi seçecek kişi ya da gruptur.

• Karar kriteri: Karar vericinin, alternatifleri yargılamada kullanacağı değer sistemidir.

• Seçenekler: Karar vericinin, arasından tercih yapacağı alternatif faaliyetlerdir.

• Olaylar: Karar vericinin, denetimi altında olmayan faktörlerdir.

• Amaç: Karar vericinin, varmayı veya gerçekleştirmeyi hedeflediği neticedir [27].

3.4 Karar Ölçütleri

Genel olarak, en uygun stratejiyi veren ölçütün seçimi pek kolay değildir. Çünkü stratejinin seçiminde kullanılan ve en iyi sonucu veren tek bir ölçüt yoktur. Bunun

nedeni, en iyinin, karar vericinin politikası, gelenek ve davranışıyla yakından ilgili olmakla beraber çevre koşullarının da etkisi altında olmasıdır [28].

Karar alma sürecinde, var olan problemler ile çok sayıda çözüm yolları geliştirmek amacıyla stratejiler saptanır. Stratejiler arasından en uygun olanı seçilirken değişik nitelikte karar ölçütleri uygulanabilir.

Bu ölçütler:

- Belirlilik altında karar verme,
- Risk altında karar verme,
- Belirsizlik altında karar vermedir.

3.4.1 Belirlilik altında karar verme

Belirlilik ortamında karar vermede, stratejilerin hangi koşullar altında gerçekleşeceği kesin olarak bilinmektedir. Bu tip bir karar alma problemi deterministik bir yapıya sahiptir. Bu ortamda, amaç fonksiyonunun en büyükleme ve en küçükleme olduğu göz önüne alınarak, stratejilerden biri seçilir [28]. Doğrusal programlama modelleri belirlilik altında karar vermenin bir örneğidir.

Burada, fikirlerin, duygu ve heyecanların, karar alternatiflerinin sayısal bir ölçekle sıralanmasını sağlayacak bir şekilde ölçüldüğü durumlara farklı bir yaklaşım getirmektedir. Analitik Hiyerarşi Prosesi bu yaklaşımın içindedir [29].

3.4.2 Risk altında karar verme

Risk ortamında karar vermede alınacak belirli bir karara ilişkin değişik sayıda koşullar söz konusudur. Her stratejinin her koşul altında elde edilebileceği sonuçlar belirli bir olasılık çerçevesinde oluşur. Bu gibi durumlarda stratejilerin ne gibi sonuçlar doğuracağı önceden bilinmez. Sonuçların gerçekleşmesi belirli olasılıklara dayanmaktadır. Olasılıklar göz önünde tutularak yapılan strateji seçimine risk ortamında karar verme denir [28].

3.4.3 Belirsizlik altında karar verme

Belirsizlik altında karar verme, risk altında karar vermede olduğu gibi sonuçları doğal durumlara bağlı alternatif hareketlerle ilgilenir. Belirsizlik durumunda, doğal durumların bağlı olasılık dağılımı ya bilinmiyor ya da belirlenemiyordur. Bu bilgi

eksikliği karar probleminin analizi için aşağıdaki yeni kriterleri ortaya çıkarmıştır:

- Laplace
- Minimaks
- Savage
- Hurwicz

Bu kriterler karar vericinin belirsizlik karşısındaki tutarlılık derecesine göre farklılık gösterir [29].

3.5 Çok Kriterli Karar Verme Yöntemleri

İşletmelerin hızla değişen çevresel koşullara uyum sağlamaları ve bu değişime paralel olarak etkin kararlar alabilmeleri, karar verme sürecinde çok sayıda nitel ve nicel faktörü bir arada değerlendirebilen bilimsel yöntemleri kullanmaları ile mümkündür [30].

ÇKKV yöntemleri karar verme sürecine, konu ile ilgili farklı kişileri dâhil edebilmekte ve karar verme problemi ile ilgili birçok faktörün farklı seviyelerde eşzamanlı olarak değerlendirilmesine imkân sağlamaktadır [30].

ÇKKV tanımlayıcı bir yaklaşımdır ve olası kararları tanımlayarak, nitelikleri ve değerlendirme kriterini tanımlayarak ve kriterin saptandığı bir "f" fayda fonksiyonunu da katarak problemi tanımlamayı içermektedir [31]. Anlam olarak ÇKKV birden fazla ve aynı anda uygulanan kriterlerin içerisinden en iyi tercihin seçilmesine imkân sağlayan bir araçtır. Rasyonel bir karar verme çevresinden iyi tercih edilmiş seçim, genellikle kısıtlar ve yönetimin amacı doğrultusunda sınırlandırılır. Burada adı geçen kısıt, amaçların başarı ile yerine getirilmesi ve seçilmesidir [32].

ÇKKV yöntemleri, 1960'lı yıllarda, karar verme işlerine yardımcı olacak bir takım araçların gerekli görülmesiyle geliştirilmeye başlanmıştır. Seçimde ulaşılmak istenen hedefi birçok parametrenin belirlediği ve seçim için değerlendirilecek alternatiflerin

her birinin kendine has avantajlarının bulunduğu durumlarda karar verme işi çok zor bir durum alacaktır. Böyle durumlarda kararı verecek olan kişi ya tüm bu kararsızlık sıkıntısından kurtulmak için, sağlıklı olup olmadığını önemsemeden, bir karara varacak; ya da uzun ve rasyonel olmayan analizler sonunda kuşku içerisinde bir

karara varacaktır. ÇKKV yöntemlerini kullanmaktaki amaç alternatif ve parametre (kriter) sayılarının fazla olduğu durumlarda karar verme mekanizmasını kontrol altında tutabilmek ve karar sonucunu mümkün olduğu kadar kolay ve çabuk elde etmektir [33].

ÇKKV metotlarını diğer bir sınıflandırma yöntemi ise karar prosesindeki karar verici sayısına göre olanıdır. Bu nedenle tek bir karar vericinin bulunduğu ÇKKV metotları ve çok karar vericinin bulunduğu ÇKKV metotları olarak da sınıflandırma yapılabilir [34].

ÇKKV problemleri; Çok Nitelikli Karar Verme (ÇNKV) ve Çok Amaçlı Karar Verme (ÇAKV) olarak sınıflandırılmaktadır. ÇNKV problemleri önceden belirlenen sayıda alternatife sahiptir ve bu alternatiflerin her birine ilişkin ulaşılacak başarı düzeyleri belirlenmektedir. ÇNKV problemlerinde kararlar, her bir alternatif için var olan niteliklerin karşılaştırılması yolu ile verilmektedir. Öte yandan ÇAKV problemlerinde ise, alternatiflerin sayısı önceden belirlenememektedir ve modelin amacı "en iyi" alternatifi belirlemektir. Kantitatif karar verme tekniklerinde optimal çözümü verecek olan alternatiflerin sayısına önceden karar verilememektedir.

Bu nedenle işletme sorunlarının çözümünde kullanılacak olan optimizasyon tekniğinin ÇAKV metotları arasından seçilmesi gerekmektedir [35].

ÇAKV probleminin çözümü sırasında farklı şekillerde ele alınan birçok yöntem geliştirilmiştir [36].

- Karar vericiden açıkça bilgi istemeyen yöntemler,
- Karar vericiden başlangıçta bilgi isteyen yöntemler,
- Karar vericiden karar esnasında ardışık olarak bilgi isteyen yöntemler,
- Karar vericiden bilgiyi sonradan isteyen yöntemler gibi.

ÇNKV metotlarının sınıflandırılması çizelge 3.1'de gösterilmiştir.

Çizelge 3.1 : ÇNKV metotlarının sınıflandırılması [34].

	Karar Vericiden Gelen Bilgi	**Bilginin Önem Durumu**	**Metodun Temel Sınıfı**
ÇNKV	Bilgi Yok		Dominant
			Maxmin
			Minmax
	Niteliklere Ait Bilgi	Standart Seviye	Birleşik
			Birleşik Olmayan
		Ordinal	Permütasyon ile
			Lexicographic
			Eliminasyon
		Kardinal	Doğrusal Atama
			SAW
			AHP
			ELECTRE
			TOPSIS
		İkamenin Marjinal Oranı	Hiyerarşik Değişim
	Alternatiflere Ait Bilgi	Tercihler	LINMAP
			İnteraktif SAW
		Yakınlık Sıralanması	MDS

ÇKKV modelleri ile karar analizi aşamasında aşağıdaki dört aşamalı genel süreç izlenir:

- Kriter ve alternatiflerin belirlenmesi,
- Kriterlerin göreli önemini gösteren nümerik ölçütler atanması,
- Her bir kritere göre alternatiflere nümerik ölçütler atanması,
- Alternatifleri sıralamak için nümerik değerlerle işlemler yapılması.

ÇKKV yöntemleri bu sürecin dördüncü aşamasını etkin olarak gerçekleştirmek üzere geliştirilmişlerdir. ÇKKV yöntemleri;

- Ağırlıklı toplam yöntemi,
- Ağırlıklı çarpım yöntemi,
- Analitik ağ süreci yöntemi,
- Analitik Hiyerarşi Prosesi yöntemi,
- Uyum-uyumsuzluk yöntemi (ELECTRE),
- Uzlaşma yöntemi (TOPSIS) olarak sayılabilir [37].

3.5.1 Ağırlıklı toplam yöntemi

Bu teknikte; her alternatif, her kritere göre ayrı ayrı puanlanır. Ardından her kritere, o kriterin diğer kriterlere göre önemini gösteren ağırlıklar verilir. Daha sonra tüm alternatifler için ağırlıklı ortalama puan hesaplanır [37].

$$A_i = \sum_{j=1}^{n} a_{ij} w_j \quad \textbf{(3.1)}$$

A_i = i alternatifinin ağırlıklı toplam skoru

a_{ij} = i alternatifinin j kriterine göre puanı

w_j = j kriterinin ağırlığı

En yüksek puanı toplayan alternatif, karar olarak belirlenir. Yöntem basittir, ancak kriterlerin göreli önemini gösteren ağırlıklar karar verici tarafından subjektif olarak verildiğinden, bu ağırlıkların tahmininde yapılan hatalar sonuçları tamamen değiştirebilir.

3.5.2 Ağırlıklı çarpım yöntemi

Bu yöntemde her alternatif diğer alternatiflerle, her kriter için değerlerin oranları çarpılarak karşılaştırılır. Değerlerin oranları, karşılık gelen kriterin ağırlığı üs olarak yer alır. A_K ve A_L alternatiflerini karsılaştırmak için aşağıdaki çarpım işlemi yapılır [38].

$$R(A_K / A_L) = \prod_{j=1}^{N} (a_{Kj} / a_{Lj})^{w_j} \quad \textbf{(3.2)}$$

$R(A_K / A_L)$ = K alternatifinin L alternatifine göre ağırlıklı çarpım yöntemi ile elde edilen skoru

A_{Kj} / A_{Lj} = K alternatifinin j kriterine göre puanının, L alternatifinin j kriterine göre puanına oranı

w_j = j kriterinin ağırlığı

$R(A_K / A_L)$ ifadesinin birden büyük çıkması K alternatifinin L alternatifine göre tercih edileceği anlamını taşımaktadır. Dolayısıyla en iyi alternatif diğer tüm alternatiflere göre daha iyi ya da eşit orana sahip olandır.

3.5.3 Analitik ağ prosesi yöntemi

Analitik Ağ Prosesi (AAP) AHP yönteminin uzantısı olarak Saaty tarafından geliştirilmiş çok ölçütlü bir karar verme yöntemidir. AAP, karar verme sürecinde faktörler arasındaki ilişkilerin dikkate alınmasını sağlamakta ve problemi tek bir yöne bağlı kalarak modelleme zorunluluğunu ortadan kaldırmaktadır. Aynı zamanda karar seviyeleri ve özellikler arasında daha kompleks ilişkilerin dikkate alınmasını sağlar [30].

Karar verme sürecinde faktörler arasındaki ilişkileri dikkate alan ve problemin tek bir yöne bağlı kalarak modelleme zorunluluğunu ortadan kaldıran yöntem Thomas L. Saaty tarafından geliştirilen Analitik Ağ Prosesi (AAP) yöntemidir.

AAP yönteminde karar verme problemi bir ağ yapısı ile modellenmekte ve modelleme aşamasında faktörler arasındaki bağımlılıklar ve faktör içindeki iç bağımlılıklar dikkate alınmaktadır. AHP hiyerarşik ilişkileri tek yönlü bir iskelet ile gösterirken, AAP karar seviyeleri ve özellikler arasında daha karmaşık ilişkilerin dikkate alınmasını sağlar. Bu şekilde hiyerarşik yapılar ile modellenemeyen karmaşık problemlerin kolay bir şekilde modellenmesini sağlar.

AAP ile karar problemlerinin çözümü dört ana adımın uygulanmasıyla yapılır [30].

Adım 1: Problemin Tanımlanması ve Modelin Kurulması: Bu aşamada karar verme problemi açık bir şekilde tanımlanmalı ve ağ seklinde rasyonel bir biçimde ayrıştırılmalıdır. Bu yapı beyin fırtınası ya da diğer ayırma metotları vasıtasıyla karar vericilerin fikirlerinden yararlanılarak elde edilebilir.

Adım 2: İkili Karşılaştırma Matrisleri ve Öncelik Vektörleri: AAP'de de, AHP'de olduğu gibi her kararı etkileyen faktörler ikili karşılaştırmalara tabi tutulur ve böylece faktörlerin önem ağırlıkları belirlenir. Karar vericiler ikili karşılaştırmalarda seri şekilde bir takım sorulara cevap vererek iki faktörü aynı anda karşılaştırır ve bunların hedefe olan katkılarının nasıl olduğunu belirler.

AAP'de ikili karşılaştırma matrislerinin oluşturulması ve nispi önem ağırlıklarının belirlenmesinde AHP'de olduğu gibi Saaty tarafından önerilen 1-9 önem skalası kullanılır.

AHP'de olduğu gibi AAP'de de ikili karşılaştırmalar bir matris çatısı altında yapılır ve lokal öncelik vektörü $A_w=\lambda_{enb}w$ denkleminin çözülmesi ile elde edilen özvektör

ile belirlenir. Burada A ikili karşılaştırma matrisi, w özvektör, λ_{enb} ise A'nın en büyük özdeğeridir. Saaty, w'nin yaklaşık çözümü için normalleştirme algoritmasını önermiştir.

Adım 3: Süpermatris Oluşumu: Süpermatrisin genel yapısı markov zinciri prosesine benzerdir. Birbirine bağımlı etkilerin bulunduğu bir sistemde global önceliklerin elde edilmesi için, lokal öncelik vektörleri süper matris olarak bilinen bir matrisin kolonlarına tahsis edilerek yazılır. Sonuç olarak bir süper matris gerçekte parçalı bir matristir ve buradaki her bir matris bölümü bir sistem içindeki iki faktör arasındaki ilişkiyi gösterir.

Elementlerin birbiri üzerindeki uzun dönemli nispi etkileri süper matrisin kuvveti alınarak belirlenir. Önem ağırlıklarının bir noktada eşitlenmesini sağlamak için süper matrisin (2k+1) kuvveti alınır, burada k rastgele seçilmiş büyük bir sayıdır ve elde edilen yeni matris limit süper matris olarak isimlendirilir.

Adım 4: En İyi Alternatifin Seçilmesi: Limit süper matris ile alternatiflere veya karşılaştırılan faktörlere ilişkin önem ağırlıkları belirlenmiş olur. Seçim probleminde en yüksek önem ağırlığına sahip olan alternatif en iyi alternatif, ağırlıklandırma probleminde ise en yüksek önem ağırlığına sahip olan faktör karar sürecini etkileyen en önemli faktördür.

3.5.4 Analitik hiyerarşi prosesi yöntemi

Günümüzde çoğu insan, ya kişisel değerlendirmeye dayalı yargılarla ya da geçerliliği yetersiz sonuçlara ulaşan ve doğruluğu kanıtlanamayan matematik modelleri kullanarak karar vermektedir.

Bu durumda ihtiyaç duyulan, karmaşık problemi basitleştirip, daha kolay anlaşılabilecek hale getiren ve problemi oluşturan bileşenler arasındaki ilişkiyi gösteren bir karar metodolojisidir. Böyle bir metodoloji de, Analitik Hiyerarşi Prosesinin içinde bulunmaktadır [27].

AHP, karar vericiye, problem için belirlediği her faktör veya kriter için karşılaştırma imkanı verirken, belirlemiş olduğu faktör ve kriterleri ardı ardına gelen seviyelerde bir hiyerarşik yapı içerisinde sıralamasına da olanak sağlamaktadır. AHP'nin uygulanmasında iki temel süreç vardır: hiyerarşik yapının oluşturulması ve sonuçların hesaplanmasıdır [39].

3.5.5 Uyum uyumsuzluk yöntemi (ELECTRE)

ELECTRE (Elimination et Choix Traduisant La Realite - Elimination and Choice Translating Reality) ilk olarak Benayoun, Roy ve arkadaşları tarafından 1966 yılında geliştirilmiştir. Metot var olan karar verme metotlarına bir cevap olarak geliştirilmiştir. Aslında sadece bir çözüm metodu değil kendi içinde çok tartışılan bir felsefedir [40].

ELECTRE yönteminin temelinde, her bir kriter için ayrı ayrı alternatifleri ikili olarak karşılaştırıp üst sıralama ilişkilerini elde etmek yatar. Bir alternatifin diğer alternatife göre sayısal olarak baskın olmadığı durumlar da dâhil olmak üzere karar verici tarafından birinin diğerine tercih edileceği durumlar tanımlanır [34]. ELECTRE metodunun ana konsepti; her bir kriter için ayrı ayrı olmak üzere alternatiflerin aralarındaki ikili karşılaştırmaları kullanmaktır. İki alternatifin (A_i ve A_j) tercih edilebilirliğinin üstünlük ilişkisi $A_i \rightarrow A_j$ şeklinde gösterilir ve eğer i.nci alternatif j.nci alternatife niceliksel baskınlık kuramazsa karar verici, A_i 'nin A_j'ye göre daha iyi olduğu riskini alabilmelidir. Alternatifler, eğer başka bir alternatif bir veya daha fazla kriterde üstün ve kalan diğer kriterlerde eşit olursa baskın olarak adlandırılabilirler.

ELECTRE metodu her bir kriter için alternatiflerin ikili karşılaştırmaları ile başlamaktadır. Alternatiflerin tercih edilebilme üstünlük ilişkisinin ardışık yargıları arasından, A_j alternatifi A_k alternatifine üstünlük sağlar veya daha önemlidir sonucunu destekleyen kanıt sayısı şeklinde tanımlanan uyumluluk indeksini ve uyumluluk indeksinin karşı tarafı olan uyumsuzluk indeksini çıkartmaktadır. Sonuç olarak ELECTRE metodu alternatifler arasında ikili tercih edilebilirliğinin üstünlük ilişkisi sistemini getirmektedir. Bu metot alternatiflerin daha açık bir görüntüsünü daha az favori olanları eleyerek sağlamaktadır. Metot özellikle birkaç kriter fakat çok sayıda alternatif içeren karar problemleri için uygundur [34].

ELECTRE farklı alternatiflerin bütün mümkün çiftleri arasındaki bağıntının sistematik bir analizini içine almaktadır ve her alternatifin hesaplamadaki ortak kriterler kümesi üzerindeki skorlarına dayanmaktadır. Sonuç, bir alternatifin diğeri üzerindeki sıralama dışı bırakma derecesi olarak adlandırılan bir ölçüdür. Örneğin bir A alternatifi B alternatifini sıralama dışı bırakır denilir. Dolayısıyla ELECTRE ve benzer temellere dayalı diğer yöntemler, aynı zamanda, sıralama dışı bırakma

yöntemleri olarak da adlandırılır [41]. ELECTRE yönteminin I, II, III, IV, Tri ve 1S olmak üzere altı ana versiyonu vardır.

ELECTRE 7 aşamadan oluşmaktadır [34].

- Karar Matrisinin Normalize Edilmesi
- Normalize Karar Matrisinin Ağırlıklandırılması
- Uyumluluk ve Uyumsuzluk Setinin Belirlenmesi
- Uyumluluk ve Uyumsuzluk Matrislerinin Oluşturulması
- Uyumluluk ve Uyumsuzluk Üstünlük Matrislerinin Belirlenmesi
- Toplam Üstünlük Matrisinin Belirlenmesi
- Daha Az Uygun Alternatiflerin Elenmesi

ELECTRE'nin diğer yöntemlere göre önemli üstün yanları bulunmaktadır. Kalitatif ve kantitatif verinin karışık olarak değerlendirilmesine olanak tanıyan kuvvetli ve aynı zamanda kolayca uyum sağlayabilen bir yöntemdir. Birçok durumda, baskın olmayan alternatifler alt kümesi verilebilir ve alternatiflerin güçlü ve kesin bir ön sıralamasını vermeyebilir fakat mevcut verilerin kalitesine göre, alternatif seçimi problemi için daha gerçekçi bir çözüm verebilir. Aynı zamanda, diğer yöntemlerin daha yüksek düzeyde zaman ve insan gücü kaynağı gerektiren ayrıntılı veri gereksinimi, bu yöntemlerin planlama aşamasındaki bir mühendislik projesinin değerlendirilmesinde kullanılmalarını engeller. ELECTRE böyle yüksek düzeyde kaynaklara gereksinim duymamaktadır [41].

3.5.6 Uzlaşma yöntemi (TOPSIS)

TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) Yoon ve Hwang tarafından 1980 yılında ELECTRE metoduna bir alternatif olarak geliştirilmiştir ve kabul edilmiş varyasyonlar içinde en yaygınlarından biri olarak düşünülebilmektedir. Metodun temel konsepti; seçilmiş alternatif bir nevi geometrik anlamda ideal çözüme en kısa mesafede ve negatif-ideal çözümden en uzak mesafede olmalıdır. TOPSIS metodu her bir kriterin tekdüze bir şekilde artan ya da azalan fayda eğilimine sahip olduğunu varsaymaktadır. Bundan dolayı, ideal ve negatif-ideal çözümleri tanımlamak kolaydır. Öklid mesafesi yaklaşımı alternatiflerin ideal çözüme göreli yakınlıklarını değerlendirmeyi amaçlamaktadır.

Böylece bu göreli mesafelerin karşılaştırılmalarının bir serisi aracılığıyla alternatiflerin tercih sırası çıkarılabilmektedir [34]. Daha sonraları bu düşünce Zeleny (1982) ve Hall (1989) tarafından da uygulanmış, ve Yoon (1987) ve Hwang, Lai ve Liu (1993) tarafından geliştirilmiştir [51].

TOPSIS metodu, n kriter için değerlendirilen m alternatifi kapsayan aşağıdaki karar matrisini değerlendirmektedir.

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Burada xij; i.nci alternatifin j.nci kriter için performans değerini göstermektedir. Karar matrisinin satırlarında üstünlükleri sıralanmak istenen karar noktaları, sütunlarında ise karar vermede kullanılacak değerlendirme faktörleri yer alır. A matrisi karar verici tarafından oluşturulan başlangıç matrisidir. Karar matrisi aşağıdaki gibi gösterilir:

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

A matrisinde m karar noktası sayısını, n değerlendirme faktörü sayısını verir.

TOPSIS aşağıda gösterilen 6 aşamadan oluşmaktadır [34].

- Normalize Karar Matrisinin Oluşturulması
- Ağırlıklandırılmış Normalize Karar Matrisinin Oluşturulması
- İdeal ve Negatif-İdeal Çözümlerin Belirlenmesi
- Ayırma Ölçümünün Hesaplanması
- İdeal Çözüme Göreli Yakınlığın Hesaplanması
- Tercih Sırasının Düzenlenmesi

TOPSIS metodu ELECTRE'nin temeli üzerine geliştirilmiştir. Bu nedenle metotların ilk iki aşamalarının aynı olması şaşırtıcı değildir. Hem ELECTRE hem de TOPSIS ölçeklerin karşılaştırılabilmesi için bir normalize karar matrisi temeli ile başlamaktadır. İkinci aşamada da her iki metot da tercihlerin ağırlık değerlerini karar

vericilerden almaktadırlar. Üçüncü aşamada metotlar farklılaşmaktadır. Bir fark olarak ise ELECTRE alternatiflerden birinin diğerine olan üstünlüğüne göre elemeleri yaparken, TOPSIS ideal çözüme en yakın, negatif ideal çözüme en uzak alternatifin en iyi alternatif olduğunu göstermektedir [43]. TOPSIS ‘in bir avantajı her bir alternatifin kendi değerini almasıdır.

3.6 Uygulamada Kullanılacak ÇKKV Yöntemi

Bu bölümde çok kriterli karar verme yöntemleri tanıtılmıştır. Problem olarak belirlenen Yazılım geliştirme metodolojisi seçiminde ÇKKV yöntemlerinden Analitik Hiyerarşi Prosesi kullanılacaktır. AHP yönteminin seçilme nedeni; hem objektif, hem de subjektif değerlendirme ölçütlerini kullanması, değerlendirmelerin tutarlılığının test edilmesini sağlaması, özellikle de¸ çok sayıdaki ölçüte göre değerlendirilmesi gereken alternatifler içerisinden hangisine öncelik verilmesi gerektiği gibi¸ çok önemli bir kararın karar verici tarafından uygulanmasıdır. Ayrıca kullanıcılardan alınacak bilgilerin değerlendirilebilir ve ölçülebilir veriler içerecek olması AHP’nin rahatlıkla uygulanabilir olmasını sağlayacaktır. En sık kullanılan ve üzerine en çok makale ve tez yazılan ÇKKV yöntemlerinden biri olması da AHP’nin seçilmesinde önemli bir faktördür. Yöntemin ana avantajı ise; anlaşılması diğer yöntemlere göre daha kolay olup gereksiz matematiksel işlemler içermemesi ve çok yönlü kriterlerin kolaylıkla yönetilebiliyor olmasıdır. Sonraki bölümde Analitik Hiyerarşi Prosesi yöntemi ayrıntılı olarak anlatılacaktır.

4. ANALİTİK HİYERARŞİ PROSESİ YÖNTEMİ

1970'lerde Profesör Thomas L. Saaty tarafından geliştirilen Analitik Hiyerarşi Proses Yöntemi (AHP), birden çok kriter içeren karmaşık problemlerin çözümünde kullanılan bir karar verme yöntemidir. AHP; kişileri nasıl karar vermeleri gerektiği konusunda bir yöntem kullanmaya zorunlu kılmak yerine, onlara kendi karar verme sistemlerini tanıma imkânı sağlayarak, daha iyi karar verilmesini sağlayan bir karar verme modelidir [44]. AHP karar vericilerin karmaşık problemleri, problemin ana hedefi, kriterleri, alt kriterler ve alternatifleri arasındaki ilişkiyi gösteren hiyerarşik yapıda modellemelere olanak verir. AHP'nin en önemli özelliği karar vericinin hem objektif hem de subjektif düşüncelerini karar sürecine dâhil edebilmesidir. Bir diğer ifade ile AHP, bilginin, deneyimin, bireyin düşüncelerinin ve önsezilerinin mantıksal bir şekilde birleştirildiği bir yöntemdir. AHP'de karar vericinin amacı doğrultusunda faktörlerin ve faktörlere ait olan alt faktörlerin belirlenmesi ilk adımdır. AHP'de öncelikle amaç belirlenir ve bu amaç doğrultusunda amacı etkileyen faktörler saptanmaya çalışılır [45].

AHP yargılar ve kişisel değerleri mantıksal bir düzende birleştirmektedir. Proses; hayal gücü, deneyim ve bilgi ile bir problemin hiyerarşisini oluşturmaya ve mantık, önsezi ve deneyim ile yargıları yapmaya bağlıdır. Başta kabul edilip devam edildikten sonra AHP, problemin bir parçasının elemanlarının diğer parçalarla birleştirilmiş sonuca ulaşmak için nasıl bağlanacağını göstermektedir. Sistemin bir bütün olarak bağlantılarını tanımlama, anlama ve yargılamak için kullanılan bir prosestir [46].

Amaç, faktör ve alt faktörler belirlendikten sonra, faktör ve alt faktörlerin kendi aralarındaki önem derecelerinin belirlenmesi için ikili karşılaştırma karar matrisleri oluşturulur. Bu matrislerin oluşturulmasında Saaty tarafından önerilen 1-9 önem skalası kullanılır.

Yapılan çalışma sonunda verilecek karar birçok kişiyi etkileyecek yapıda ise ikili karşılaştırma karar matrisleri farklı kişilerin yargılarının birleştirilmesi ile oluşturur. Bu birleştirme işleminde birçok araştırmacı, tutarlı ikili karşılaştırma matrisleri elde

edebilmek için, geometrik ortalama yönteminin kullanılmasını önermektedir.

Saaty tarafından önerilen 1-9 önem skalası en iyi sonuçların elde edilmesini sağlamaktadır.

AHP' nin teorik alt yapısı üç aksiyoma dayanır. Bu aksiyomlardan ilki, iki taraflı olma/tersi olma aksiyomudur. Sözel olarak ise, örneğin, "A elemanı B elemanının 5 katı büyüklüğünde ise B, A 'nın 5'te 1'idir" denir. İkincisi, homojenlik aksiyomudur ve karşılaştırılan elemanların birbirinden çok fazla farklı olmaması gerektiğini, olursa yargılarda hataların ortaya çıkabileceğini ifade etmektedir. Üçüncü aksiyom bağımsız olma aksiyomudur ve bir hiyerarşideki belirli bir kademeye ait elemanlara ilişkin yargıların veya önceliklerin başka bir kademedeki elemanlardan bağımsız olmasını gerektirir. Bu ifade, üst kademe kriterlerin önceliklerinin yeni bir alternatif eklendiğinde veya çıkarıldığında değişmeyeceği anlamına gelmektedir [47].

4.1 AHP Aşamaları

AHP kullanılan bir karar verme sürecinde aşağıda belirtilen adımlar izlenmelidir [48].

• Problemin anahtar elemanlarını ve bu elemanlar arasındaki ilişkileri temsil eden bir model inşa edilmelidir.

• Duygu ve bilgileri yansıtan yargılar ortaya çıkarılmalıdır.

• Bu yargılar anlamlı rakamlar ile temsil edilmelidir.

•Bu rakamlar hiyerarşideki elemanların önceliklendirilmesinde kullanılmalıdır.

• Bu sonuçlar kapsamlı bir çıktıya ulaşmak için sentezlenmelidir.

• Yargılardaki değişikliklerin duyarlılığı analiz edilmelidir.

Yine Thomas L. SAATY tarafından yazılan bir makalede AHP'nin 7 temel noktası aşağıdaki şekilde tanımlanmıştır [49].

• Oran ölçekleri, orantılılık ve normalize edilmiş oran ölçekleri,

• Karşılıklı ikili karşılaştırmalar,

• Temel doğru özvektörün duyarlılığı,

• Homojenlik ve kümelenme,

• Bağımlılık ve geri besleme ile genişletilebilecek sentezler,

• Derece koruması ve ters dönme,

• Grup yargılamaları.

Çok ölçütlü karar problemlerinin AHP ile modellenmesinde aşağıdaki aşamalar gerçekleştirilir:

• Problemin tanımlanması (kriterlerin belirlenmesi, alternatiflerin ortaya konulması, hiyerarşik diyagramın çizilmesi),

• Kriter ağırlıklarının belirlenmesi,

• Alternatiflerin her kritere göre puanlanması,

• Her alternatifin çok kriterli puanının elde edilmesi,

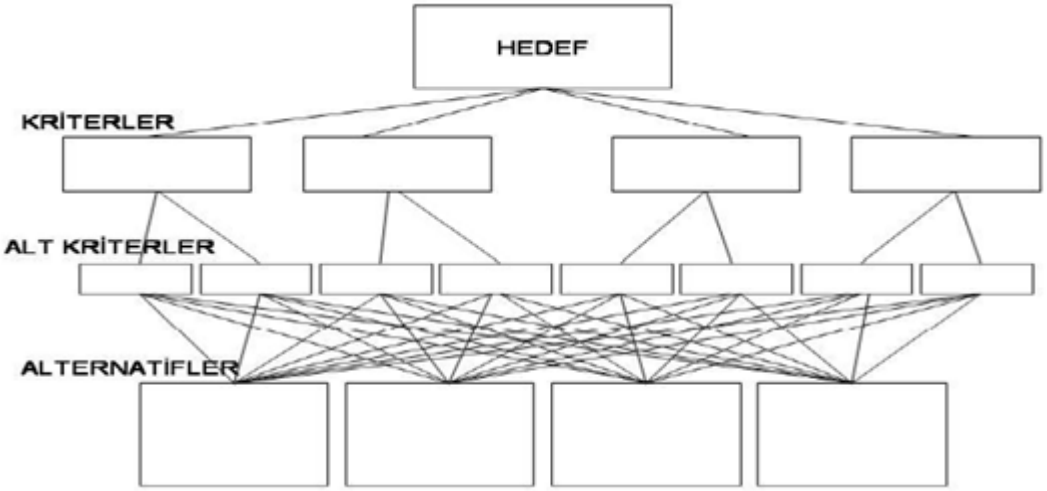
•Genel puanların karşılaştırılması ve en iyi alternatifin seçilmesi [37].

4.1.1 Kriterlerin hiyerarşik yapısının oluşturulması

Aynı anda dikkate alınması zor olan ve çok sayıda ortak özellikleri bulunan birçok varlığı içine alan sistemlerin incelenmesi, bu sistemleri alt sistemlere bölerek kolaylaşır. Tanımlanan varlıkların, belli bir grubuna dâhil olanların yalnızca tek bir değer gruba dâhil olanları etkilediği ve yalnızca bir tek grubun varlıkları tarafından etkilendiği ayrık kümelere ayrılabildiği varsayımına dayanarak oluşturulan, her biri çeşitli sayıda eleman ya da faktör bulunduran sıralı düzeylerden oluşan sisteme "Hiyerarşi" denir [50].

Bu aşama, aslında diğer bütün karar verme yöntemlerine ait problemlerinin çözümünde de kullanılan ilk aşamadır. Hiyerarşik yapının oluşturulması öncesinde problemin tanımlanması sırasında dikkat edilmesi gereken en önemli husus, bu problemin AHP yöntemine uygun olup olmadığı, diğer bir deyişle, elemanların kantitatif göstergeleri bulunup bulunmadığıdır. Bunun nedeni, AHP yönteminin öznel değerlendirmeler için bir ölçü birimi yaratmasıdır [51]. Hiyerarşiler oluşturulurken, problemi olabildiğince temsil edebilecek, konu ile ilgili yeterince ayrıntı belirlenmelidir fakat hedef ve kriterleri değiştirerek sonuçların duyarlılığını kaybetmemek gerekir. Problemi çevreleyen çevre düşünülmelidir. Çözüme katkı sağlayacağına inanılan önemli nokta ve nitelikler tanımlanmalıdır. Problemle alakalı katılımcılar tanımlanmalıdır.

AHP'nin en önemli özelliği, karar vericinin karar problemini birbirleri ile hiyerarşik ilişkisi olan elemanlara ayırmasıdır. Bu hiyerarşinin en tepesinde karar vericinin nihai hedefi bulunur. Bir personel seçimi kararında bu hedef "ise en uygun adayın belirlenmesi" olarak ifade edilebilir. Hiyerarşinin daha alt seviyelerde bu nihai hedefe ulaşmak için göz önüne alınması gereken kriterler sıralanır. Hiyerarşide aşağıya doğru inildikçe kriterlerin belirginliği artar. Hiyerarşinin en alt seviyesinde ise alternatifler yer alır. Yöntemin birinci aşamasında oluşturulan hiyerarşik yapı Şekil 4.1'deki gibi gösterilebilir [52].



Şekil 4.1 : AHP hiyerarşik yapı.

Burada hedef; tam olarak karar vericinin ihtiyacı ya da isteği olarak tanımlanabilir. Kriterler; değerlendirme yapmak için etkinliğin ve esasın ölçüsü, alt kriterler ana kriterin alt bileşenleridir. Alternatifler ise değerlendirilmeye tabi tutulacak karar vericiye uygun olan değişik davranış türleri ya da nesnelerdir.

4.1.2 Kriterler ve alternatifler arasında ikili karşılaştırmalar yapılması

Bu aşamada ikili karşılaştırmalar yapılır. İkili karşılaştırmalar; genelde karşılaştırılan elemanların bazı niteliklere uymalarına göre insanların tercih edilme, hoşlanma ya da önem sıralarına olan duyarlılıklarını açıklayabilen doğal bir süreçtir [53].

Bu ikili karşılaştırmalar bir kare matriste gösterilir. Her aşamada, matriste en soldaki sütundaki bir elemanın en üst satırdaki bir elemana göre üstünlüğünü gösterir. Karşılaştırmalar şu iki sorunun cevabını yansıtır. "Bir üst seviyedeki kritere göre bu iki elemandan hangisi daha önemlidir?", "Bu önemin derecesi nedir?".

AHP'de ikili karşılaştırma yargılarını sayısal değerlere dönüştürebilmek için Saaty

ve diğerleri 1-9 ölçeğini geliştirmişlerdir. Bu temel ölçek geliştirilirken, özvektörleri bilinen üç ayrı problem için özniteliklere ya da nitel farklara (eşit, zayıf, kuvvetli, çok kuvvetli ve mutlak) karşılık gelen sayısal değerleri farklı olan 27 ölçek kullanmışlardır. Her bir problemin ikili karşılaştırmalar matrisindeki yargılar her bir ölçekten farklı değer almaktadır.

Her bir problemde her bir ölçek için ikili karşılaştırma matrisleri elde edilmiş ve bir problem için 27 özvektör bulunmuştur. Her bir problemde her bir ölçek için bulunan özvektörlerle problemlerin önceden bilinen özvektörleri için sapmaların kareler ortalamasının kökü ve ortanca etrafında ortancadan mutlak sapma değerleri hesaplanmıştır. Bu çalışma sonucunda 1-9 ölçeğinin diğerlerinden daha üstün olduğu sonucuna ulaşılmıştır [54]. Bir ölçekte üst sınırın sonsuz olmasının, kişilerin ikili karşılaştırmaları yaparken ayrım yapma yeteneklerinin sınırlanmasına neden olduğu bilinmektedir. Bu nedenle kullanılan ölçek değerlerinin ve üst sınırın kişilerin karşılaştırma yeteneklerini sınırlamayacak şekilde olması gerekmektedir. 1-9 ölçeği hem bunu vermesi hem de kullanımının kolay olması bakımından temel ölçek olarak seçilmiştir [54].

Çizelge 4.1 : 1-9 ölçeği.

Önem Derecesi	Tanım	Açıklama
1	Eşit önemli	İki seçenekte eşit derecede öneme sahip
3	Orta derecede önemli	Tecrübe ve yargı bir kriteri diğerine karşı biraz üstün kılmakta
5	Kuvvetli derecede önemli	Tecrübe ve yargı bir kriteri diğerine karşı oldukça üstün kılmakta
7	Çok kuvvetli derecede önemli	Bir kriter diğerine göre üstün sayılmıştır
9	Kesin önemli	Bir kriterin diğerinden üstün olduğunu gösteren kanıt çok büyük güvenilirliğe sahiptir
2, 4, 6, 8	Ara değerler	Uzlaşma gerektiğinde kullanılmak üzere iki ardışık yargı arasındaki değerler

AHP çözümlenirken Çizelge 4.1' de görüldüğü gibi üst sınır 9 ile sınırlandırılmıştır.

Rakamları değerlendirmek için çoğu kez kullanılan pratik bir yöntem, hislerimizi üç kategoride sınıflandırmaktır. Bunlar yüksek, orta ve düşük seviyeleridir. Daha detaylı bir sınıflandırma için ise bu kategorilerin her biri tekrar kendi içinde yüksek, orta ve düşük sınıflamasına tabi tutulur. Buradan da anlaşılır ki anlam farklılıklarını her

zaman 9 değişik tür ifade etmektedir. Bu nedenle 9 rakamının üstüne çıkılmaması gerekmektedir. Saaty'nin geliştirdiği bu metot n<10 kriter için en iyi sonuçları vermektedir. Bir matrisin elemanları çok büyük sayılardan oluşuyorsa, bu durum daha büyük tutarsızlıklar meydana getirebilir.

Faktörler arası karşılaştırma matrisi, nxn boyutlu bir kare matristir. Bu matrisin köşegeni üzerindeki matris bileşenleri 1 değerini alır. Bunun nedeni, köşegen üzerindeki elemanlar, her elemanın kendisi ile kıyaslanmasıdır. Karşılaştırma matrisi aşağıda gösterilmiştir.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \cdot & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Karşılaştırma matrisinin köşegeni üzerindeki bileşenler, yani i = j olduğunda, 1 değerini alır. Çünkü bu durumda ilgili faktör kendisi ile karşılaştırılmaktadır. Faktörlerin karşılaştırılması, birbirlerine göre sahip oldukları önem değerlerine göre birebir ve karşılıklı yapılır. Faktörlerin birebir karşılıklı karşılaştırılmasında daha önce verilen önem skalası (Çizelge 4.1) kullanılır.

Karşılaştırmalar, karşılaştırma matrisinin tüm değerleri 1 olan köşegeninin üstünde kalan değerler için yapılır. Köşegenin altında kalan bileşenler için ise doğal olarak aşağıdaki denklemi kullanmak yeterli olacaktır.

$$a_{ji} = \frac{1}{a_{ij}} \qquad \textbf{(4.1)}$$

Yukarıda verilen örnek dikkate alınırsa karşılaştırma matrisinin birinci satır üçüncü sütun bileşeni (i=1, j=3) 3 değerini alıyorsa, karşılaştırma matrisinin üçüncü satır birinci sütun bileşeni (i=3,j=1), 1/3 değerini alacaktır.

Ölçeğe göre oluşturulan ve aşağıda gösterilen N matrisinin her bir sütununun, sütunun ilk değerine bölünmesiyle elde edilen n satır ve sütunlu A kare matriste n(n-1)/2 adet ikili karşılaştırma yapılır. Matristeki elemanların yarısı diğer elemanların tersidir. $w_1/w_2=3$ ise, $w_2/w_1=1/3$ olur [52].

İkili karşılaştırma, i satırındaki (i=1,2,3,...,n) kriterlerin n sütunla temsil edilen her bir kritere bağlı olarak derecelenmesiyle yapılır. a_{ij} ($a_{ij} = w_i / w_j$), A'nın (i,j) elemanını tanımladığında, AHP, 1 ile 9 arasında bir ölçek önerir, burada aij = 1, i ve

j'nin eşit önemde olduğunu, $a_{ij} = 5$, i'nin j'den kuvvetli düzeyde önemli olduğunu, $a_{ij} = 9$ ise, i'nin j'den kesinlikle çok önemli olduğunu yansıtır. 1 ile 9 arasındaki diğer değerler ara değerler olarak yorumlanır [55].

Temel ölçek olarak AHP'de 1-9 ölçeği kullanıldığı için A matrisinin öğeleri daima pozitif ve kare matris olacaktır .

$$a_{ij}>0 \text{ , } i,j =1,2,\ldots,n$$

İkili karşılaştırma matrisi veya yargı matrisi eğer tam tutarlı ise aşağıdaki eşitliği sağlar:

$$a_{ij} \text{ . } a_{jk} = a_{ik} \quad i,j,k =1,2,\ldots,n \qquad \textbf{(4.2)}$$

$$a_{ij} \text{ . } a_{jk} = (w_i / w_j).(w_j / w_k)= w_i / w_k = a_{ik} \quad i,j,k =1,2,\ldots,n \qquad \textbf{(4.3)}$$

Tam tutarlılığın göreceli karşılaştırmalarda elde edilmesi oldukça zordur. Bu nedenle AHP'de ağırlıkların veya öncelik vektörünün hesaplanmasında bazı farklı yöntemler kullanılmaktadır.

Eğer A matrisi tam tutarlı ise öncelik veya ağırlık vektörlerini elde etmek oldukça kolaylaşmaktadır, ayrıca herhangi bir satırından matrisin diğer tüm öğeleri kolaylıkla elde edilebilir. Toplam olarak C(n,2) kadar karşılaştırma yapılır [56].

Matrisin en büyük özdeğerine karşılık gelen özvektör matrisi AHP'de ağırlık veya öncelik vektörü olarak adlandırılır. A matrisinin köşegen değerleri 1'e eşittir [56].

4.1.3 Faktörlerin yüzde önem dağılımları

Karşılaştırma matrisi, faktörlerin birbirlerine göre önem seviyelerini belirli bir mantık içerisinde gösterir. Ancak bu faktörlerin bütün içerisindeki ağırlıklarını, diğer bir deyişle yüzde önem dağılımlarını belirlemek için, karşılaştırma matrisini oluşturan sütun vektörlerinden yararlanılır ve n adet-n bileşenli B sütun vektörü oluşturulur.

Aşağıda bu vektör gösterilmiştir:

$$B_i=\begin{bmatrix} b_{11} \\ b_{21} \\ \cdot \\ \cdot \\ b_{n1} \end{bmatrix}$$

B sütun vektörlerinin hesaplanmasında aşağıdaki denklemden yararlanılır.

$$b_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \quad \textbf{(4.4)}$$

Örneğin; değerlendirme faktörlerinin birbirleriyle karşılaştırılmalarını gösteren A karşılaştırma matrisi aşağıdaki gibi tanımlanmışsa ve B1 vektörü hesaplanmak isteniyorsa,

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{1/3} & \mathbf{5} \\ \mathbf{3} & \mathbf{1} & \mathbf{4} \\ \mathbf{1/5} & \mathbf{1/4} & \mathbf{1} \end{bmatrix}$$

bu durumda B_1 vektörünün b_{11} elemanı $\boldsymbol{b_{11}} = \frac{\mathbf{1}}{\mathbf{1+3+0,2}}$ olarak hesaplanacaktır.

Benzer şekilde B_1 vektörünün diğer elemanları hesaplandığında, vektör aşağıdaki gibi elde edilebilir ve sütun vektörünün bileşenleri toplandığında toplamın 1 olduğu görülebilir.

$$B_1 = \begin{bmatrix} 0{,}238 \\ 0{,}714 \\ 0{,}048 \end{bmatrix}$$

Yukarıda anlatılan adımlar diğer değerlendirme faktörleri içinde tekrarlandığında faktör sayısı kadar B sütun vektörü elde edilecektir. n adet B sütun vektörü, bir matris formatında bir araya getirildiğinde ise aşağıda gösterilen C matrisi oluşturulacaktır.

$$\mathrm{C} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \cdot & c_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{bmatrix}$$

Yukarıdaki örnek göz önüne alındığında C matrisi aşağıdaki gibi oluşur.

$$\mathrm{C} = \begin{bmatrix} 0{,}238 & 0{,}210 & 0{,}500 \\ 0{,}714 & 0{,}632 & 0{,}400 \\ 0{,}048 & 0{,}158 & 0{,}100 \end{bmatrix}$$

C matrisinden yararlanarak, faktörlerin birbirlerine göre önem değerlerini gösteren yüzde önem dağılımları elde edilebilir. Bunun için denkleminde gösterildiği gibi C matrisini oluşturan satır bileşenlerinin aritmetik ortalaması alınır ve Öncelik Vektörü olarak adlandırılan W sütun vektörü elde edilir.

$$w_i = \frac{\sum_{j=1}^{n} c_{ij}}{n} \quad \textbf{(4.5)}$$

W vektörü aşağıda gösterilmiştir.

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_n \end{bmatrix}$$

Yukarıdaki örnek çözüldüğünde öncelik vektörünün elemanları aşağıdaki gibi hesaplanabilir. Bu durumda her üç faktör birlikte değerlendirildiğinde yaklaşık değerlerle, birinci faktör %32, ikinci faktör %58 ve üçüncü faktör %10 öneme sahip olacaktır.

$$W = \begin{bmatrix} \frac{0{,}238 + 0{,}210 + 0{,}500}{3} \\ \frac{0{,}714 + 0{,}632 + 0{,}400}{3} \\ \frac{0{,}048 + 0{,}158 + 0{,}100}{3} \end{bmatrix} \cong \begin{bmatrix} 0{,}32 \\ 0{,}58 \\ 0{,}10 \end{bmatrix}$$

4.1.4 Ağırlık ya da öncelik vektörünün hesaplanması

İkili karşılaştırma veya yargı matrisi oluşturulduktan sonra izleyen aşama ağırlık ya da öncelik vektörlerinin hesaplanmasıdır. AHP metodolojisine göre karşılaştırma matrisinin özdeğer ve özvektörleri öncelik sırasını belirmeye yardımcı olur. En büyük özdeğere karşılık gelen özvektör öncelikleri belirlemektedir. A matrisinin en büyük özdeğeri λ_{max} olarak ele alınırsa, W öncelik vektörü olmak üzere; $(A-\lambda_{max}I)W = 0$, denklem sisteminin çözümü ile elde edilir. Ancak bu denklem sisteminin özdeğer ve özvektörlerini hesaplamak özellikle büyük boyutlu matrisler (n>5) için karmaşık ve zaman alıcı olabilmektedir [14].

4.1.4.1 Özvektör yöntemi

İkili karşılaştırma matrisinin en büyük özdeğerine (λ_{max}) karşılık gelen özvektör ağırlık ya da öncelik vektörüne karşılık gelmektedir ve bu vektör $w=(w_1,w_2,\ldots,w_n)$ ile gösterilmektedir. Bu vektörün normalleştirilmiş her öğesi öncelik değerinin bir tahminini göstermektedir ve karşılaştırma yapılırken düşülen hataları da içermektedir.

$$a_{ji} = \frac{w_j}{w_i} = \frac{1}{w_i/w_j} = \frac{1}{a_{ij}} \quad \textbf{(4.6)}$$

$$a_{ij}\,\frac{w_j}{w_i} = 1 \quad \textbf{(4.7)}$$

$$\sum_{j=1}^{n} a_{ij}\, w_j\, \frac{1}{w_i} = n \quad i = 1,2,\dots,n \quad \textbf{(4.8)}$$

$$\sum_{j=1}^{n} a_{ij}\, w_j = w_i n \quad i = 1,2,\dots,n \quad \textbf{(4.9)}$$

$$Aw = nw \quad \textbf{(4.10)}$$

Matris kuramında da bu eşitlik w'nun, n özdeğeriyle A'nın özvektörü olan bir özvektör problemini ifade etmektedir [48].

Uygulamada a_{ij} öznel yargılara dayandığından, w_{ij} / w oranından sapmalar göstermektedir.

Bundan dolayı matris kuramında, eğer $\lambda_1, \lambda_2, \dots, \lambda_n$

Ax = λx eşitliğini sağlayan A'nın özdeğerleri ve her i için a_{ii}= 1 ise $\sum_{i=1}^{n} \lambda_i = n$ 'dir.

Bu nedenle *Aw = nw* eşitliği sağlandığında, n değerine sahip özdeğer dışında bütün özdeğerler sıfırdır. Dolayısıyla A matrisi tutarlı olduğunda, A'nın en büyük özdeğeri n olur.

Eğer A pozitif karşılıklı matrisinin a_{ij} girdileri küçük miktarlarda değişiyorsa, özdeğerler de küçük miktarlarda değişmektedir.

A matrisinin köşegen elemanları (a_{ii}) birlerden oluşuyorsa ve A tutarlıysa, a_{ij} 'deki değişikliklerin n'e yakın olan en büyük özdeğerde (λ_{max}) herhangi bir değişikliğe neden olmadığı ve geri kalan özdeğerlerin de sıfıra yakın değerler aldıkları sonucuna ulaşılmıştır [54].

A matrisi ikili karşılaştırma yargılarından oluştuğundan, öncelik vektörünü bulmak için w vektörü bulunmalıdır:

$$Aw = \lambda_{max} w \quad \textbf{(4.11)}$$

Normalleştirilmiş sonuç gerektiğinden $\alpha = \sum_{i=1}^{n} w_i$ hesaplanarak ve w yerine $\frac{1}{\alpha} w$ alınarak yukarıdaki eşitliğin normalleştirilmiş sonucuna ulaşılır. Bu işlem, tekliği ve

$\sum_{i=1}^{n} w_i$ olmasını sağlamaktadır. Burada bulunan öncelik vektörüne özvektör yaklaşımı denilmektedir.

Hesaplama işlemine, öncelik vektörünün iki ardışık hesaplaması arasındaki fark önceden belirlenen bir değerden küçük olduğunda son verilmektedir. Bu özvektöre karşılık gelen en büyük özdeğer ise;

$$\lambda_{max} = \sum_{j=1}^{n} \frac{a_{1j} w_j}{w_1} \quad \textbf{(4.12)}$$

A_{ij} 'de küçük değişiklikler olması, λ_{max}'da küçük değişiklikler olacağı anlamına geldiğinden, λ_{max}'ın n'den sapması bir tutarlılık ölçüsü vermektedir [54].

4.1.4.2 Tutarlılık ölçümü

Tutarlılık, kriterlerin ya da alternatiflerin ikili karşılaştırmasının belirlenmesinde kararın uyumluluk göstermesidir. Matematiksel olarak, eğer tüm i, j ve k'ler için $a_{ij}.a_{jk} = a_{ik}$ ise, A karşılaştırma matrisi tutarlıdır denebilir [55]. Önemli olan tutarsızlığın kabul edilebilir bir derecede olmasıdır. Genel bir kural olarak tutarsızlığın %10 seviyesini aşmaması istenir. Bu, kıyaslama matrisinde şu şekilde test edilir. İkili kıyaslama matrisinin tutarlı olması için $\lambda_{max} = n$ olmalıdır. Burada λ_{max}, matrisin maksimum özdeğeri, n ise ikili kıyaslamaya tabi tutulan eleman sayısıdır. İkili karşılaştırma yargılarının tutarlılığını hesaplayabilmek için özvektör yöntemi büyük kolaylık sağlamaktadır. İkili karşılaştırma matrisinin a_{ij} girdilerindeki değişiklikler matrisin en büyük λ_{max} özdeğerinde de değişime neden olduğundan, $\lambda_{max} - n$ farkı bir tutarlılık ölçüsünü vermektedir. İkili karşılaştırma matrisinin büyüklüğüyle (n) bu ölçümün normalleştirilmesini, Saaty tutarlılık indeksi (CI) olarak tanımlamıştır [57].

$$CI = \frac{\lambda_{max} - n}{n-1} \quad \textbf{(4.13)}$$

Bir tutarlılık oranı (CR) hesaplayabilmek için, Saaty ve diğerleri bir rastgele indeks (RI) oluşturmuşlardır. Rastgele indeks çizelge 4.2'de gösterilmiştir. Bu rastgele indeks, 1-15 boyutlu matrislerin her bir boyutunda 100'er matris rastgele doldurularak, CI eşitliği ile verilen tutarlılık indeksleri hesaplanmış ve her bir boyut için bu tutarlılık indekslerinin ortalaması alınarak oluşturulmuştur. Ancak 11-15 boyutlu matrislerin ortalama rastgele indekslerinde düzensiz artışlar gerçekleşmiştir. Matris boyutu arttıkça rastgele indekslerin de artmasının beklenen bir sonuç olması nedeniyle, matris boyutu 11-15 olan matrisler için 500'er rastgele ikili karşılaştırma matrisi oluşturularak hesaplamalar tekrar edilmiştir [54]. Tutarlılık oranı tutarlılık indeksinin aynı boyuttaki matrise karşılık gelen rastgele indekse oranlanmasıdır.

$$CR = \frac{CI}{RI} \quad \textbf{(4.14)}$$

Tutarlılık oranı 0.1'den küçük olduğunda yargıların tutarlı olduğu sonucuna ulaşılır. Eğer, CR 0.1'den büyükse yargıların tutarsız olduğu anlaşılır ve karar vericinin tutarsızlık oranını düşürmek için yargılarını yeniden gözden geçirmesi gerekmektedir.

Hesaplanan CI değeri, faktör sayısına göre çizelge 4.2'de görülen rastgele indeks tablosunda karşılık gelen RI değerine denklem 4.14' te gösterildiği şekilde bölünür. Bu işlem sonrası CR değerine ulaşılır. Elde edilen CR değeri 0.1'den küçük ise model tutarlıdır.

Çizelge 4.2 : Rastgele indeks.

n	R.I	n	R.I
1	0	9	1,45
2	0	10	1,49
3	0,58	11	1,51
4	0,9	12	1,54
5	1,12	13	1,56
6	1,24	14	1,57
7	1,32	15	1,59
8	1,41		

4.2 AHP'nin Avantaj ve Dezavantajları

AHP'nin Avantajları:

• AHP ile bir hiyerarşi kurularak karar problemleri biçimsel olarak ifade edilir. Bu şekilde karmaşık problemler bileşenlerine ayrılarak karışıklıkları giderilir ve basit bir yapıya kavuşturulur.

• AHP' de elemanların ikili karşılaştırmaları sırasında karar vericinin kişisel hükümleri kullanılır. Böylece karar verme sürecinde sadece sayısal verilere dayalı çözüm aranmamakta, karar verme işlemini yapan kişilerin fikir ve düşünceleri de dikkate alınmaktadır.

• Karar verici, ikili karşılaştırmaları kullanmak suretiyle problemin her bir parçasına daha fazla yoğunlaşabilir. Bu esnada sadece iki elemanın düşünülmesi nedeniyle verilecek hükümler basitleşmektedir. Öte yandan hükümleri sayısal değer

ile ifade etme güçlüğü söz konusu ise sözel hükümlerin kullanılması da mümkündür.

• AHP' de karar verici hem objektif hem de subjektif faktörleri beraberce dikkate alarak alternatiflerini değerlendirebilir ve en uygun alternatifin seçilmesine yönelik karar alabilir.

• Karar vericinin yaptığı ikili karşılaştırmaların tutarlılığını test etmesi de mümkündür. Böylece karar verici, tutarsızlık durumunda verdiği hükümleri tekrar ele alarak düzeltme imkânına sahiptir.

Genel itibariyle AHP'nin kullanımı kolaydır. Sıkı matematiksel temelleri olmasına rağmen AHP, hem kullanıcılar hem de akademisyenler tarafından teorinin önceki gelişim bilgisi tam olmaksızın kendi uzmanlık alanlarındaki aşırılığa rağmen kullanım basitliği kriterini karşılamaktadır. AHP'nin kullanım kolaylığı ile ilgili bazı sebepler ise şunlardır [56]:

• İnsanlar AHP'yi kolay bulmakta ve metoda yabancılaşma yerine genellikle metottan etkilenmektedirler.

• AHP ileri seviyede teknik bilgi gerektirmemekte ve neredeyse herkes kullanabilmektedir.

• AHP insanların düşünceleri kadar duyguları ve sezgileri üzerine yaptığı yargılar ile ilgilenmektedir.

• AHP elle tutulabilenlerin yanı sıra tutulamayanlarla da uğraşmaktadır. Duygulardan algılanılması gereken nasıl hissedildiğinin benzer şekilde akıl aracılığıyla dağıtılmasıdır.

• AHP akıldan doğrudan olarak alınmış rakamları atamak yerine iki taraflı karşılaştırılan ölçek çıkarmaktadır.

• AHP varsayılan ölçekteki ölçümleri kabul etmemekte fakat ölçek değerlerinin problemin kriterlerine göre yorumlarını sormaktadır.

• AHP karar problemlerini temsil etmek için basit ayrıntılandırılmış hiyerarşik yapılara güvenmektedir. Böyle uygun bir temsil ile risk, çatışma ve tahmin etme problemleri ile uğraşabilmektedir.

• AHP direk kaynak tahsisi, kar/zarar analizi, çatışmaların çözümü, sistemleri dizayn ve optimize etmede kullanılabilmektedir.

• AHP kararların nasıl verilebileceğini emretmek yerine nasıl iyi bir karar verileceğini tanımlayan bir yaklaşımdır. O anlık bir zamanda yasayan hiç kimse her zaman insanlar için neyin iyi olduğunu bilemez.

• AHP farklı uzmanlık ve tercihlerin düşünülmesi gereken grup kararları verilirken bir cevaba ulaşmak için basit ve etkili bir prosedür sağlamaktadır.

• AHP her bir bölüm için fayda ve mahsurların bağlantıları arasındaki ilişkilere odaklanarak çatışmaların çözülmesinde uygulanabilmektedir.

AHP'nin Dezavantajları:

• AHP'nin metodolojisi doğru kararı garanti etmez. AHP daha iyi karar verilmesine ve fikir birliğine ulaşılmasını sağlar.

• Hiyerarşik yapıda artış olduğu zaman ikili karşılaştırma matrislerinin sayısında da artış olur. Bu da zaman ve efor harcanmasına sebep olur.

• Yöntemin bireysel performans değerlendirmesinde kullanılmasında, değerlendirici sürekli ikili karşılaştırmalar yaptığı için konuya çok daha fazla konsantre olmakta ve dolayısıyla daha fazla zaman harcamaktadır.

• Yeni bir alternatifin eklenmesi diğer alternatiflerin sıralamasını değiştirebilmektedir.

Bu haliyle AHP karmaşık karar problemlerinin analizinde sağladığı basitlik, esneklik, kullanım kolaylığı ve rahat yorumlanması ile her türlü kişisel, kurumsal, ulusal vb. problemlere kolaylıkla uygulanabilecek durumdadır.

4.3 AHP Tekniğinin Uygulama Alanları

AHP özellikle 1973 yılından sonra pek çok alanda uygulama alanı bulmuştur.

AHP'nin literatürdeki uygulama alanları aşağıda verilmiştir [57].

A. Ekonomi/Yönetim problemleri:

• hesap denetimi,

• veri tabanı seçimi,

• dizayn ve mimarlık,

• muhasebe ve finans,

- sermaye yatırımı,
- karar destek,
- üretim,
- makro-ekonomik planlama,
- pazarlama,
- tüketici seçimi,
- ürün tasarımı,
- pazarlama stratejisi,
- planlama,
- portföy seçimi,
- risk analizi,
- başvuru değerlendirmeleri,
- grup karar verme,
- tesis yeri seçimi,
- kaynak tahsisi,
- politika/strateji,
- ulaştırma,
- su araştırma.

B- Politik problemler:

- silah kontrolü,
- çatışma analizi,
- politik adaylık,
- güvenlik değerlendirmesi.

C- Sosyal problemler:

- rekabetteki davranış sekli,
- eğitim,

- çevresel kararlar,
- sağlık,
- kanun düzenleme,
- tedavi seçimi,
- nüfus dinamikleri,
- kamu sektörü.

D- Teknolojik problemler:

- pazar seçimi,
- portföy seçimi,
- teknoloji transferi,
- bilgisayar ve bilgi seçimi,
- uzay araştırmaları.

5. UYGULAMA

Uygulama bölümünde öncelikle yazılım geliştirme metodolojilerinden scrum ve şelale hakkında bilgi verilecektir. Scrum ve şelale metodolojilerinin genel yapısı, işleyişi, kuralları, roller ve önemli noktaları üzerinde durulduktan sonra yapılan anket çalışmaları anlatılacaktır. Anket çalışmaları ile bu metodolojilerin kıyaslanmasında kullanılacak faktörlerin belirlenmesi sağlanacaktır. Ana faktörler ve alt faktörler belirlendikten sonra Analitik Hiyerarşi Prosesi yöntemiyle yazılım geliştirme metodolojileri karşılaştırma modeli kurulacak ve super decision programına aktarılacaktır. Superdecision programında 5 uzman kişinin görüşü ile faktörlerin öncelik seviyeleri ve ağırlıklandırmaları yapılacaktır. 1-9 skalasında faktörler arası karşılaştırmalar yapılarak öncelikler belirlenecektir. AHP metoduyla faktörlerin öncelikleri belirlendikten sonra model ev öncelik listesi excele aktarılacaktır. 5 uzmanın görüşüyle excelde scrum ve şelale metodolojileri için 1-5 skalasında puanlandırmalar yapılacaktır. Bu puanlara göre her iki metodoloji arasındaki kıyaslama yapılacak ve çıkan sonuçların analizi yapılarak hangi metodolojinin seçilmesi gerektiğine karar verilecektir. Sonuçlar analiz edilerek ve öneriler getirilerek uygulama çalışması sonlandırılacaktır.

Uygulama çalışması bilişim sektöründe bulunan ve Türkiye'nin önemli bankalarından birinin yazılımlarını yapan , bilişim firmaları sıralamasında ilk 500 şirket arasında bulunan ve hem şelale hemde scrum metodolojileriyle çalışan bir firmada yapılmıştır. Anket çalışmaları ve karşılaştırma modelinin kurularak kıyaslamaların yapılmasında bu şirketin çalışanlarından destek alınmıştır.

5.1 Scrum

Scrum Jeff Sutherland ve Ken Schwaber tarafından 1990'ların ortalarında geliştirilmiştir. Yüksek seviyede belirsizlik arz eden projelerde son yıllarda yoğun biçimde uygulanan ve başarılı sonuçlar üreten bir tür metodolojidir. Scrum, küçük takımlarda uygulanması kolay olan bir proje yönetimi yöntemi olup, büyük projelerde takımların küçük parçalara ayrılması ile etkinleştirilir. Değişimi bir istisna

olarak görmeyen, adaptif sistem geliştirme hayat döngüsü perspektiflerinin genel özelliğine paralel olarak proje kapsamının proje boyunca sürekli değişeceğini en baştan kabul eden bir yöntemdir. Bu kabul doğrultusunda Scrum, fazla dokümantasyon yapmadan, kısa periyotlarda geliştirme yapılıp, sonuçların müşteriye sıkça gösterildiği ve geri beslemeler doğrultusunda geliştirmelerin yönünün sürekli değiştiği, gereksiz geliştirmelerin minimuma indirilmeye çalışıldığı bir sürecin yönetimini üstlenir.

Scrum, süreçlerini zeki çözümlerle tasarlamaları için geliştiricileri özgür bırakır. Temel olarak nesne tabanlı teknolojiyi benimser. Böylece birbirinden ayrık ve yönetilebilir ortamlar sağlar. Fakat basit arayüzlerle ve güçlü veri oryantasyonu ile CMMI gibi disiplinli süreç değerlendirme sistemlerine de uyarlanabilir.

Scrum'da 4 tür toplantı vardır.

- Sprint planlama toplantısı (Sprint Planning Meeting)
- Sprint gözden geçirme toplantısı (Sprint Review Meeting)
- Geriye Bakış toplantısı (Retrospective Metin)
- Günlük Scrum toplantısı (Daily Scrum Meeting)

Ayrıca Scrum için çok önemli dört kavram daha vardır.

- Ürün gereksinim dokümanı (Product Backlog)
- Sprint dokümanı (Sprint Backlog)
- Aksaklık dokümanı (Impediment Backlog)
- Sprint kalan zaman grafiği (Burndown Chart)

5.1.1 Scrum rolleri

Scrum süreçlerinde ana roller ve yardımcı roller olmak üzere iki tür rol vardır. Ana roller; Ürün Sahibi, Scrum Yöneticisi (Scrum Master), Takım Üyesidir. Yardımcı roller ise Paydaşlar (müşteriler, satıcılar) ve Yöneticilerdir.

Ürün Sahibi (Product Owner)

Alan bilgisine sahip, müşteri isteklerini anlayabilen, gereksinimleri belirleyen ve belgeleyen kişidir. Müşterinin takımdaki yansımasıdır. Asıl görev ve sorumluluğu sürekli takım ve müşteriyle iç içe çalışmak ve bunun sonucunda takımı

yönlendirmektir. Ne yapılması gerektiğine ve sürümün ne zaman teslim edilmesi gerektiğine karar verir. Ürün Sahibi, klasik proje yönetimi yaklaşımındaki Proje Yöneticisi'nin sorumluluklarının bir kısmını üzerine alır. Ürünün karlılığından (ROI) doğrudan sorumludur. Ürün gereksinimlerini, ihtiyacın müşteri gözündeki ve pazardaki kıymetine göre önceliklendirir. Belirli periyotlarla yapılan toplantılarda ürün gereksinimlerinin önceliklerini değiştirebilir. Projenin çıktılarını kabul veya reddeder. En kritik nokta olan projenin başarısından veya başarısızlığından tamamen Ürün Sahibi sorumludur. Henüz belirli bir müşterisi olmayan projelerde ise alan bilgisine sahip kurum içinden biri veya destek alınan bir danışman ürün sahibi olabilir.

Scrum Yöneticisi (Scrum Master)

Scrum süreçlerinin istenildiği gibi gidip gitmediğinden sorumludur. Scrum yöneticisi takımın verimliliğini en üst düzeye çıkarmaya çalışır ve proje sırasında karşılaşılan engelleri ortadan kaldırmaya çalışır. Takımın lideri değildir ama takımla diğer dış etkiler arasında bir tampon görevi görür. Takım üyelerini kurallara zorlayan kişidir. Proje süreci hakkındaki bilginin tüm takım üyelerine açık olması ve bu bilginin güncellenmesinden sorumludur. Sprint toplantılarını organize eder, yönetir ve toplantıların amacına ulaşması için gerekli önlemleri alır.

Takım üyesi (Team Member)

Takım üyeleri ürünün yetiştirilmesinden sorumludur. Genel farklı yeteneklere(tasarım, geliştirme, test, teknik iletişim gibi) sahip 5-9 kişiden oluşur.

Yönetici (Manager)

Ürünün geliştirildiği organizasyonda ürünün geliştirilmesi için çevresel şartları sağlayan kişilerdir.

Diğer Paydaşlar (Other Stakeholders)

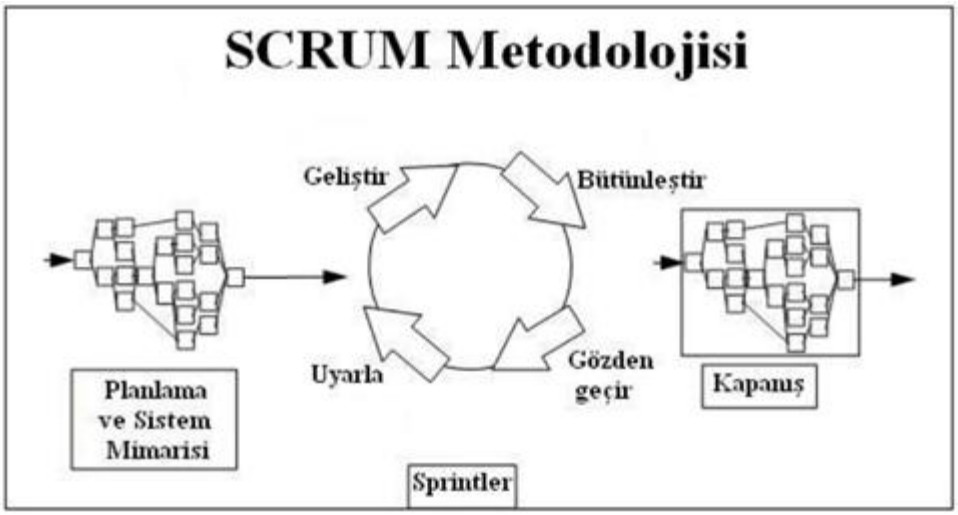
Müşteri, satıcı ve tedarikçilerden oluşur. Bu kişiler projenin sonunda ortaya çıkacak ürünü kullanacak olan ve faydalarını doğrulayabilecek kimselerdir. Sprint gözden geçirme toplantılarına katılırlar.

5.1.2 Scrum safhaları

Scrum şekil 5.1' de de görüldüğü gibi şu safhalardan oluşur[58]:

• Planlama

Scrum için en önemli safhadır. İlk olarak ürün gereksinim dökümanı geliştirilir. Sürüm ya da sürümlerin teslim tarihleri ve sürümlerin sağlayacakları yeni fonksiyonlar tanımlanır. Hemen geliştirilmesi en uygun sürüm bu safhada seçilir ve seçilen sürümdeki sürüm dokümanındaki maddelerle ürün paketleri eşleştirilir. Sürümü gerçekleştirecek takım tasarlanır, riskler değerlendirilir ve uygun risk kontrolleri sağlanır. Geliştirme araçları ve altyapı seçilir. Geliştirme, pazarlama, eğitim gibi konular göz önünde bulundurularak sürüm maliyeti tahmini yapılır. Yönetim onayı ve mali kaynak sağlanır.



Şekil 5.1 : Scrum safhaları.

• Mimari/Yüksek Seviye Tasarım

Gereksinim dokümanında atanan işler gözden geçirilir. Böylece değişiklik ihtiyacı varsa belirlenir. Sistem mimarisi yeni içerikleri ve gereksinimleri destekleyecek şekilde düzenlenir. Değişikliklerin geliştirilmesinde ortaya çıkabilecek problemler tanımlanır. Gerekli ise işlerde yeniden atamalar yapılır.

• Geliştirme (Sprint)

Geliştirme safhası, iteratif iş geliştirme döngüsü şeklindedir. Bu safhada takım toplantıları yapılarak sürüm planları gözden geçirilir. Ürünün uygun geliştirileceği standartlar için uyarlamalar yapılır. Ürün dağıtıma hazır olduğu düşünülene kadar

iteratif sprintler devam eder. Yönetim ise ürünün süre, kalite ve fonksiyonalite açısından istenenleri karşılayıp karşılamadığını, iterasyonların tamamlandığını ve kapanış safhasının geldiğini belirler. Bu yaklaşıma Eşzamanlı Mühendislik de denilir.

• Kapanış

Yönetim geliştirme safhasının bittiğine karar verdiği zaman başlar. Bu safhada daha genel bir ürün sürümü oluşturulur. Entegrasyon, sistem testi, kullanıcı dokümanı, eğitim malzemeleri ve pazarlama malzemelerinin hazırlanması kapanış görevleridir. Scrum safhalarındaki amaç ve faaliyetler şekil 5.2'de gösterilmiştir.

HAZIRLIK		GELİŞTİRME	SÜRÜM
Planlama	**Hazırlama**		
Amaç Vizyonu oluştur, beklentileri tespit et, parasal desteği/ kaynakları güvence altına al/ elde et	Amaç Daha fazla gereksinimi belirle ve ilk iterasyon için gerekli olan miktarını önceliklendir	Amaç 30 günlük iterasyonlarla (Sprint'ler) sistemi kodla/gerçekleştir	Amaç Operasyonel uygulama
Faaliyetler Vizyonu,bütçeyi,ilk başlangıç *Product Backlog'u* oluştur/yaz, öğelere ait tahminleri oluştur. Keşfedici tasarım ve prototipler ...	Faaliyetler Planlama Keşfedici tasarım ve prototipler ...	Faaliyetler Her iterasyonda Sprint'in(iterasyonun) planlanması için toplantılar Sprint Backlog'unun tanımlanması, tahminler Günlük scrum toplantıları Sprint(iterasyon) gözden geçirmeleri(review) ...	Faaliyetler Belgeleme Son Kullanıcı Eğitimi Pazarlama ve Satış ...

Şekil 5.2 : Scrum safhalarındaki amaç ve faaliyetler.

5.1.3 Scrum süreci

Tüm rollerin aktif olarak görev aldıkları genel bir Scrum süreci şöyledir:

Projenin başlangıç adımı olarak yazılım gereksinimlerinin ürün sahibi tarafından ürün gereksinim dokümanına yazılmasını düşünebiliriz. Ürün gereksinim dokümanı, Scrum'ın kalbidir [59].

Bu dokümanın sahibi ürün sahibidir ve gereksinimleri önceliklerine göre sıralar. Daha sonra takımla yapılan toplantılarda henüz net olmayan gereksinimler için geliştirme süresi tahminleri yapılır. Ayrıca ürün sahibi bu dokümana yazılım

geliştirme süresince eklemeler ve çıkarmalar yapıp, öncelikleri değiştirme hakkına sahiptir.

Böylece ürün sahibi değişen ihtiyaçlarına uygun olarak bir yazılıma sahip olma şansını yakalamış olur. Ama bunun öncesinde Scrum yöneticisi takım üyeleri ile birlikte takımın hızını ve Sprint uzunluğunu belirler. Sprint süresi en az 2 en fazla 4 hafta olarak belirlenir. İdeal olan proje boyunca tüm Sprint'lerin aynı süreli olmasıdır. Scrum yöneticisi ayrıca takımın karşılaşacağı engel ve problemlerle ilgili Aksaklık Dokümanı'nı hazırlar.

	Item #	Description	Est	By
Very High				
	1	**Finish database versioning**	16	KH
	2	**Get rid of unneeded shared Java in database**	8	KH
	-	**Add licensing**	-	-
	3	Concurrent user licensing	16	TG
	4	Demo / Eval licensing	16	TG
		Analysis Manager		
	5	File formats we support are out of date	160	TG
	6	Round-trip Analyses	250	MC
High				
	-	**Enforce unique names**	-	-
	7	In main application	24	KH
	8	In import	24	AM
	-	**Admin Program**	-	-
	9	Delete users	4	JM
	-	**Analysis Manager**	-	-
	10	When items are removed from an analysis, they should show up again in the pick list in lower 1/2 of the analysis tab	8	TG
	-	**Query**	-	-
	11	Support for wildcards when searching	16	T&A
	12	Sorting of number attributes to handle negative numbers	16	T&A
	13	Horizontal scrolling	12	T&A
	-	**Population Genetics**	-	-
	14	Frequency Manager	400	T&M
	15	Query Tool	400	T&M
	16	Additional Editors (which ones)	240	T&M
	17	Study Variable Manager	240	T&M
	18	Haplotypes	320	T&M
	19	**Add icons for v1.1 or 2.0**	-	-
	-	**Pedigree Manager**	-	-
	20	Validate Derived kindred	4	KH
Medium				
	-	**Explorer**	-	-
	21	Launch tab synchronization (only show queries/analyses for logged in users)	8	T&A
	22	Delete settings (?)	4	T&A

Şekil 5.3 : Ürün gereksinim dokümanı.

Gereksinimler belirlendikten sonra yazılım geliştirme takımı Sprint planlama toplantısında bir sonraki Sprint'te geliştirilmek üzere şekil 5.3'te örneği gösterilen ürün gereksinim dokümanından ürün sahibinin belirlediği yüksek öncelikli gereksinimleri seçerek Sprint dokümanına aktarırlar. Bu toplantıya Scrum yöneticisi, ürün sahibi ve takım üyeleri katılırlar.

Sprint planlama toplantılarını Scrum yöneticisi yönetir. Scrum yöneticisinin asıl görevi Scrum'ın temel prensiplerinin projeye uygulanmasını, bu prensiplerin takım üyelerince doğru şekilde anlaşılmasını sağlamaktır. En önemli görevi ise Sprint

süresince takımı dışardan gelebilecek etkilere karşı korumak ve takımın ihtiyaçlarını karşılamaktır.

Scrum her Sprint'in sonunda mutlaka ürün sahibine kullanabileceği bir yazılım sağlamayı hedefler, bundan dolayı planlanan Sprint süresi (2-4 hafta) asla uzatılmaz. Fakat eğer bir gereksinim belirlenen Sprint süresi içerisinde gerçekleştirilemeyecekse bir sonraki Sprint'e aktarılabilir. Ve aynı şekilde eğer Sprint süresi bitmeden Sprint dokümanındaki gereksinimlerin hepsi tamamlanmışsa ürün gereksinim dokümanından yeni gereksinimler Sprint dokümanına aktarılabilir.

Sprint planlama toplantısında belirlenen gereksinimler takım üyelerince küçük görevlere bölünerek takım üyelerine geliştirilmek üzere atanır. Scrum takımı geleneksel yazılım geliştirme süreçlerinden farklı olarak kesin rollere sahip değildir. Scrum takımındaki bütün üyeler çapraz görevlerde yer alabilirler, böylece kodun tek bir kişiye bağımlılığı riski ortadan kaldırılmış olur. Sprint dokümanının sahibi bu sefer ürün sahibi değil yazılım geliştirme takımıdır, dolayısıyla bu dokümana ürün sahibi değil takım üyeleri katkıda bulunurlar.Sprint dokümanı şekil 5.4'te gösterilmiştir.

Görevler	Pzt	Salı	Çar	Per	Cuma
	8	4	8		
	16	12	10	4	
	8	16	16	11	8
	12				
	8	8	8	8	8
			8	4	

Şekil 5.4 : Sprint dokümanı.

Sprint dokümanına aktarılan gereksinimlerin tahmini geliştirme süresi saat bazında takım üyelerince belirlenir ve Sprint boyunca sürekli olarak tahmini bu zamanlar güncellenerek Sprint kalan zaman grafikleri (burndown chart) oluşturulur. Böylece Sprint süresince ürün sahibi ve Scrum yöneticisi Sprint'in genel gidişi hakkında bilgi sahibi olur, aynı zamanda takım elemanları da kalan iş sürelerini ve harcadıkları zamanı takip edebilirler. Sprint kalan zaman grafiği şekil 5.5'te gösterilmiştir.

Şekil 5.5 : Sprint kalan zaman grafiği.

Scrum'ın belki de verimliliği artıran en önemli kavramlarından biri de Günlük Scrum toplantılarıdır. Bu toplantılar her gün belirli saatlerde farklı bir takım üyesinin liderliğinde şekil 5.6'da görüldüğü gibi ayaküstü yapılır ve en fazla 15 dakika sürer. Bu toplantılarda her takım üyesi şu 3 soruya cevap verir;

- Dün ne yaptım?
- Bugün ne yapacağım?
- Önümde olan engeller ve karşılaştığım sorunlar neler?

Bu toplantılara herkesin zamanında ve davet edilmeden katılması ve uzun sürmemesi çok önemlidir. Bu toplantılar sayesinde takım üyelerinin her biri diğer üyelerin nelerle uğraştığını öğrenme fırsatını edinirler ve çalışacakları işleri diğerleriyle paylaştıkları için işlerine daha iyi konsantre olabilirler. Scrum yöneticisi bu konuda takımı desteklemek ve diğer dış etkenlerle arayüz olmaktan sorumludur.

Şekil 5.6 : Günlük scrum toplantısı.

Her Sprint'in bitiminde ortaya konulan ürün hakkında geri besleme alabilmek için yazılımla alakalı her türlü kişiye (ürün sahibi, pazarlama, diğer takımlar vs.) açık Sprint Gözden Geçirme Toplantısı yapılır. Bu toplantının amacı yazılımın ürün sahibinin gereksinimlerine uygun olarak geliştirildiğinden emin olmaktır. Bu sayede müşterinin gereksinimleri bir şekilde yanlış anlaşılmış ise bu fark edilir ve bir sonraki Sprint'de bu hataların önüne geçilir.

Daha sonra Scrum yöneticisi ve takım üyeleri bir araya gelerek Geriye Bakış Toplantısı düzenlerler. Bu toplantıda Sprint'te karşılaşılan problemler tartışılır. Sprint'in eksikleri ve iyi yönleri konusunda fikirler paylaşılır. Üretkenliği ve kaliteyi olumsuz etkileyen durumlar aksaklık olarak tanımlanır. Bu durumlar süreç ile ilgili olabileceği gibi (yetersiz toplantı süreleri, Sprint süresinin kısa olması, dokümantasyon yetersizlikleri vs.) teknik sorunlar da olabilir (yavaş internet bağlantısı, uzaktan erişim hakları vs). Tüm aksaklıklar önceliklendirilir ve çözülmek üzere Aksaklık Dokümanı'na eklenir.

Bu adımlar ürün sahibinin ürün gereksinim dokümanına yazdığı, zaman içinde geliştirip, değiştirdiği gereksinimler bitene kadar tekrarlanır. Böylece Scrum yönetimi ile ürün geliştirme sağlanır.

5.1.4 Scrum'ın faydaları

Çevik süreçlerin planlama ve yönetimi konusunda etkili çözümler sunan Scrum'ın yazılım projelerine sağladığı en önemli katkılar şunlardır:

- Hızlı hayata geçirilen projeler
- Esnek alt yapı sayesinde kaliteli yazılımlar
- Maliyetlerin azaltılması
- Sürekli teslimat ile yüksek müşteri memnuniyeti
- Motivasyonu yüksek çalışanlar
- Kendi kendine organize olabilen takımlar

5.2 Şelale Metodolojisi

Bu metodoloji ortaya atılan ilk yazılım geliştirme metodolojisiydi ve önerildiği 1970 yılında adı bile bu değildi. 1970'li yıllarda Winston W. Royce tarafından yazılan bir

makalede oluşturulan Şelale modeli yıllarca birçok firma ve kurum tarafından kullanıldı[16].

Royce yazdığı makalesinde 7 süreçten bahsediyordu,

- Gereksinim analizi
- Teknik Tasarım
- Yazılım Geliştirme
- Entegrasyon
- Test ve doğrulama
- Canlı hayata geçiş
- Bakım

1. Gereksinim analizi: Bu süreç müşteriden ihtiyaçların alınıp analizin yapılması sürecidir. Bir yazılım projesinin köşe taşlarından birisi analiz alınma safhasıdır. İster müşteri ihtiyacını fark edip size ulaşmış olsun, ister siz müşteriye ulaşmış olun müşteriden tam olarak ihtiyacını ve isteklerini almak gerçekten başarı isteyen bir süreçtir. Bu noktada müşteri ile etkin bir iletişim şarttır. Ayrıca bu süreç program yöneticisi ve proje yöneticisinin kontrolünde alanında uzman iş analistleri tarafından yürütülür.

Deneyimli bir iş analisti, müşteriden ihtiyaçları alırken çoğu zaman müşterinin ilerde talep edebileceği şeyleri de kestirip bunları analize dahil eder. Bu da size birçok zaman kazandırır. Ancak ne kadar deneyimli olursa olsun hiç bir iş analisti müşteriden tek seferde mükemmel bir analiz alamaz. Çünkü müşteri hiç bir zaman tam olarak ne istediğini bilmez, çoğu zaman müşteri uygulamanın ilk prototiplerini gördükçe aslında neye ihtiyacı olduğunu anlamaya başlar ve burada değişiklik istekleri gelmeye başlar. Bu nokta da ise Değişiklik Yönetim Sürecini başarılı şekilde yönetmek çok önemlidir.

2. Teknik Tasarım: Alınan analize uygun olarak sistem mimarisinin ve yapısının çıkartılması sürecidir. Başlangıç analizi alındıktan ve gereksinimler belirlendikten sonra nihayatinde ortaya çıkacak uygulamanın mimari tasarımı ve teknik çözümlemesi yapılır. Bu aşama yine konusunda uzman olan ekip üyeleri tarafından yapılır. Kullanılacak teknoloji belirlendikten sonra oluşturulacak sisteminin taslağı

(blue print) oluşturulur. Burada önemli olan bir önceki adımın ne kadar başarı bir şekilde yönetildiğidir. Eğer analiz aşaması yerine getirilmezse, yazılım geliştirme safhasında öngörülemeyecek birçok değişiklik isteği gelebilir ve bunlarda bazen sistemin başta kurulan yapısına tamemen ters düşebilir. Bu durum ise projenin kararsız haline gelmesine yol açabilir ya da ön görülmemiş ve planlanmamış bir çok eforun harcanmasına sebep olabilir.

3. Yazılım Geliştirme: Alınan analiz ve oluşturulan mimariye uygun olarak kod yazımı sürecidir. Sonuçta amaçlanan iş yazılım geliştirmektir, dolayısı ile bu sürecin de gerekli önem verilerek, gerekli yeterlilikteki uzmanlar ile gerçekleştirilmesi gerekir.

Yazılım geliştirme sürecinde önem verilmesi gereken bir konu kod gözden geçirmesidir; çünkü yazılım belli bir mimari yapıya ve kurallara uygun olarak yapılmaktadır. Özellikle yazılım ekibine yeni katılan üyeler olmak üzere yazılımcıların bu kurallara ve mimariye mümkün olduğunca sadık kalmasının zorlanması gerekir. Bu amaçla yazılan kodlar daha deneyimli yazılımcılar tarafından gözden geçirilerek kod kaynağına atılmalıdır ki mimariye ve kurallara uyuma gerekli önem verilsin. Şayet bu süreç de gereğince yerine getirilmezse mimariye ve kurallara uymayan yazılımcılarda bu alışkanlık haline gelecektir.

4. Entegrasyon: Bir uygulama genellikle birden fazla bileşen ve sistemden oluşur. Bazen bir web servis, bazen bir web uygulaması, bazen de windows servisi. Uygulama canlı hayata çıktığı zaman tüm bu farklı bileşenler beraber çalışacağı için öncelikle hepsinin bir araya getirilmesi gerekmektedir. Bu aşama yazılım geliştirme aşamasında geliştirilen farklı bileşen ve yapıların birbirine entegre edilmesi ve bir sistem olarak test edilmesi aşamasıdır.

5. Test ve doğrulama: Bu süreçte geliştirmesi bitirilen yazılım test edilir; zaten Şelale metodolojisinin önemli bir eksikliği de budur. Eğer test süreci sadece yazılım bittikten sonra uygulanırsa ciddi sorunlarla karşılaşma ihtimali yüksektir. Şöyle düşünelim, 6 ay veya 12 ay boyunca yazılım geliştirilmiş ve sadece geliştiriciler tarafından yapılan yüzeysel testler ile geliştirmeler bitirilmiş. Tüm iş bittikten sonra da yazılım asıl test sürecine alınmış ve o güne kadar fark edilmemiş birçok ciddi yapısal bozukluklar ortaya çıkmış. Bu durumda hesaplanmayan birçok efor ve maliyet ortaya çıkacaktır.

Şelale modelinde bir süreç tamamlanmadan bir sonrakine geçilmez yani yazılım geliştirilirken test aynı anda yürütülemez. Ayrıca nasıl bir Şelalede su yukarı akmayacaksa, Şelale metodolojisinde de testten tekrar yazılım geliştirme veya teknik tasarım sürecine dönülmez.

6. Canlı hayata geçiş: Bu süreçte ise sağ salim yazılımı tamamlanmış ve testlerden geçmiş yazılımın canlı hayata alımı yapılmaktadır. Yazılımı tamamlanan ve kabul testlerinden geçen projelerin canlı hayata alınması sürecine geçilir. Bu süreç normalde bu konu için özelleşen ekipler tarafından yürütülmesi en idealidir. Ancak yine çoğu zaman görülüyor ki bu işte proje ekibine kalmaktadır.

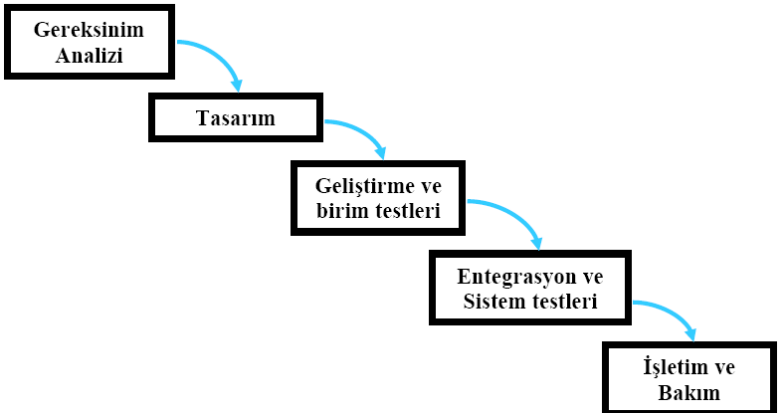
Uygulamanın canlı hayata nasıl alınacağı gereksinim analizi sürecinden sonra belirlenmesi gerekir. Çünkü bazen projenin sonuna gelindiğinde uygulamanın canlı hayata alınma şekli tamamen projenin yapısına ters düşmekte ve yine planlanmayan eforların harcanmasına sebep olmaktadır.

7. Bakım: Bakım süreci ise uygulama canlı hayatta iken verilecek destek ve güncelleme çalışmalarını kapsamaktadır. Uygulama canlıya alındıktan sonra, garanti süresine benzeyen bir süreç yürütülür. Uygulama da kodlamadan veya yazılımdan kaynaklanan oluşabilecek kesintilere destek verilir ve anlaşmalara bağlı olmak üzere yazılım üzerindeki güncelleştirmeler ile bakım yapılır.

Bu süreç de profesyonelce yürütülmesi gereken bir süreçtir, çünkü bu süreci yürütmekteki başarınız size aynı müşteriden yeni işlerin gelmesini sağlayıp müşteri memnuniyetini arttırabileceği gibi; sürecin başarısız bir şekilde yürütülmesi halinde ise hukuki anlaşmazlıklara varan sorunların ortaya çıkması mümkündür.

Modelin sıralı bir akış takip etmesi, modelin proje açısından yönetimini kolaylaştırır. Aynı sebepten ötürü, yazılımın standartlara uygunluğunun kontrolü de kolaydır Bütün bu avantajların yanında Şelale modelinin çok önemli bir dezavantajı vardır. Projenin geç safhalarında gelen değişiklikler çok pahalıya mal olur. Şelale modeli sıralı bir akış izlediğinden, doğrulama safhasında gereksinimlerde oluşan bir değişiklik akış içersindeki tüm süreçlerden geçmek zorundadır (tasarım, kodlama ve doğrulama). Bu yüzden gereksinimlerdeki bu değişiklik çok pahalıya mal olur.

Şelale modeli şekil 5.7'de görüldüğü şekilde özetlenebilir.



Şekil 5.7 : Şelale modeli.

Şelale modeli, yazılım geliştirmenin gereksinim analizi, tasarım, implementasyon, test, entegrasyon ve bakım gibi fazları boyunca aşağı doğru (Şelale gibi) devam eder gibi göründüğü sıralı geliştirme sürecidir. Şelale modelinin temel prensipleri;

- Proje ardışık fazlara bölünmüştür.

- Temel vurgu, tüm sistemin planı, iş programı, hedeflenen tarihi, bütçesi ve implementasyonu üzerindedir.

- Proje süreci devam ederken yeni bir faza başlamadan önce tamamlanmış son faza ait özellikle kullanıcıların ve IT yönetiminin formal yorumlarını, onay ve kabul imzalarını içeren kapsamlı şekilde yazılmış bir dokümanın varlığı.

Görüldüğü üzere temel süreçler hep aynı, ancak Şelale metodolojisinin sıkıntılı noktası süreçler arasında zıplama yapılamaması veya geri gidilememesi. Başka bir sıkıntı da dikkat ettiyseniz müşteriden gelebilecek değişiklik isteklerini yönetebilecek bir süreç yok Şelale 'de. Yani müşteriden yazılım sürecinin ortasında gelebilecek bir değişiklik isteği, şayet alt yapıyı toptan değiştirebilecek birşeyse ciddi bir sorunla karşı karşıyasınız demektir.

Bu olumsuzluklara rağmen Şelale yıllar boyunca özellikle Amerika da Savunma Bakanlığı ve Nasa projelerinde yıllarca kullanıldı, şimdi ise bu kurumlardaki yazılım süreçleri Scrum gibi daha güncel süreçlerle değiştirilmektedir.

Şelale, şayet yazılımda çok birşey değişmeyecek ise, ihtiyaçlar açık ve çok kompleks değilse, yazılım ekibi çok kaliteli ve testte öngörülmedik bir durum ortaya çıkmasına sebep olmayacak durumda ise kullanılabilir. Tabi ki böyle ideal bir durum ne yazık ki pek gerçekçi değildir, Ken Schewaber'a göre bir yazılım projesindeki ihtiyaçların %35'i değişmektedir. Böyle bir rakam ise ciddi bir riski teşkil etmektedir.

Ayrıca, Steve McConnel'ın hesaplarına göre "bir gereksinim probleminin yazılım geliştirme veya bakım aşamasına kadar tespit edilememesi, bu problemin gereksinim analizi sırasında fark edilmesine göre 50 ile 200 kat daha fazla maliyete sebep oluyor". Bu durumda Şelale metodolojisinin her sürecinin tam doğruluk ile yapılması zorunluluğunu açıklıyor.

5.3 Karşılaştırmada Kullanılacak Faktörlerin Seçimi

Bilişim teknolojileri hayatımıza hızlı bir giriş yaptı ve gün geçtikçe de gerek sosyal hayatımıza gerek iş dünyasına olan etkileri hızla artmaktadır. 2009 yılında yaşanan ekonomik krizin ardından 2010 yılında Avrupa bilişim sektörü hızlı bir canlanma gösterdi. Avrupa Bilişim pazarı 2010 yılında, yüzde 1,2 büyüme gerçekleştirdi. Bilişim pazarı 2010 yılında dünyada 2,5 trilyon Euro'ya (3,3 trilyon ABD dolar) ulaştı. Türkiye ise bu pazardan binde 8'lik bir pay aldı. 2010 yılında Türkiye Bilgi Teknolojileri ve İletişim sektörü büyüklüğü 27,3 milyar dolara ulaştı. Bu hacmin 20 milyar doları İletişim, 7,5 milyar doları ise Bilgi Teknolojileri sektöründen geldi. 2011 yılının ilk yarısında Dünya Bilişim sektöründe BT harcamaları yüzde 4,3 artarak 963,4 milyar avro seviyesine ulaşmış oldu. Bu hedeflenilenin de üzerinde büyük bir rakamdır (EITO- European Information Technology Observatory).

Bilişim sektöründeki firmalar teorik kısımda anlattığımız yazılım metodolojilerini kullanarak çalışmaktadır. Bu metodolojilerin en yaygın olarak kullanılanları ise scrum ve şelale metodolojileridir. Şelale bilişim sektöründe yıllardır kullanılan bir metodoloji olmasına rağmen son yıllarda firmalar çevik metodolojilerin en yaygın kullanılanı scrum'a geçiş yapmaktadır.Bu iki metodolojinin avantaj ve dezavantajları ile ilgili çok sayıda çalışma vardır, aşağıdaki uygulamada ise scrum ve şelale metodolojilerinin karşılaştırılmasında çok ölçütlü karar verme yöntemlerinden analitik hiyerarşi prosesi (AHP) yöntemi kullanılacaktır.

İlk olarak metodolojilerin karşılaştırılmasında kullanılacak faktörleri belirlemek amacıyla tek soruluk bir anket çalışması yapılmıştır. Ankette; her biri scrum ve Şelale metodolojilerini çalışma hayatında kullanmış 12 katılımcıya aşağıdaki soru yöneltilmiştir.

Scrum ve Şelale metodolojilerinin kıyaslanması ile ilgili yapılacak bir tez çalışmasında, kıyaslama konusunda hangi faktörlerin kullanılması gerekmektedir?

12 katılımcının görev tanımları Çizelge 5.1'de gösterilmiştir.

Çizelge 5.1 : Anket 1 katılımcı görev tanımları.

Görev Tanımı	Kişi Sayısı
İş Analisti	7
Yazılımcı	3
Grup Lideri	1
Servis Yöneticisi	1

Anket sorusuna verilen cevaplarda katılımcıların belirttiği faktörler ve hangi faktöri kaç katılımcının cevap olarak yazdığı Çizelge 5.2 de gösterilmiştir.

Çizelge 5.2 : Yazılan faktörler ve faktörü yazan katılımcı sayıları.

Faktörler	Katılımcı Sayısı
Maliyet	12
Öncelikli işlerin teslim süreleri	12
Fazla mesai süreleri	12
Müşterinin süreç içerisinde yer alması	12
Zaman Planlaması	11
Kapsam	11
Müşteri talebinin doğru şekilde karşılanması	11
Müşteri memnuniyeti	11
Proje ihtiyaçlarındaki değişimin sıklığı	11
Analiz Maliyeti	10
Yazılım Maliyeti	10
Destek Maliyeti	10
Yazılım Kalitesi	10
Planlama	10
Müşterinin ek taleplerinin karşılanması	10
Doküman ihtiyacı	10
Teknoloji gelişimi	10
Test Kolaylığı	9
Zaman	9
Yazılım Süresi	9
Doküman sayısı	9
Müşteri ile uyum	9
Müşteri ile iletişim	9
Süreç Maliyetleri	8
Test Maliyeti	8
Dış Kaynak Maliyetleri	8
Dış Kaynak Destek Maliyeti	8

Çizelge 5.2 (devam) : Yazılan faktörler ve faktörü yazan katılımcı sayıları.

Kalite	8
Doküman kalitesi	8
Kod kalitesi	8
Teslim Süreleri	8
Analiz Süresi	8
Test Süresi	8
Yaygınlaştırma Süresi	8
Takım toplantılarının verimliliği	8
Yaygınlaştırma Maliyeti	7
Tasarım Süresi	7
Destek Süresi	7
Kapsamın Karmaşıklığı	7
Dış Kaynak Analiz Maliyeti	6
Dış Kaynak Yazılım Maliyeti	6
Analiz Tasarım Kalitesi	6
Yazılımın Kullanılabilirliği	6
Organizasyon Alışkanlıkları	6
Takım üyelerinin motivasyonu	6
Tasarım Maliyeti	5
Kullanım Kolaylığı	5
Yazılımın Güvenilirliği	5
Takım üyelerinin konsantrasyonu	5
Tasarımın Doğruluğu	4
Yazılımın Genişletilebilirliği	4
Yazılımın Çalışabilirliği	4
Takım içi organize yapı	4
Takımdaki üye sayısı	4
Tasarımın Bütünlüğü	3
Yazılımın Esnekliği	3
Yazılımın Tekrar Kullanılabilirliği	3
Takımın tecrübesi	3
Takımın uyumu	3
Tasarımın Etkin Olması	2
Kültürel değişiklikler	1

Anket sonucunda katılımcıların tamamı; maliyet, öncelikli işlerin teslim süreleri, fazla mesai süreleri ve müşterilerin süreç içerisinde yer alması konularını faktör olarak belirtmişlerdir. En fazla katılımcı tarafından vurgulanan faktörler bunlardır. Kültürel değişiklikler ise tek bir katılımcı tarafından yazılarak en az katılımcının belirttiği faktör olmuştur.

Anket sonucunda elde edilen faktörler düzenlenerek ana faktör ve alt faktörler olarak ayrılmış ve yeni bir anketle 22 katılımcıya sunulmuştur. Bu ankette faktörlerin ne

kadar etkili olduğu sorusuna cevap aranarak modeli oluşturacak faktörlerin son halinin belirlenmesi amaçlanmıştır.

4 ana faktör Çizelge 5.3'de gösterilmiştir.

Çizelge 5.3 : Ana faktörler.

Ana Faktörler
Maliyet
Kalite
Zaman
Kapsam

Ana faktörlere ait alt faktörler Çizelge 5.4'de gösterilmiştir.

Çizelge 5.4 : Alt faktörler.

Ana Faktörler	Alt Faktörler
Maliyet	Süreç Maliyetleri
	Dış Kaynak Maliyetleri
Kalite	Analiz Tasarım Kalitesi
	Yazılım Kalitesi
Zaman	Teslim Süreleri
	Zaman Planlaması
Kapsam	Dış Etkenler
	İç Etkenler
	Takımın Durumu

Anket 2'ye katılan 22 katılımcının görev tanımları Çizelge 5.5'te gösterilmiştir.

Çizelge 5.5 : Anket 2 katılımcı görev tanımları.

Görev Tanımı	Kişi Sayısı
İş Analisti	13
Yazılımcı	5
Grup Lideri	2
Servis Yöneticisi	2

Anket 2'ye iki ayrı servisten toplam 22 kişi katılmıştır. 2 servis yöneticisi ve 2 grup lideri de katılımcılar arasında bulunmaktadır.

Yapılan bu ankette;

1- Hiçbir etkisi yok,

2- Az etkilidir,

3- Orta seviyede etkilidir,

4- Yüksek seviyede etkilidir,

5- Çok yüksek seviyede etkilidir olarak değerlendirilmiştir.

Anket sonucunda; faktörler için Çizelge 5.6'daki sonuçlar ortaya çıkmıştır.

Çizelge 5.6 : Faktörlere göre anket 2 sonuçları.

No	Ana Faktörler	Katılımcı Sayısı	Etki Düzeyi Ortalaması
1	Maliyet	22	4,59
2	Müşteri talebinin doğru şekilde karşılanması	22	4,59
3	Yazılım Kalitesi	22	4,55
4	Kapsam	22	4,55
5	Müşterinin ek taleplerinin karşılanması	22	4,55
6	Müşterinin süreç içerisinde yer alması	22	4,55
7	Öncelikli işlerin teslim süreleri	22	4,5
8	Kalite	22	4,45
9	Zaman	22	4,45
10	Zaman Planlaması	22	4,45
11	Teslim Süreleri	22	4,32
12	Kod kalitesi	22	4,27
13	Fazla mesai süreleri	22	4,27
14	İç Etkenler	22	4,23
15	Proje ihtiyaçlarındaki değişimin sıklığı	22	4,23
16	Dış Kaynak Maliyetleri	22	4,18
17	Doküman ihtiyacı	22	4,14
18	Doküman kalitesi	22	4,09
19	Müşteri memnuniyeti	22	4,09
20	Dış Etkenler	22	4,09
21	Süreç Maliyetleri	22	4,05
22	Yazılım Maliyeti	22	4,05
23	Analiz Tasarım Kalitesi	22	4,05
24	Test Kolaylığı	22	4,05
25	Teknoloji gelişimi	22	4,05
26	Dış Kaynak Yazılım Maliyeti	22	3,86
27	Yazılım Süresi	22	3,82
28	Doküman sayısı	22	3,82

Çizelge 5.6 (devam) : Faktörlere göre anket 2 sonuçları.

29	Dış Kaynak Destek Maliyeti	22	3,77
30	Test Maliyeti	22	3,73
31	Test Süresi	22	3,73
32	Destek Süresi	22	3,73
33	Kullanılabilirlik	22	3,64
34	Müşteri ile uyum	22	3,64
35	Destek Maliyeti	22	3,55
36	Planlama	22	3,55
37	Müşteri ile iletişim	22	3,45
38	Kullanım Kolaylığı	22	3,41
39	Çalışabilirlik	22	3,41
40	Tasarım Maliyeti	22	3,32
41	Analiz Süresi	22	3,32
42	Yaygınlaştırma Süresi	22	3,32
43	Tasarım Süresi	22	3,27
44	Analiz Maliyeti	22	3,23
45	Yaygınlaştırma Maliyeti	22	3,23
46	Karmaşıklık	22	3,18
47	Takım toplantılarının verimliliği	22	3,14
48	Dış Kaynak Analiz Maliyeti	22	3,09
49	Organizasyon Alışkanlıkları	22	3,09
50	Doğruluk	22	2,95
51	Bütünlük	22	2,95
52	Takım içi organize yapı	22	2,91
53	Güvenilirlik	22	2,86
54	Takım üyelerinin konsantrasyonu	22	2,86
55	Genişletilebilirlik	22	2,77
56	Takım üyelerinin motivasyonu	22	2,68
57	Takımın uyumu	22	2,68
58	Esneklik	22	2,59
59	Takımın tecrübesi	22	2,45
60	Takımdaki üye sayısı	22	2,32
61	Tekrar Kullanılabilirlik	22	1,59
62	Etkinlik	22	1,55
63	Kültürel değişiklikler	22	1,36

Anket sonucunda; Kalite ana başlığı altında bulunan Yazılım kalitesi başlığı altındaki Tekrar Kullanılabilirlik kriteri 1.59, Tasarım kalitesi başlığı altındaki Etkinlik kriteri 1.55 ve Kapsam ana başlığı altında bulunan Dış Etkenler başlığı altındaki Kültürel değişiklikler 1.36 puan alarak 2 puanın altında değer alan kriterler olmuştur. Bu nedenle oluşturulacak AHP modelinden çıkarılmışlardır. Bu kriterler dışındaki tüm kriterler AHP modelinde yer alacaktır. Çizelge 5.6'da gösterilen alt kriterlere ait 2. Seviye alt kriterlerin son hali Çizelge 5.7'de gösterilmiştir. En yüksek puanı ise 4.59

ile maliyet ana kriteri ve Kapsam ana başlığı altında bulunan müşteri talebinin doğru şekilde karşılanması kriterleri almıştır. Alt Faktörler ve 2. Seviye Alt Faktörler Çizelge 5.7'de gösterilmiştir.

Çizelge 5.7 : Alt faktörler ve 2. seviye alt faktörler.

Alt Kriterler	2. Seviye Alt Kriterler
Süreç Maliyetleri	Analiz Maliyeti
	Destek Maliyeti
	Tasarım Maliyeti
	Test Maliyeti
	Yaygınlaştırma Maliyeti
	Yazılım Maliyeti
Dış Kaynak Maliyetleri	Dış Kaynak Analiz Maliyeti
	Dış Kaynak Destek Maliyeti
	Dış Kaynak Yazılım Maliyeti
Analiz Tasarım Kalitesi	Bütünlük
	Doküman Kalitesi
	Doğruluk
	Kullanım Kolaylığı
Yazılım Kalitesi	Esneklik
	Genişletilebilirlik
	Güvenilirlik
	Kod Kalitesi
	Kullanılabilirlik
	Test Kolaylığı
	Çalışabilirlik
Teslim Süreleri	Analiz Süresi
	Destek Süresi
	Tasarım Süresi
	Test Süresi
	Yaygınlaştırma Süresi
	Yazılım Süresi
Zaman Planlaması	Fazla Mesai Süreleri
	Öncelikli İşlerin Teslim Süreleri

Çizelge 5.7 (devam) : Alt faktörler ve 2. seviye alt faktörler.

Dış Etkenler	Müşteri ile İletişim
	Müşteri ile Uyum
	Proje İhtiyaçlarındaki Değişimin Sıklığı
	Teknolojik Gelişmeler
İç Etkenler	Doküman İhtiyacı
	Doküman Sayısı
	Karmaşıklık
	Müşteri Memnuniyeti
	Müşteri Talebinin Doğru Şekilde Karşılanması
	Müşterinin Ek Taleplerinin Karşılanması
	Müşterinin Süreç İçerisinde Yer Alması
	Organizasyon Alışkanlıkları
	Planlama
Takımın Durumu	Takım İçi Organize Yapı
	Takım Toplantılarının Verimliliği
	Takım Üyelerinin Konsantrasyonu
	Takım Üyelerinin Motivasyonu
	Takımdaki Üye Sayısı
	Takımın Tecrübesi
	Takımın Uyumu

5.4 Faktörlerin İkili Karşılaştırmalarının Yapılması

Çizelgelerde gösterildiği şekilde scrum ve Şelale metodolojilerinin kıyaslanmasında kullanılacak ana ve alt faktörler anketlerle belirlenmiştir. Bu faktörler arasında ikili karşılaştırmalar yapılarak faktörlerin önceliklendirmesi yapılacaktır. Çizelge 5.8'de ikili kıyaslamada kullanılacak sözel ifadelerin etki dereceleri gösterilmektedir.

Çizelge 5.8 : İkili kıyaslamada kullanılan sözel ifadelerin etki dereceleri.

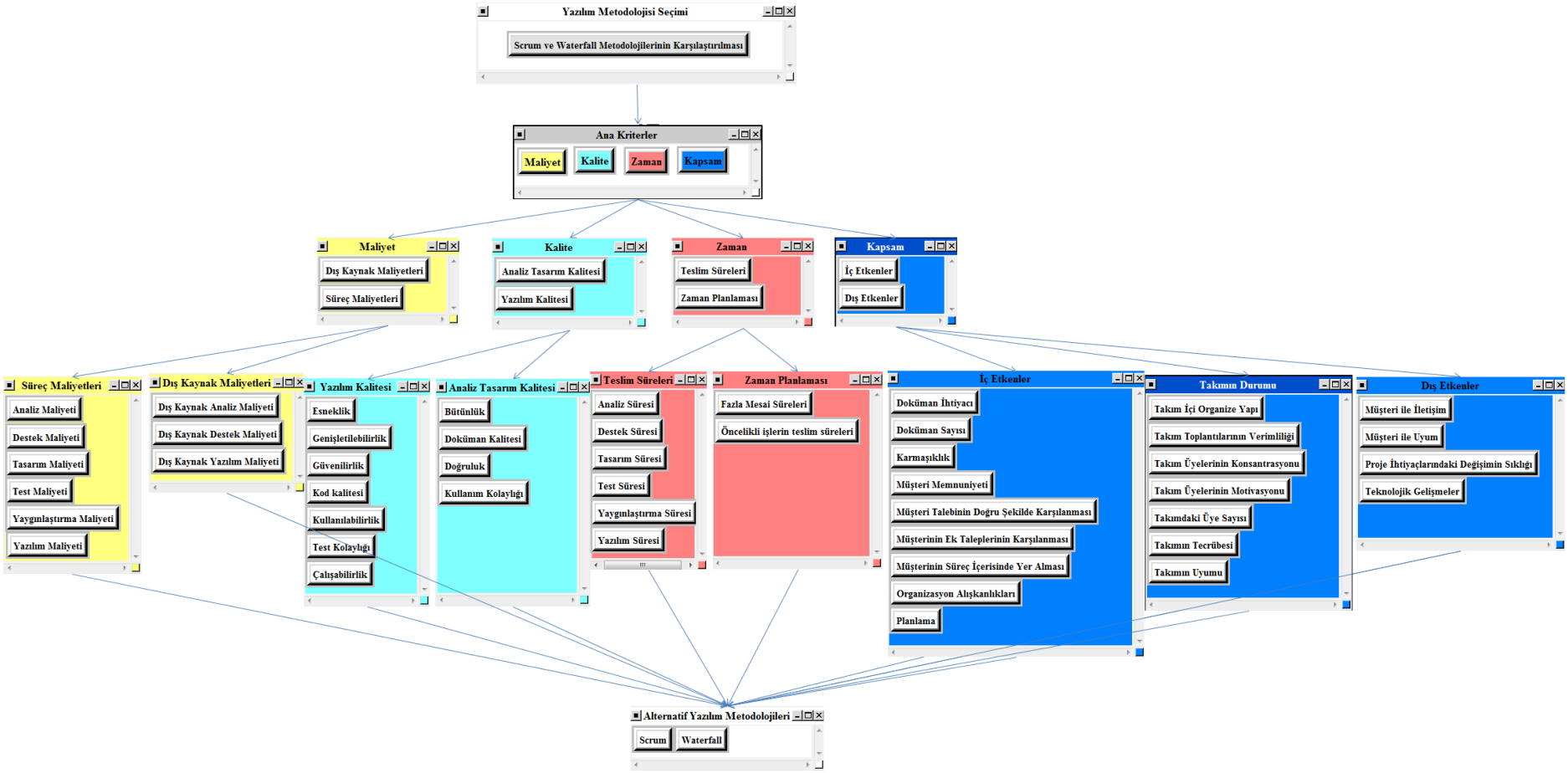
PUAN	SÖZEL ETKİ DERECESİ	TANIM
1	Eşit Etkidedir	İki faktör de eşit etki göstermektedir
3	Biraz daha fazla etkilidir	Tecrübe ve yargı ile bir faktör, diğerine göre biraz daha fazla derecede etkilidir
5	Kuvvetli derecede etkilidir	Tecrübe ve yargı ile bir faktör, diğerine göre kuvvetli derecede etkilidir
7	Çok kuvvetli derecede etkilidir	Bir faktör diğerine göre çok daha fazla etkilidir ve baskınlığı uygulamada da görülür
9	Tamamıyla etkilidir	Bir faktör diğerine göre çok daha fazla etkili olduğu çok büyük bir güvenilirliğe sahiptir
2, 4, 6, 8	Uzlaşma	Uzlaşma gerektiğinde kullanılmak üzere 2 yargı arasına düşen değer

Faktörlerin kıyaslanmasında scrum ve şelale metodolojilerinin her ikisinde de tecrübesi olan ve bilişim sektöründe çalışan 5 katılımcı ile toplantı yapılmıştır. Bu 5 kişinin fikir birliğine dayalı olarak kıyaslamalar yapılmış ve öncelikler belirlenmiştir. Toplantı sırasında tüm kriterler ayrıntılı şekilde değerlendirilerek sonuçlara ulaşılmıştır. Yapılan bu çalışmaya katılan 5 katılımcının görev tanımları Çizelge 5.9'da gösterilmiştir.

Çizelge 5.9 : İkili kıyaslama anketine katılan uzmanların profili.

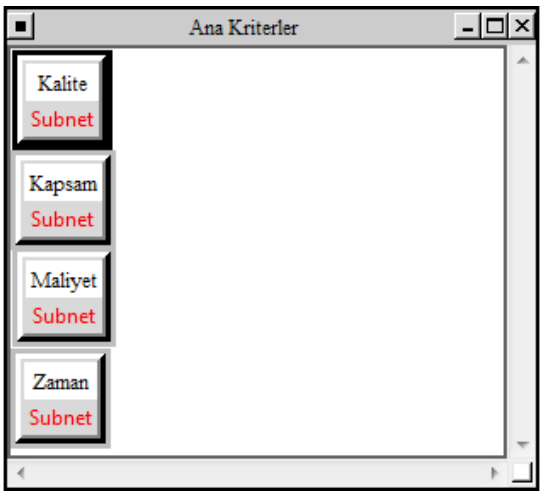
Unvan	Cinsiyet	Eğitim Durumu
Servis Yöneticisi	Erkek	Lisans
Grup Lideri	Erkek	Yüksek Lisans
Kıdemli Yazılım Uzmanı	Erkek	Lisans
Kıdemli Analiz Uzmanı	Kadın	Lisans
Analiz Uzmanı	Erkek	Yüksek Lisans

İkili karşılaştırmalar ile yapılacak AHP Yöntemi için Super Decision programı kullanılmıştır. Modelin genel görünümü Şekil 5.8'de gösterilmiştir.

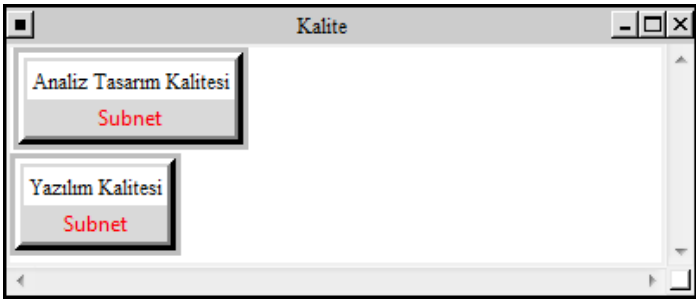


Şekil 5.8 : Scrum-şelale karşılaştırma modeli.

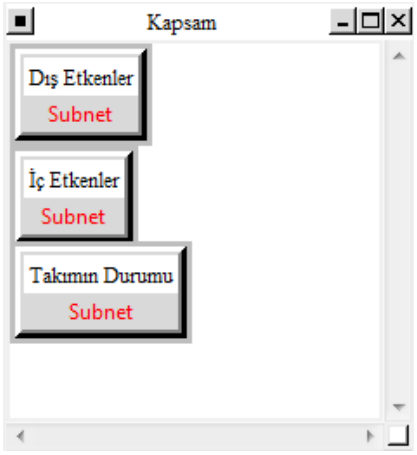
Superdecision programında oluşturulan modelin görüntüleri aşağıdaki şekillerde gösterilmiştir.



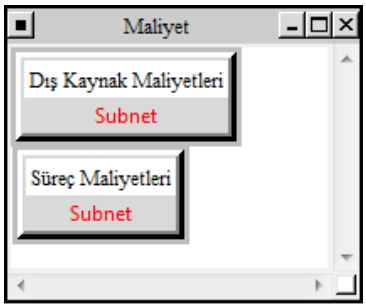
Şekil 5.9 : Ana kriterler.



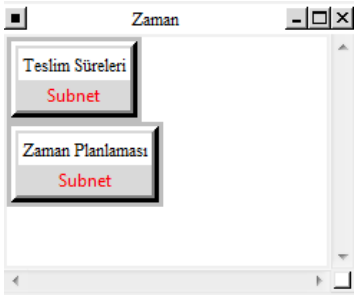
Şekil 5.10 : Kalite.



Şekil 5.11 : Kapsam.



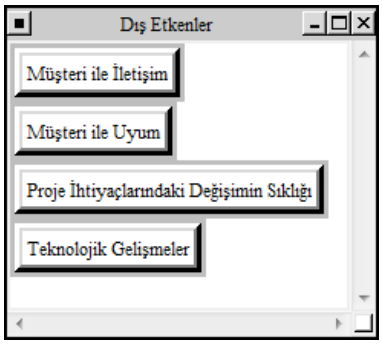
Şekil 5.12 : Maliyet.



Şekil 5.13 : Zaman.

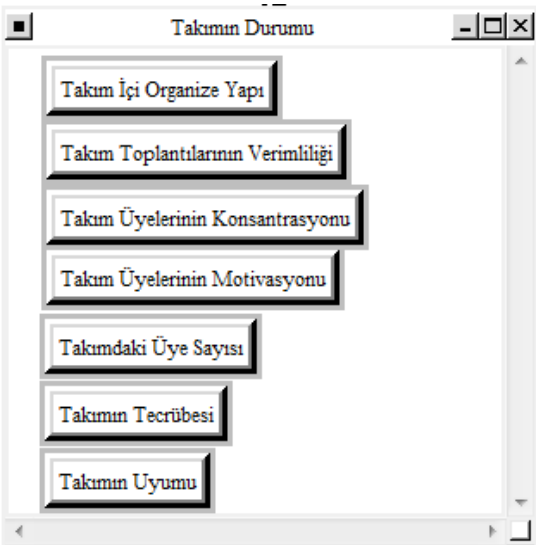
Şekil 5.14 : Analiz tasarım kalitesi.

Şekil 5.15 : Yazılım kalitesi.



Şekil 5.16 : Dış etkenler.

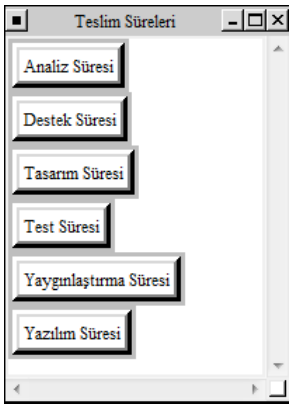
Şekil 5.17 : İç etkenler.



Şekil 5.18 : Takımın durumu.

Şekil 5.19 : Dış kaynak maliyetleri.

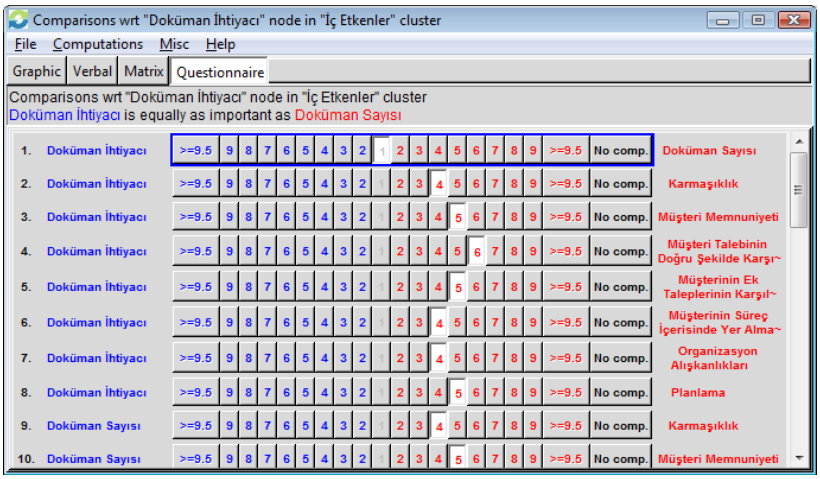
Şekil 5.20 : Süreç maliyetleri.



Şekil 5.21 : Teslim süreleri.

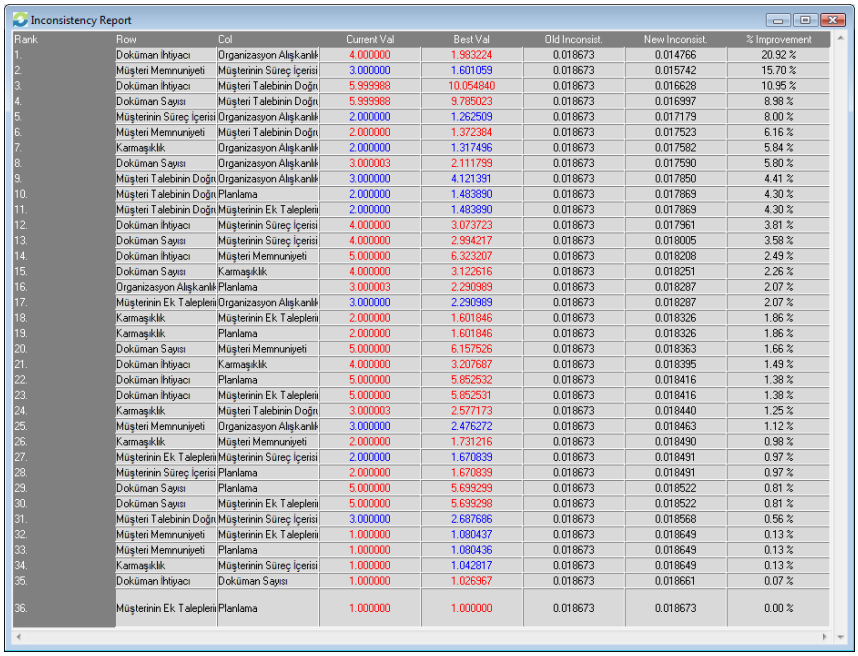
Şekil 5.22 : Zaman planlaması

Ankete uzmanların fikir birliğine dayalı olarak puanlar girilmiştir. Toplamda 134 adet ikili karşılaştırma yapılmıştır. Örnek anket giriş ekranı Şekil 5.23'te gösterilmiştir.



Şekil 5.23 : Anket giriş ekran görüntüsü.

AHP yönteminde tutarlılığın daha önceki bölümde de bahsedildiği gibi 0,1 değerine eşit veya küçük olması gereklidir. Anket girişleri yapıldıktan sonra , verilen cevapların tutarlılığını incelemek için programda "Basic Inconsistency Report" bölümünden faydalanılmıştır. Bu rapor doğrultusunda tutarsızlık değerleri 0,1 den küçük olduğu için faktörlerin tekrar değerlendirilmesine gerek kalmamıştır. Şekil 5.24'te modele dair örnek bir tutarsızlık raporu görünmektedir.



Rank	Row	Col	Current Val	Best Val	Old Inconsist.	New Inconsist.	% Improvement
1.	Doküman İhtiyacı	Organizasyon Alışkanlı	4.000000	1.983224	0.018673	0.014766	20.92 %
2.	Müşteri Memnuniyeti	Müşterinin Süreç İçerisi	3.000000	1.601059	0.018673	0.015742	15.70 %
3.	Doküman İhtiyacı	Müşteri Talebinin Doğru	5.999988	10.054840	0.018673	0.016628	10.95 %
4.	Doküman Sayısı	Müşteri Talebinin Doğru	5.999988	9.785023	0.018673	0.016997	8.98 %
5.	Müşterinin Süreç İçerisi	Organizasyon Alışkanlı	2.000000	1.262509	0.018673	0.017179	8.00 %
6.	Müşteri Memnuniyeti	Müşteri Talebinin Doğru	2.000000	1.372384	0.018673	0.017523	6.16 %
7.	Karmaşıklık	Organizasyon Alışkanlı	2.000000	1.317496	0.018673	0.017582	5.84 %
8.	Doküman Sayısı	Organizasyon Alışkanlı	3.000003	2.111799	0.018673	0.017590	5.80 %
9.	Müşteri Talebinin Doğru	Organizasyon Alışkanlı	3.000000	4.121391	0.018673	0.017850	4.41 %
10.	Müşteri Talebinin Doğru	Planlama	2.000000	1.483890	0.018673	0.017869	4.30 %
11.	Müşteri Talebinin Doğru	Müşterinin Ek Taleplerin	2.000000	1.483890	0.018673	0.017869	4.30 %
12.	Doküman İhtiyacı	Müşterinin Süreç İçerisi	4.000000	3.073723	0.018673	0.017961	3.81 %
13.	Doküman Sayısı	Müşterinin Süreç İçerisi	4.000000	2.994217	0.018673	0.018005	3.58 %
14.	Doküman İhtiyacı	Müşteri Memnuniyeti	5.000000	6.323207	0.018673	0.018208	2.49 %
15.	Doküman Sayısı	Karmaşıklık	4.000000	3.122616	0.018673	0.018251	2.26 %
16.	Organizasyon Alışkanlı	Planlama	3.000003	2.290989	0.018673	0.018287	2.07 %
17.	Müşterinin Ek Taleplerin	Organizasyon Alışkanlı	3.000000	2.290989	0.018673	0.018287	2.07 %
18.	Karmaşıklık	Müşterinin Ek Taleplerin	2.000000	1.601846	0.018673	0.018326	1.86 %
19.	Karmaşıklık	Planlama	2.000000	1.601846	0.018673	0.018326	1.86 %
20.	Doküman Sayısı	Müşteri Memnuniyeti	5.000000	6.157526	0.018673	0.018363	1.66 %
21.	Doküman İhtiyacı	Karmaşıklık	4.000000	3.207687	0.018673	0.018395	1.49 %
22.	Doküman İhtiyacı	Planlama	5.000000	5.852532	0.018673	0.018416	1.38 %
23.	Doküman İhtiyacı	Müşterinin Ek Taleplerin	5.000000	5.852531	0.018673	0.018416	1.38 %
24.	Karmaşıklık	Müşteri Talebinin Doğru	3.000003	2.577173	0.018673	0.018440	1.25 %
25.	Müşteri Memnuniyeti	Organizasyon Alışkanlı	3.000000	2.476272	0.018673	0.018463	1.12 %
26.	Karmaşıklık	Müşteri Memnuniyeti	2.000000	1.731216	0.018673	0.018490	0.98 %
27.	Müşterinin Ek Taleplerin	Müşterinin Süreç İçerisi	2.000000	1.670839	0.018673	0.018491	0.97 %
28.	Müşterinin Süreç İçerisi	Planlama	2.000000	1.670839	0.018673	0.018491	0.97 %
29.	Doküman Sayısı	Planlama	5.000000	5.699299	0.018673	0.018522	0.81 %
30.	Doküman Sayısı	Müşterinin Ek Taleplerin	5.000000	5.699298	0.018673	0.018522	0.81 %
31.	Müşteri Talebinin Doğru	Müşterinin Süreç İçerisi	3.000000	2.687686	0.018673	0.018568	0.56 %
32.	Müşteri Memnuniyeti	Müşterinin Ek Taleplerin	1.000000	1.080437	0.018673	0.018649	0.13 %
33.	Müşteri Memnuniyeti	Planlama	1.000000	1.080436	0.018673	0.018649	0.13 %
34.	Karmaşıklık	Müşterinin Süreç İçerisi	1.000000	1.042817	0.018673	0.018649	0.13 %
35.	Doküman İhtiyacı	Doküman Sayısı	1.000000	1.026967	0.018673	0.018661	0.07 %
36.	Müşterinin Ek Taleplerin	Planlama	1.000000	1.000000	0.018673	0.018673	0.00 %

Şekil 5.24 : Tutarsızlık raporu.

Çizelge 5.10'da kıyaslamada kullanılacak ana faktörlerin ikili karşılaştırma sonucu görülmektedir. En yüksek önceliği 0,33329 ile Maliyet alırken en az önceliği 0,12535 ile Kapsam almıştır.

Çizelge 5.10 : Ana faktörlerin etki dereceleri.

Tutarsızlık indeksi: 0,0304 < 0,1	
ANA FAKTÖRLER	**ÖNCELİK**
Maliyet	**0,33329**
Kalite	**0,30645**
Zaman	**0,23490**
Kapsam	**0,12535**

5.4.1 Seviye 1 alt faktörlerin ikili karşılaştırması

Maliyete en büyük etkiyi yapan alt faktör 0,75002 ile Süreç Maliyetleri'dir.

Çizelge 5.11 : Maliyet ana kriteri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0000 < 0,1	
Alt Faktör	**Öncelik**
Süreç Maliyetleri	0,75002
Dış Kaynak Maliyetleri	0,24998

Kaliteye en büyük etkiyi yapan faktör 0,66667 ile Analiz Tasarım Kalitesi'dir.

Çizelge 5.12 : Kalite ana kriteri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0000 < 0,1	
Alt Faktör	**Öncelik**
Analiz Tasarım Kalitesi	0,66667
Yazılım Kalitesi	0,33333

Zaman ana kriterine en büyük etkiyi yapan faktör 0,75 ile Teslim Süreleri'dir.

Çizelge 5.13 : Zaman ana kriteri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0000 < 0,1	
Alt Faktör	**Öncelik**
Teslim Süreleri	0,75000
Zaman Planlaması	0,25000

Kapsam kriterine en büyük etkiyi yapan faktör 0,658657 ile İç Etkenler'dir.

Çizelge 5.14 : Kapsam ana kriteri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0280 < 0,1	
Alt Faktör	**Öncelik**
İç Etkenler	0,658657
Dış Etkenler	0,185164
Takımın Durumu	0,156180

5.4.2 Seviye 2 alt faktörlerin ikili karşılaştırması

Süreç maliyetlerine en büyük etkiyi yapan faktör 0,441308 öncelik seviyesi ile Test Maliyeti'dir. En küçük etkiyi ise; 0,042727 öncelik seviyesi ile Yaygınlaştırma Maliyeti yapar.

Çizelge 5.15 : Süreç maliyetleri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0675 < 0,1	
Alt Faktör	**Öncelik**
Test Maliyeti	0,441308
Yazılım Maliyeti	0,214852
Analiz Maliyeti	0,135988
Tasarım Maliyeti	0,103982
Destek Maliyeti	0,061143
Yaygınlaştırma Maliyeti	0,042727

Dış Kaynakmaliyetlerine en büyük etkiyi yapan faktör 0,625026öncelik seviyesi ile Yazılım Maliyeti'dir. En küçük etkiyi ise; 0,136498öncelik seviyesi ile Analiz Maliyeti yapar.

Çizelge 5.16 : Dış kaynak maliyetleri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0176 < 0,1	
Alt Faktör	**Öncelik**
Dış Kaynak Yazılım Maliyeti	0,625026
Dış Kaynak Destek Maliyeti	0,238476
Dış Kaynak Analiz Maliyeti	0,136498

Yazılım kalitesine en büyük etkiyi yapan faktör 0,230363 öncelik seviyesi ile Test Kolaylığı'dır. En küçük etkiyi ise; 0,037363 öncelik seviyesi ile Genişletilebilirlik yapar.

Çizelge 5.17 : Yazılım kalitesi alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0243< 0,1	
Alt Faktör	**Öncelik**
Test Kolaylığı	0,230363
Çalışabilirlik	0,223727
Kod Kalitesi	0,215941
Kullanılabilirlik	0,132776
Güvenilirlik	0,103793
Esneklik	0,056037
Genişletilebilirlik	0,037363

Analiz Tasarım kalitesine en büyük etkiyi yapan faktör 0,481438 öncelik seviyesi ile Kullanım Kolaylığı'dır. En küçük etkiyi ise; 0,092257 öncelik seviyesi ile Bütünlük yapar.

Çizelge 5.18 : Analiz tasarım kalitesi alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0304< 0,1	
Alt Faktör	**Öncelik**
Kullanım Kolaylığı	0,481438
Doküman Kalitesi	0,295404
Doğruluk	0,130901
Bütünlük	0,092257

Teslim Sürelerine en büyük etkiyi yapan faktör 0,302366 öncelik seviyesi ile Test Süresi'dir. En küçük etkiyi ise; 0,038224 öncelik seviyesi ile Yaygınlaştırma Süresi yapar.

Çizelge 5.19 : Teslim süreleri alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0202 < 0,1	
Alt Faktör	**Öncelik**
Test Süresi	0,302366
Destek Süresi	0,292905
Yazılım Süresi	0,187721
Analiz Süresi	0,120782
Tasarım Süresi	0,058002
Yaygınlaştırma Süresi	0,038224

Zaman Planlamasına en büyük etkiyi yapan faktör 0,750019 öncelik seviyesi ile Öncelikli İşlerin Teslim Süreleri'dir. En küçük etkiyi ise; 0,249981 öncelik seviyesi ile Fazla Mesai Süreleri yapar.

Çizelge 5.20 : Zaman planlaması alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0000< 0,1	
Alt Faktör	**Öncelik**
Öncelikli İşlerin Teslim Süreleri	0,750019
Fazla Mesai Süreleri	0,249981

Dış Etkenlere en büyük etkiyi yapan faktör 0,508573öncelik seviyesi ile Proje İhtiyaçlarındaki Değişimin Sıklığı'dir. En küçük etkiyi ise; 0,065896öncelik seviyesi ile Müşteri ile İletişim yapar.

Çizelge 5.21 : Dış etkenler alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0592 < 0,1	
Alt Faktör	**Öncelik**
Proje İhtiyaçlarındaki Değişimin Sıklığı	0,508573
Müşteri ile Uyum	0,248966
Teknolojik Gelişmeler	0,176565
Müşteri ile İletişim	0,065896

İç Etkenlere en büyük etkiyi yapan faktör 0,240513 öncelik seviyesi ile Müşteri Talebinin Doğru Şekilde Karşılanması'dır. En küçük etkiyi ise; 0,026842 öncelik seviyesi ile Doküman İhtiyacı yapar.

Çizelge 5.22 : İç etkenler alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0187< 0,1	
Alt Faktör	**Öncelik**
Müşteri Talebinin Doğru Şekilde Karşılanması	0,240513
Müşteri Memnuniyeti	0,161326
Müşterinin Ek Taleplerinin Karşılanması	0,151843
Planlama	0,151843
Karmaşıklık	0,090316
Müşterinin Süreç İçerisinde Yer Alması	0,087399
Organizasyon Alışkanlıkları	0,062510
Doküman Sayısı	0,027407
Doküman İhtiyacı	0,026842

Takımın durumuna en büyük etkiyi yapan faktör 0,301183 öncelik seviyesi ile Takım İçi Organize Yapı'dır. En küçük etkiyi ise; 0,039168 öncelik seviyesi ile Takımdaki Üye Sayısı yapar.

Çizelge 5.23 : Takımın durumu alt faktörlerinin etki dereceleri.

Tutarsızlık indeksi: 0,0271 < 0,1	
Alt Faktör	**Öncelik**
Takım İçi Organize Yapı	0,301183
Takım Toplantılarının Verimliliği	0,251162
Takım Üyelerinin Motivasyonu	0,14395
Takım Üyelerinin Konsantrasyonu	0,140474
Takımın Tecrübesi	0,06377
Takımın Uyumu	0,060293
Takımdaki Üye Sayısı	0,039168

5.4.3 Kıyaslama modelinin kurulması ve sonuçlar

Şelale ve scrum metodolojilerinin kıyaslanmasında kullanılan ana faktörler ve bu faktörlere de etki eden alt faktörlerin etki seviyeleri AHP yöntemi ile ağırlıklandırılmıştır. Tüm bu faktörlerin birleşimi ile kıyaslamanın yapılacağı model kurulmuştur.

Modelde kriterler 1-5 skalasında değerlendirilmiştir. Bu değerlendirme yapılırken 1-5 skalasında her puan için karşılık gelen şartlar "Ölçüm Şekli" başlığı altında ayrıntılı olarak açıklanmıştır.

Oluşturulan modelle scrum ve şelale metodolojilerinin kıyaslanmasında, önceliklerin belirlenmesinde de çalışmaya katılan bilişim sektöründe çalışan 5 tecrübeli katılımcı yer almıştır. Bu 5 kişinin fikir birliğine dayalı olarak tüm kriterler scrum ve şelale metodolojileri için ayrı ayrı 1-5 skalasında puanlandırılmıştır. Yapılan bu çalışmaya katılan 5 katılımcının profili Çizelge 5.24'te gösterilmiştir.

Çizelge 5.24 : Scrum- şelale değerlendirmesine katılan uzmanların profili.

Unvan	Cinsiyet	Eğitim Durumu
Servis Yöneticisi	Erkek	Lisans
Grup Lideri	Erkek	Yüksek Lisans
Kıdemli Yazılım Uzmanı	Erkek	Lisans
Kıdemli Analiz Uzmanı	Kadın	Lisans
Analiz Uzmanı	Erkek	Yüksek Lisans

Scrum ve şelale karşılaştırılmasında verilen puanlar sonrası öncelik seviyeleri de kullanılarak ilk olarak alt faktörlerin seviyeleri belirlenmiştir. Sonuçlar çizelge de görülmektedir. Scrum metodolojisi Kalite ana faktörünü %92, Zaman ana faktörünü %89, Kapsam ana faktörünü %85 ve Maliyet ana faktörünü %77 karşılarken; Şelale metodolojisi Kalite'yi %66, Zaman'ı %51, Kapsam'ı %60 ve Maliyet'i %54 karşılamaktadır. Çizelge 5.25 ve 5.26'da ayrıntılı olarak sonuçlar gösterilmiştir.

Çizelge 5.25 : Scrum-şelale alt faktörler uygulama sonuçları.

SCRUM			
Metrikler	**Seviye (%)**	**Ağırlık**	**Toplam**
Süreç Maliyetleri	77%	0,75002	**77%**
Dış Kaynak Maliyetleri	77%	0,24998	
Yazılım Kalitesi	83%	0,33333	**92%**
Analiz Tasarım Kalitesi	96%	0,66667	
Teslim Süreleri	90%	0,75000	**89%**
Zaman Planlaması	85%	0,25000	
Dış Etkenler	85%	0,18516	**85%**
İç Etkenler	86%	0,65866	
Takımın Durumu	82%	0,15618	
ŞELALE (WATERFALL)			
Metrikler	**Seviye (%)**	**Ağırlık**	**Toplam**
Süreç Maliyetleri	51%	0,75002	**54%**
Dış Kaynak Maliyetleri	62%	0,24998	
Yazılım Kalitesi	73%	0,33333	**66%**
Analiz Tasarım Kalitesi	63%	0,66667	
Teslim Süreleri	50%	0,75000	**51%**
Zaman Planlaması	55%	0,25000	
Dış Etkenler	58%	0,18516	**60%**
İç Etkenler	59%	0,65866	
Takımın Durumu	68%	0,15618	

Alt faktörlerinde ağırlıkları kullanılarak yapılan hesaplamada ise sonuç olarak Scrum metodolojisi %85 ve şelale metodolojisi %58 puan ile çalışmayı tamamlamıştır. Scrum metodolojisi AHP yöntemiyle yapılan bu kıyaslamada Şelale metodolojisine üstün gelmiştir. Ayrıca tüm ana faktörlerde de tek tek şelale'a üstünlük sağlamıştır.

Çizelge 5.26 : Scrum-şelale ana faktörler uygulama sonuçları.

SCRUM			
Metrikler	**Seviye (%)**	**Ağırlık**	**Toplam**
Maliyet	77%	0,33329	**85%**
Kalite	92%	0,30645	
Zaman	89%	0,23490	
Kapsam	85%	0,12535	
ŞELALE (WATERFALL)			
Metrikler	**Seviye (%)**	**Ağırlık**	**Toplam**
Maliyet	54%	0,33329	**58%**
Kalite	66%	0,30645	
Zaman	51%	0,23490	
Kapsam	60%	0,12535	

5.5 Uygulama Sonuçlarının Analizi

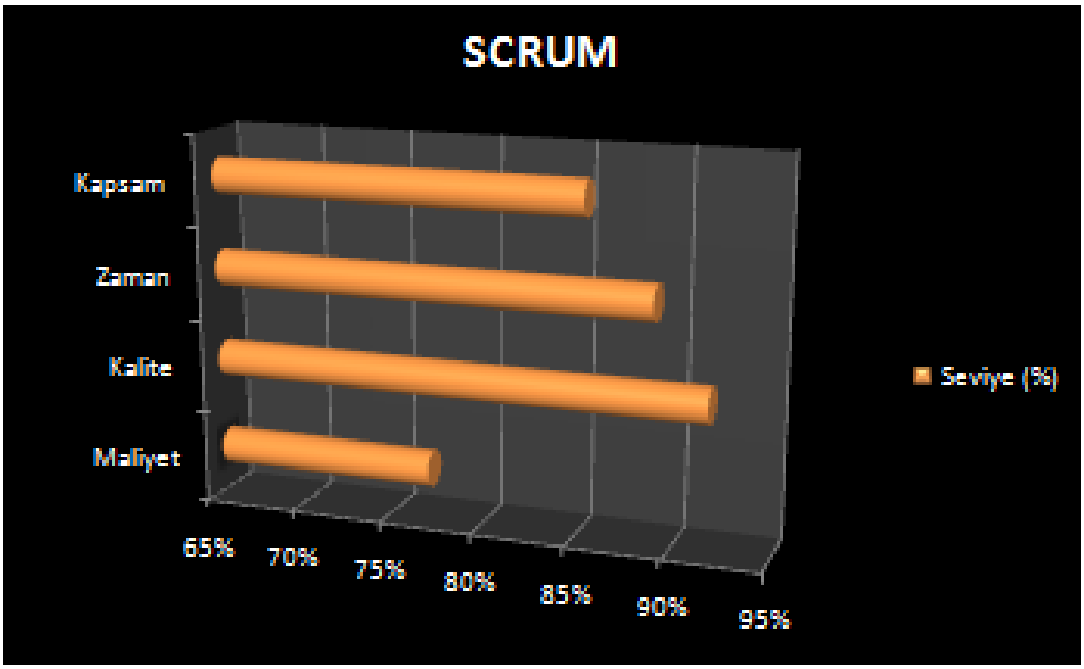
Çalışmanın başlangıcında scrum ve şelale metodolojilerinin kıyaslanmasında kullanılacak kriterlerin belirlenmesi için anket çalışmaları yapılmış ve kriterler belirlenmiştir. Daha sonra bu kriterler ana faktörler ve alt faktörler olarak düzenlenerek model kurulmuştur. Kurulan model super decision programına aktarılarak tecrübeli 5 kişilik uzman kadronun değerlendirmesi ile 1-9 skalasında faktörlerin birbirleri ile kıyaslanması sağlanmış ve öncelik seviyeleri belirlenmiştir. Sonrasında bu model üzerinden scrum ve şelale 1-5 skalasında oylanarak nihai sonuçlara ulaşılmıştır.

Ana faktörler arasında en yüksek öncelik seviyesi Maliyet faktöründeyken, 2. Sırada Kalite gelmektedir. En düşük öncelik seviyesindeki faktör Kapsam iken , en düşük 2. sırada ise Zaman faktörü bulunmaktadır. Maliyet faktörü alt faktörlerinden Süreç maliyetleri en yüksek öncelik seviyesindeyken, Kapsam faktörünün alt faktörlerinden Takımın Durumu en düşük öncelik seviyesindeki alt faktördür.

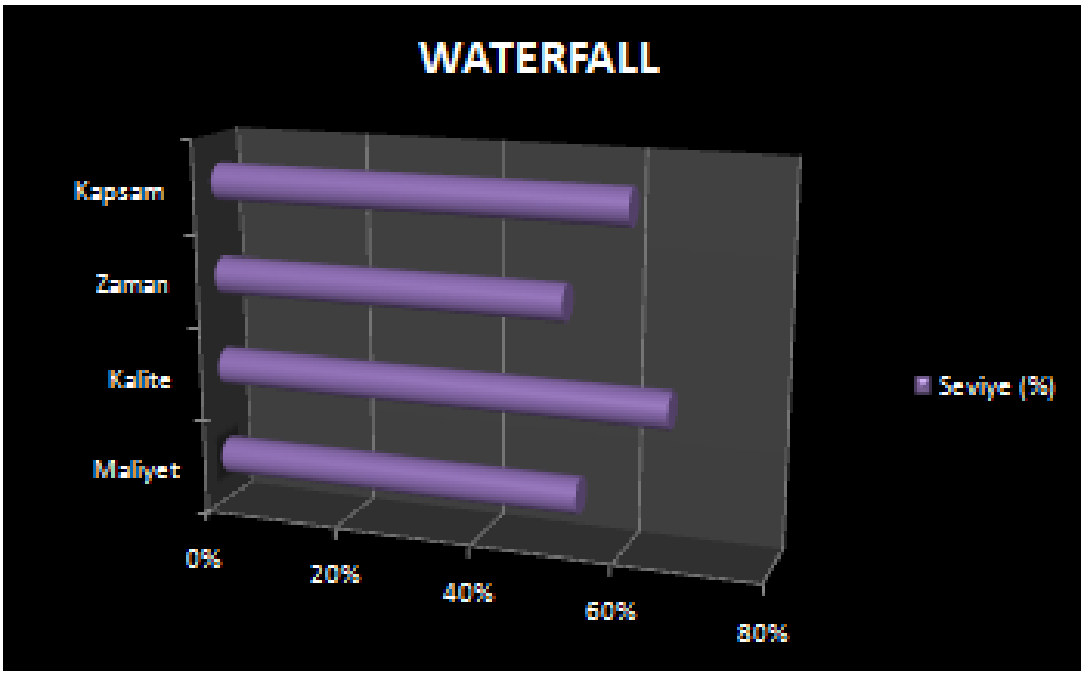
2.Seviye alt faktörleri incelediğimizde ise; Süreç maliyetleri içerisindeki Test maliyeti en yüksek öncelik seviyesin sahiptir. En düşük öncelik seviyesi ise Takımın Durumu faktörü içerisindeki Takımdaki üye sayısı faktörüne aittir.

Oluşturulan bu model üzerinden yapılan puanlamalar sonucunda Scrum metodolojisi Şelale metodolojisine Kalite ana faktöründe %92-%66, Zaman ana faktöründe %89-%51, Kapsam ana faktöründe %85-%60 ve Maliyet ana faktöründe %77-%54 üstünlük sağlamıştır. Genel değerlendirmede ise %85-%58'lik üstünlük sağlayarak AHP yöntemine göre tercih edilen yazılım metodolojisi olmuştur.

Şekil 5.25 ve 5.26'da Scrum ve Şelale metodolojilerinin 4 ana faktöre göre grafikleri gösterilmektedir.



Şekil 5.25 : Scrum ana faktörler uygulama sonucu.



Şekil 5.26 : Şelale ana faktörler uygulama sonucu.

6. SONUÇ VE ÖNERİLER

Yapılan çalışmada bilişim sektöründe kullanılan scrum ve şelale metodolojilerinin kıyaslanması için AHP yöntemi kullanılarak model oluşturulmuş ve uygulama çalışması yapılmıştır. Uygulama finans sektörüne çalışan ve 500'den fazla çalışanı olan bir yazılım şirketinde yapılmıştır. Anket çalışmaları, faktörlerin belirlenmesi, ağırlıklandırmaları ve metodolojiler arasındaki kıyaslamalar şirket çalışanları ile yapılmıştır. Şirket 2011 yazında aldığı kararla şelale metodolojisinden scrum metodolojisine geçiş yapmıştır. Bu nedenle şirket bünyesindeki yazılımcı ve analist kadrosu, aynı zamanda yönetim kadrosu hem şelale hem de scrum metodolojileri konusunda bilgi ve tecrübeye sahiptir. Şirket şuanda Türkiye'de scrum metodolojisini en kalabalık kadroyla uygulayan firmadır. Tez çalışmasında anket ve kıyaslamalarda her iki metodolojiyede hakim olan çalışanlar olduğundan dolayı objektif ve net sonuçlar alınmıştır.

AHP uygulamasını alt faktörlerden başlayarak incelememiz gerekirse;

Süreç Maliyetleri faktörü altında en yüksek etki Test maliyeti'nde, Dış Kaynak Maliyetleri faktörü altında ise Yazılım maliyeti'ndedir. Bu alt faktörlerin 1-5 skalasında aldıkları değerler ise; Scrum için Test maliyeti 4, Dış kaynak Yazılım Maliyeti 4; Şelale için ise Test Maliyeti 2, Dış Kaynak Yazılım Maliyeti ise 3'tür. Ayrıca scrum süreç maliyetleri faktöründe %77-%51, dış kaynak maliyetlerinde %77-%62 oranları ile şelale metoduna üstünlük sağlamıştır. İki metodoloji arasındaki bu fark planlamadan kaynaklanmaktadır. Scrumda işlerin küçük periyotlarda, küçük tasklar halinde yapılması ve toplantıların sıklığı planlı bir şekilde ilerlenmesini sağlarken maliyetlerinde düşük olmasını sağlamaktadır. Şelale'da ise planların daha geniş sürelerde ve geniş başlıklar altında yapılması nedeniyle planlama dışı faaliyetlere maruz kalmakta ve maliyetlerin yükseldiği görülmektedir.

Yazılım Kalitesi faktörü altında en yüksek etki Test Kolaylığı'nda, Analiz Tasarım Kalitesi faktörü altında Kullanım Kolaylığı'ndadır. Bu alt faktörlerin 1-5 skalasında aldıkları değerler ise; Scrum için Test Kolaylığı 4, Kullanım Kolaylığı 5; Şelale için

ise Test Kolaylığı 4, Kullanım Kolaylığı 3. Ayrıca scrum Yazılım Kalitesi faktöründe %83-%73, Analiz Tasarım Kalitesi faktöründe %96-%63 oranları ile şelale'a üstünlük sağlamıştır. Kalite faktörlerinde açık şekilde görülmektedirki, bilişim sektöründe yapılan işin kalitesi en çok test ve kullanım kolaylığına bağlıdır. Burada scrum'ın şelalea üstünlüğü sonuçlara yansımıştır, özellikle analiz tasarım kalitesinde 33 puanlık ciddi fark göze çarpmaktadır. Scrum metodolojisinde bu faktörde yakalanan %96 oranı da kalitenin ne kadar yükseldiğinin açık bir göstergesidir.

Teslim Süreleri faktörü altında en yüksek etki Test Süresi'nde, Zaman Planlaması faktörü altında Öncelikli İşlerin Teslim Süreleri'ndedir. Bu alt faktörlerin 1-5 skalasında aldıkları değerler ise; Scrum için Test Süresi 5, Öncelikli İşlerin Teslim Süreleri 4; Şelale için ise Test Süresi 2, Öncelikli İşlerin Teslim Süreleri 3. Ayrıca scrum Teslim Süreleri faktöründe %90-%50, Zaman Planlaması faktöründe %85-%55 oranları ile şelale'a üstünlük sağlamıştır. İki metodoloji arasındaki en net fark zaman ana faktörü altında ortaya çıkmıştır. Teslim sürelerinde 40 puan, zaman planlamasında 30 puanlık fark vardır. Bu farkın temel nedeni scrumdaki planlama yöntemleri ve sık aralıklarla ortaya ürün çıkarılmasıdır. Şelale'da ise işler uzun sürelere yayıldığından ve bu süreler içerisinde işin içeriğinde değişiklikler olmasından dolayı süreler uzamaktadır.

Kapsam dış etkenler faktörü altında en yüksek etki Proje İhtiyaçlarındaki Değişimin Sıklığı'nda, Kapsam iç etkenler faktörü altında Müşteri Talebinin Doğru Şekilde Karşılanması'nda, Takımın durumu faktörü altında Takım İçi Organize Yapı'dadır. Bu alt faktörlerin 1-5 skalasında aldıkları değerler ise; Scrum için Proje İhtiyaçlarındaki Değişimin Sıklığı 4, Müşteri Talebinin Doğru Şekilde Karşılanması 5, Takım İçi Organize Yapı 4; Şelale için ise Proje İhtiyaçlarındaki Değişimin Sıklığı 3, Müşteri Talebinin Doğru Şekilde Karşılanması 4, Takım İçi Organize Yapı 3. Ayrıca scrum Kapsam dış etkenler faktöründe %85-%58, Kapsam iç etkenler faktöründe %86-%59, Takımın durumu faktöründe %82-%68 oranları ile şelale'a üstünlük sağlamıştır. Scrum sıklıkla ürün çıkartması nedeniyle proje ihtiyaçlarındaki değişikliklere çabuk cevap verebilmektedir. Çevik yapısı gereği değişime çabuk uyum sağlar. Bu değişimler aynı zamanda müşteri talebinin doğru şekilde karşılanamamasına da neden olmakta ve memnuniyetsizlikle sonuçlanmaktadır. Scrumda müşteri her an işin içerisindedir ve tüm süreçlerde bulunmaktadır. Bu da müşteri odaklı, hızlı ve doğru bir çalışma yapılmasını ve memnuniyetle karşılanan

işlerin ortaya çıkmasını sağlamaktadır. Takımın sık toplantı yapması ve tüm takım üyelerinin tüm işlerden haberdar olması takımın uyumunu ve yapının organizr bir şekilde çalışmasını sağlamaktadır. İşler bireysel bazlı değilde takım mantığında yapıldığından zamanla hem çalışanların işler konusundaki yetkinlikleri ve tecrübeleri artmakta hem de bir kişinin eksikliğinde diğer takım üyeleri boşluğu kapatabilmektedir.

Ana faktörlerin önem sırası ise şu şekildedir; maliyet, kalite, zaman ve kapsam. Scrum AHP yöntemine göre; alt faktörlerde olduğu gibi ana faktörlerinde tamamında şelale metoduna üstünlük sağlamıştır. Maliyet'te %77-%54, Kalite'de %92-%66, Zaman'da %89-%51 ve Kapsam'da %85-%60 oranları ile kıyaslamada scrum önde gelmiştir. Tüm ana ve alt faktörlerin puanlaması ve ağırlıklandırması hesaplanarak ulaşılan nihai sonuçta ise Scrum %85 – Şelale %58lik başarı oranı sağlanmıştır.

Uygulamanın yapıldığı şirket 2011 yazında itibaren Scrum ile çalışmaya başlamıştır ve uygulama çalışmasında da görüldüğü gibi Scrum ile %85lik bir başarı oranı yakalanmıştır. Bu oran zamanla daha da yükseltilebilir.

Şirketin scrum geçmişi çok eskiye dayanmadığı için henüz oturmamış uygulamalarda bulunmaktadır. Şirkette çok kalabalık scrum takımları bulunmaktadır. Takımların büyüklüğü günlük scrum toplantılarının ve sprint toplantılarının daha da uzamasına neden olmakta ve yapılacak işlerin daha küçük tasklara ayrılmasını engellemektedir. Ayrıca toplantıların fazla uzaması motivasyon ve iş kayıplarına da yol açabilmektedir. Bu nedenle ilk olarak şirket içindeki scrum takımlarının sayılarının en fazla 8 kişi olacak şekilde düşürülmesi gerekmektedir.

Şirketin işleri teslim ettiği müşterilerde işin en önemli parçalarıdır fakat scrum metodolojisine geçildiği ve şirket içi eğitimler verildiği halde, müşteri olan finans kurumuna bu değişimle ilgili eğitimler verilmemiştir. Müşterinin scrum konusunda bilgisiz olması da iletişim ve uyum konusunda sorunlara yol açmaktadır. Scrumın temel özelliklerinden biri kullanıcılarla iletişimin güçlü olmasıdır fakat bilgi eksikliğinden ve tecrübesizlikten kaynaklanan bu tür sorunlarla ilk zamanlar karşılaşılmaktadır. Şirketin müşterilerine de scrum konusunda eğitim vererek müşterininde bu metodolojinin içerisine girmesi sağlanmalı ve verimlilik artırılmalıdır.

KAYNAKLAR

[1] **Elliot, G.** (2004). *Global Business Information Technology: An integrated systems approach.* Pearson Education. p.87.

[2] **Rossberg, J.** (2008). *Pro Visual Studio Team System Application Lifecycle Management.* Apress.

[3] **Cockburn, A.** (2001). *Agile Software Development,* Addison-Wesley Longman.

[4] **Url-1** < *http://agilemanifesto.org*>, alındığı tarih: 11.06.2011.

[5] **Yates, J. F.** (2003). *Decision Management – How to Assure Better Decisions In Your Company*, University of Michigan Business School Management Series, Jossey-Bass, San Francisco.

[6] **Büyükyazıcı, M.** (2000). *Analitik Ağ Süreci* (Yüksek Lisans Tezi), Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.

[7] **Highsmith, J.** (2002). *Agile Software Development Ecosystems,* Addison Wesley.

[8] **Koçel, T.** (2003). *İşletme Yöneticiliği*, Beta Basım, 9. Basım, İstanbul.

[9] **Schwaber, K.** (1995). *The Scrum development process*, OOPSLA'95, Texas, USA.

[10] **Jacobson, I.** (1999). *The Unified Software Development Process*. Addison-Wesley.

[11] **Url-2** <*http://en.wikipedia.org/wiki/Microsoft_Solutions_Framework*>, alındığı tarih 14.08.2011.

[12] **Turner, M.** (2006). *Microsoft Solutions Framework Essentials,* Microsoft Press.

[13] **Ambler, S.** (2002). *Agile Modeling: Effective Practices for extreme Programming and the Unified Process,* Wiley Computer Publishing.

[14] **Lorcu, F.** (2000). *Analitik Hiyerarşi Prosesi Tekniği ile Kişisel Bilgisayar Tercihi Konusunda Bir Uygulama* (Yüksek Lisans Tezi), İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü, İstanbul.

[15] **Harp Akademileri Komutanlığı Yayınları.** (2001). Karar Verme ve Problem Çözme, İstanbul.

[16] **Tekin, M.** (1999). *Kantitatif Karar Verme Teknikleri*, 4. Baskı, Kuzucular Ofset, Konya.

[17] **Eren, E.** (2001). *Yönetim ve Organizasyon*, 5. Baskı, Beta Basım, İstanbul, Ocak 2001.

[18] **Evren, R. ve Ülengin, F.** (1992). *Yönetimde Çok Amaçlı Karar Verme*, İTÜ Matbaası, İstanbul

[19] **Clemen, R. T.** (1996). *Making Hard Decisions – An Introduction To Decision Analysis*, 2nd Edition, Duxbury Press, California.

[20] **İraz, R.** (2006). Organizasyonlarda Karar Verme ve İletişim Sürecinin Etkinliği Bakımından Bilgi Teknolojilerinin Rolü.

[21] **Esin, A.** (2003). *Yöneylem Araştırmalarında Yararlanılan Karar Yöntemleri*, Gazi Kitabevi, Ankara

[22] **Falino, D. F.** (2003). *Etkili Karar Verme*, Hayat Yayıncılık,İstanbul.

[23] **Clemen, R. T.** (1996). *Making Hard Decisions – An Introduction To Decision Analysis*, 2nd Edition, Duxbury Press, California.

[24] **Hammond, J. S., Keeney, R. L. ve Raiffa, H.** (1998). *Karar Verme Sanatı*, 1. Basım, Beyaz Yayınları, İstanbul.

[25] **Gökçen, H.** (2002). *Yönetim Bilgi Sistemleri – Analiz ve Tasarım Perspektifi*, Epi Yayıncılık, Ankara.

[26] **Marakas, G. M.** (2003). *Decision Support Systems In The 21st Century*, 2nd Edition, Prantice Hall, New Jersey.

[27] **Kocamaz, M. ve Soyuer, H.** (2003). İsletmelerde Bilgisayar Destekli İnsan Kaynağı Değerlendirme ve Seçme Süreci.

[28] **Yağcı, A.** (2002). *Analitik Hiyerarşi Proses Yöntemi ve Tedarikçi Seçimi Probleminde Bir Uygulaması* (Yüksek Lisans Tezi), Ankara Üniversitesi, Sosyal Bilimler Enstitüsü, Ankara.

[29] **Esin, A.** (2003). *Yöneylem Araştırmalarında Yararlanılan Karar Yöntemleri*, Gazi Kitabevi, Ankara.

[30] **Taha, H. A.** (1997). *Operations Research an Introduction*, Upper Saddle River, N.J., Prentice Hall.

[31] **Dağdeviren, M.** (2005). *Performans Değerlendirme Sürecinin Çok Ölçütlü Karar Verme Yöntemleri ile Bütünleşik Modellenmesi* (Doktora Tezi) Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara .

[32] **Tikindt, V. ve Billaut, J. C.** (2002). *Multicriteria Scheduling Theory Models and Algorithms*, Springer Press, New York.

[33] **Umarusman, N.** (2002). *Bulanık Çok Amaçlı Hedef Programlama ve Bir Üretim Süreci Uygulaması* (Yüksek Lisans Tezi), Dokuz Eylül Üni.Sos.Bil.Ens., İzmir.

[34] **Herisçakar, E.** (1999). Gemi Ana Makine Seçiminde Çok Kriterli Karar Verme Yöntemleri AHP ve SMART Uygulaması, *Gemi İnşaatı ve Deniz Teknolojisi Teknik Kongresi*, İstanbul.

[35] **Triantaphyllou, E.** (2000). *Multi – Criteria Decision Making Methods : A Comperative Study*, Kluwer Academic Publishers, Dordrecht.

[36] **Özdemir, A.** (2004). *Yönetsel Karar Verme Sürecinde Dinamik Amaç Programlama Yaklaşımı ve Bir Uygulama* (Doktora Tezi), Dokuz Eylül Üni.Sos.Bil.Ens., İzmir.

[37] **Bölat, B. ve Kuzucu, A.** (2006). Çok Amaçlı Karar Verme Problemlerine Etkileşimli Bir Yaklaşım, *İTÜ Dergisi*, Cilt:5, Sayı:1, Kısım:1.

[38] **Ulucan, A.** (2004). *Analitik Hiyerarşi Süreci ve Uygulamaları*, Ankara.

[39] **Miller, D. W. ve Starr, M. K.** (1969). *Executive Decisions and Operations Research*, Prentice- Hall, Inc., N.J.

[40] **Vargas, L. G.** (1990). An Overview of the Analytic Hierarchy Process and its Applications, European Journal of Operational Research.

[41] **Buchanan, J. ve Sheppard, P.** (2006). Ranking Projects Using the ELECTRE Method.

[42] **Serinkaya, O.** (2001). *Çok Kriterli Karar Destek Sistemi ELECTRE Yöntemleri Üzerine Bir Uygulama* (Yüksek Lisans Tezi), Gazi Üni. Fen Bil.Ens., Ankara.

[43] **Kaya, Y. ve Kahraman, C.** (2004). *Çok Amaçlı Karar Verme Yöntemlerinden TOPSIS ve ELECTRE Yöntemlerinin Karşılaştırılması*, Havacılık ve Uzay Teknolojileri Enstitüsü, İstanbul.

[44] **Spee, B.** (2005). *Multi-Criteria Decision Making An Application Study of ELECTRE &TOPSIS.*

[45] **Saaty, T.** (1987). *Concepts, Theory and Techniques, Rank Generation, Preservation and Reversal in the Analytic Hierarchy Decision Process, Decision Sciences.*

[46] **Akal, E.** (2006). *Yükletenlerin Lojistik Hizmet Sağlayıcı Seçim Kriterlerinin Belirlenmesi : Kimya Sektörü Üzerinde Bir Uygulama* (Yüksek Lisans Tezi), Dokuz Eylül Üniversitesi Sosyal Bilimler Enstitüsü.

[47] **Saaty, T. L.** (1999). *Decision Making For Leaders : The Analytic Hierarchy For Decisions In A Complex World*, 3rd Edition, RSW Publishers, San Francisco, Pittsburg.

[48] **Kuruüzüm, A. ve Atsan, N.** (2006). Analitik Hiyerarşi Yöntemi ve İşletmecilik Alanındaki Uygulamaları, *Akdeniz B.F. Dergisi*.

[49] **Saaty, T.** (2006). *How To Make A Decision*.

[50] **Saaty, T.** (2006). *The Seven Pillars of The Analytic Hierarchy Process*.

[51] **Hacımen, E.** (1998). *Analitik Hiyerarşi Süreci ve Bilişim Teknolojisi Kararlarında Uygulaması* (Doktora Tezi), Dokuz Eylül Üni. Sos. Bil. Ens., İzmir.

[52] **Erikan, L.** (2002). *Hv. K. K.lığı'nda Aday Seçiminde Analitik Hiyerarşi Prosesi ile Etkin Karar Verme* (Yüksek Lisans Tezi), İTÜ, Fen Bilimleri Enstitüsü, İstanbul.

[53] **Aktaş, R. ve Doğanay, M.** (2000). Personel Seçiminde Analitik Hiyerarşi Modelinin Kullanılması, *İnsan Kaynakları Yönetimi Sempozyumu*, Konya.

[54] **Saaty, T.** (2001). Deriving the AHP 1-9 Scale from First Principles, *ISAHP 2001*, Berne, SWITZERLAND.

[55] **Saaty, T.** (1980). *The Analytic Hierarchy Process*, McGraw-Hill, New York.

[56] **Taha, H. A.** (2002). *Yöneylem Araştırması*, (Çeviren : S. Alp BARAY, Sakir ESNAF), 6. Basım, İstanbul Üniversitesi, İstanbul.

[57] **Saaty, T.** (2000). *Fundamentals of Decision Making and Priority Theory With The Analytic Hierarchy Process*, Pittsburg.

[58] **Golden, B. L., Wasil, E. A. ve HARKER, P. T.** (1989). *The AHP Applications and Studies*, Springer-Verlag, Berlin.

[59] **SEI CMMI Product Team**. (2006). CMMI for Development Version 1.2, Software Engineering Institute, Carnegie Mellon University.

[60] **Kniberg, H.** (2007). *Scrum and XP from the Trenches*, InfoQ.

EKLER

EK A : Scrum-şelale kıyaslama faktörleri belirleme anketi 1
EK B : Scrum-şelale kıyaslama faktörleri belirleme anketi 2
EK C : Scrum-şelale etki düzeyi anket sonucu
EK D : Scrum uygulama değerlendirme sonuçları
EK E : Şelale uygulama değerlendirme sonuçları

EK A

SCRUM-ŞELALE KIYASLAMA FAKTÖRLERİ BELİRLEME ANKETİ 1

İsim-Soyisim:
Unvanı:
Telefonu:

Scrum ve Şelale metodolojilerinin kıyaslanması ile ilgili yapılacak bir tez çalışmasında, kıyaslama için hangi faktörlerin kullanılması gerekmektedir?

Şekil A.1 : Scrum-şelale kıyaslama faktörleri belirleme anketi 1.

EK B

SCRUM-ŞELALE KIYASLAMA FAKTÖRLERİ BELİRLEME ANKETİ 2

İsim-Soyisim:
Unvanı:
Telefonu:

Aşağıda verilen kriterler Scrum ve Şelale metodolojilerinin kıyaslanmasında kullanılacaktır. Her alan için belirlenen kriterlerin bu kıyaslamaya etki düzeyini belirleyiniz. Seçtiğiniz düzeyin ilgili alanına "x" koymanız gerekmektedir.

Etki Düzeyi	Etki İfadesi
1	**Hiçbir etkisinin yoktur**
2	**Düşük etkilidir,** bu madde olmasa da konu yürütülebilir
3	**Orta seviyede bir etkisi vardır.** (Üst başlığın konularından birisidir, ancak bu madde olmadan daseviye ölçülebilir)
4	**Etkilidir**, bu madde olmadan ilgili konunun seviyesi ölçülemez.
5	**Doğrudan etkilidir** ve başlığın en temel konularından birisidir

Örnek:

1	2	3	4	5
	x			
		x		

Şekil B.1 : Scrum-şelale kıyaslama faktörleri belirleme anketi 2.

Çizelge B.1 : Kıyaslama faktörleri tablosu.

	Kriterler	1	2	3	4	5
1	Maliyet					
1.1	Süreç Maliyetleri					
1.1.1	Analiz Maliyeti					
1.1.2	Tasarım Maliyeti					
1.1.3	Test Maliyeti					
1.1.4	Yazılım Maliyeti					
1.1.5	Yaygınlaştırma Maliyeti					
1.1.6	Destek Maliyeti					
1.2	Dış Kaynak Maliyetleri					
1.2.1	Dış Kaynak Analiz Maliyeti					
1.2.2	Dış Kaynak Yazılım Maliyeti					
1.2.3	Dış Kaynak Destek Maliyeti					
2	Kalite					
2.1	Analiz Tasarım Kalitesi					
2.1.1	Doğruluk					
2.1.2	Kullanım Kolaylığı					
2.1.3	Etkinlik					
2.1.4	Bütünlük					
2.1.5	Doküman kalitesi					
2.2	Yazılım Kalitesi					
2.2.1	Güvenilirlik					
2.2.2	Kullanılabilirlik					
2.2.3	Kod kalitesi					
2.2.4	Test Kolaylığı					
2.2.5	Genişletilebilirlik					
2.2.6	Esneklik					
2.2.7	Tekrar Kullanılabilirlik					
2.2.8	Çalışabilirlik					
3	Zaman					
3.1	Teslim Süreleri					
3.1.1	Analiz Süresi					
3.1.2	Tasarım Süresi					
3.1.3	Test Süresi					
3.1.4	Yazılım Süresi					
3.1.5	Yaygınlaştırma Süresi					
3.1.6	Destek Süresi					
3.2	Zaman Planlaması					
3.2.1	Öncelikli işlerin teslim süreleri					
3.2.2	Fazla mesai süreleri					
4	Kapsam					
4.1	İç Etkenler					

Çizelge B.1 (devam) : Kıyaslama faktörleri tablosu

4.1.1	Karmaşıklık					
4.1.2	Organizasyon Alışkanlıkları					
4.1.3	Planlama					
4.1.4	Müşteri talebinin doğru şekilde karşılanması					
4.1.5	Müşterinin ek taleplerinin karşılanması					
4.1.6	Müşterinin süreç içerisinde yer alması					
4.1.7	Müşteri memnuniyeti					
4.1.8	Doküman ihtiyacı					
4.1.9	Doküman sayısı					
4.2	Takımın Durumu					
4.2.1	Takım üyelerinin konsantrasyonu					
4.2.2	Takım üyelerinin motivasyonu					
4.2.3	Takım toplantılarının verimliliği					
4.2.4	Takım içi organize yapı					
4.2.5	Takımın tecrübesi					
4.2.6	Takımdaki üye sayısı					
4.2.7	Takımın uyumu					
4.3	Dış Etkenler					
4.3.1	Kültürel değişiklikler					
4.3.2	Müşteri ile uyum					
4.3.3	Teknoloji gelişimi					
4.3.4	Proje ihtiyaçlarındaki değişimin sıklığı					
4.3.5	Müşteri ile iletişim					

EK C

Çizelge C.1 : Scrum-şelale etki düzeyi anket sonucu.

No	Kriterler	Etki Düzeyi
1	Maliyet	4,59
2	Müşteri talebinin doğru şekilde karşılanması	4,59
3	Yazılım Kalitesi	4,55
4	Kapsam	4,55
5	Müşterinin ek taleplerinin karşılanması	4,55
6	Müşterinin süreç içerisinde yer alması	4,55
7	Öncelikli işlerin teslim süreleri	4,5
8	Kalite	4,45
9	Zaman	4,45
10	Zaman Planlaması	4,45
11	Teslim Süreleri	4,32
12	Kod kalitesi	4,27
13	Fazla mesai süreleri	4,27
14	İç Etkenler	4,23
15	Proje ihtiyaçlarındaki değişimin sıklığı	4,23
16	Dış Kaynak Maliyetleri	4,18
17	Doküman ihtiyacı	4,14
18	Doküman kalitesi	4,09
19	Müşteri memnuniyeti	4,09
20	Dış Etkenler	4,09
21	Süreç Maliyetleri	4,05
22	Yazılım Maliyeti	4,05
23	Analiz Tasarım Kalitesi	4,05
24	Test Kolaylığı	4,05
25	Teknoloji gelişimi	4,05
26	Dış Kaynak Yazılım Maliyeti	3,86
27	Yazılım Süresi	3,82
28	Doküman sayısı	3,82
29	Dış Kaynak Destek Maliyeti	3,77
30	Test Maliyeti	3,73
31	Test Süresi	3,73
32	Destek Süresi	3,73
33	Kullanılabilirlik	3,64
34	Müşteri ile uyum	3,64
35	Destek Maliyeti	3,55
36	Planlama	3,55
37	Müşteri ile iletişim	3,45
38	Kullanım Kolaylığı	3,41
39	Çalışabilirlik	3,41
40	Tasarım Maliyeti	3,32
41	Analiz Süresi	3,32

Çizelge C.1 (devam) : Scrum-şelale etki düzeyi anket sonucu

42	Yaygınlaştırma Süresi	3,32
43	Tasarım Süresi	3,27
44	Analiz Maliyeti	3,23
45	Yaygınlaştırma Maliyeti	3,23
46	Karmaşıklık	3,18
47	Takım toplantılarının verimliliği	3,14
48	Dış Kaynak Analiz Maliyeti	3,09
49	Organizasyon Alışkanlıkları	3,09
50	Doğruluk	2,95
51	Bütünlük	2,95
52	Takım içi organize yapı	2,91
53	Güvenilirlik	2,86
54	Takım üyelerinin konsantrasyonu	2,86
55	Genişletilebilirlik	2,77
56	Takım üyelerinin motivasyonu	2,68
57	Takımın uyumu	2,68
58	Esneklik	2,59
59	Takımın tecrübesi	2,45
60	Takımdaki üye sayısı	2,32
61	Tekrar Kullanılabilirlik	1,59
62	Etkinlik	1,55
63	Kültürel değişiklikler	1,36

EK D

1. MALİYET												
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Süreç Maliyetleri	1	Analiz Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	4	0,135988	77%
						x					
	2	Destek Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	4	0,061143	
						x					
	3	Tasarım Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	3	0,103982	
					x						
	4	Test Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	4	0,441308	
						x					
	5	Yaygınlaştırma Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	3	0,042727	
					x						
	6	Yazılım Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	4	0,214852	
						x					

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Dış Kaynak Maliyetleri	1	Dış Kaynak Analiz Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	3	0,136498	77%
					x						
	2	Dış Kaynak Destek Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	4	0,238476	
						x					
	3	Dış Kaynak Yazılım Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2)Maliyet yüksek 3)Maliyet normal. 4)Maliyet düşük 5)Maliyet çok düşük	4	0,625026	
						x					

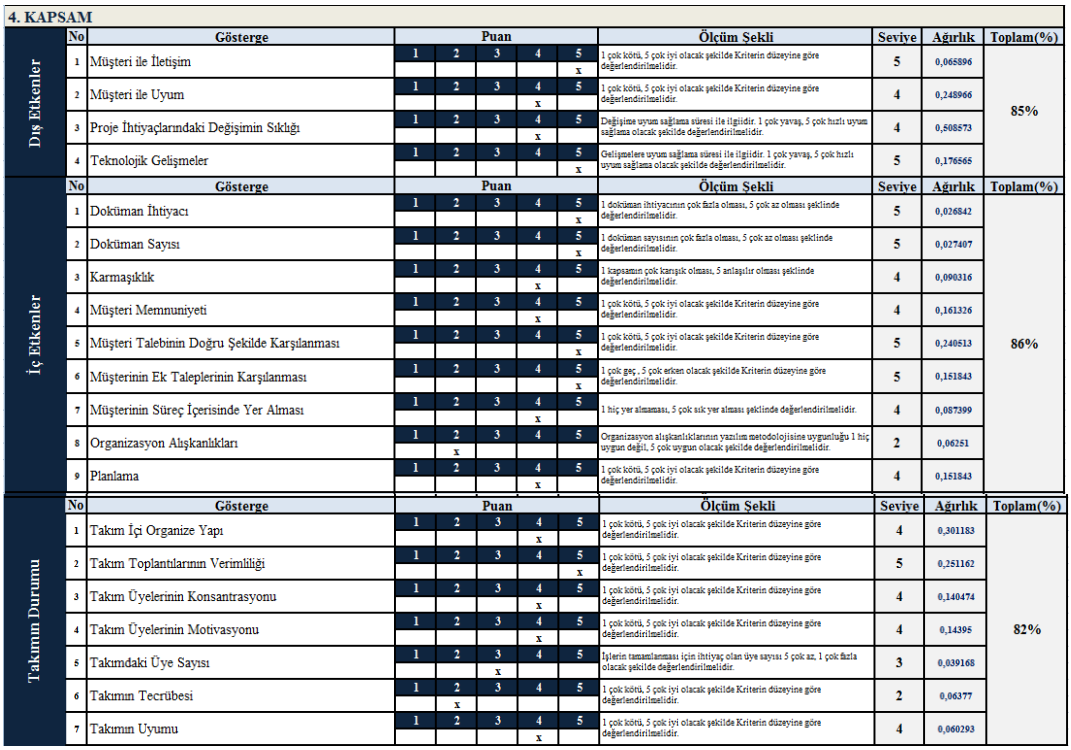
2. KALİTE											
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Yazılım Kalitesi	1	Esneklik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,056037	83%
							x				
	2	Genişletilebilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,037363	
						x					
	3	Güvenilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,103793	
						x					
	4	Kod Kalitesi	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,215941	
							x				
	5	Kullanılabilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,132776	
					x						
	6	Test Kolaylığı	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,230363	
						x					
	7	Çalışabilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,223727	
						x					

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Analiz Tasarım Kalitesi	1	Bütünlük	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,092257	96%
					x						
	2	Doküman Kalitesi	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,295404	
							x				
	3	Doğruluk	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,130901	
							x				
	4	Kullanım Kolaylığı	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,481438	
							x				

3. ZAMAN											
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Teslim Süreleri	1	Analiz Süresi	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,120782	90%
							x				
	2	Destek Süresi	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,292905	
						x					
	3	Tasarım Süresi	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,058002	
					x						
	4	Test Süresi	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,302366	
							x				
	5	Yaygınlaştırma Süresi	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,038224	
					x						
	6	Yazılım Süresi	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,187721	
							x				

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Zaman Planlaması	1	Fazla Mesai Süreleri	1	2	3	4	5	1 yüksek sayıda fazla mesai, 5 fazla mesai olmaması skalasına göre değerlendirilmelidir.	5	0,249981	85%
							x				
	2	Öncelikli İşlerin Teslim Süreleri	1	2	3	4	5	1 çok geç teslim , 5 çok erken teslim olacak şekilde değerlendirilmelidir.	4	0,750019	
						x					

Şekil D.1 : Scrum uygulama değerlendirme sonuçları 1.



4. KAPSAM												
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Dış Etkenler	1	Müşteri ile İletişim	1	2	3	4	5 x	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,065896	85%
	2	Müşteri ile Uyum	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,248966	
	3	Proje İhtiyaçlarındaki Değişimin Sıklığı	1	2	3	4 x	5	Değişime uyum sağlama süresi ile ilgilidir. 1 çok yavaş, 5 çok hızlı uyum sağlama olacak şekilde değerlendirilmelidir.	4	0,508573	
	4	Teknolojik Gelişmeler	1	2	3	4	5 x	Gelişmelere uyum sağlama süresi ile ilgilidir. 1 çok yavaş, 5 çok hızlı uyum sağlama olacak şekilde değerlendirilmelidir.	5	0,176565	
İç Etkenler	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
	1	Doküman İhtiyacı	1	2	3	4	5 x	1 doküman ihtiyacının çok fazla olması, 5 çok az olması şeklinde değerlendirilmelidir.	5	0,026842	86%
	2	Doküman Sayısı	1	2	3	4	5 x	1 doküman sayısının çok fazla olması, 5 çok az olması şeklinde değerlendirilmelidir.	5	0,027407	
	3	Karmaşıklık	1	2	3	4 x	5	1 kapsamın çok karışık olması, 5 anlaşılır olması şeklinde değerlendirilmelidir.	4	0,090316	
	4	Müşteri Memnuniyeti	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,161326	
	5	Müşteri Talebinin Doğru Şekilde Karşılanması	1	2	3	4	5 x	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,240513	
	6	Müşterinin Ek Taleplerinin Karşılanması	1	2	3	4	5 x	1 çok geç , 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,151843	
	7	Müşterinin Süreç İçerisinde Yer Alması	1	2	3	4 x	5	1 hiç yer almaması, 5 çok sık yer alması şeklinde değerlendirilmelidir.	4	0,087399	
	8	Organizasyon Alışkanlıkları	1	2 x	3	4	5	Organizasyon alışkanlıklarının yazılım metodolojisine uygunluğu 1 hiç uygun değil, 5 çok uygun olacak şekilde değerlendirilmelidir.	2	0,06251	
	9	Planlama	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,151843	
Takımın Durumu	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
	1	Takım İçi Organize Yapı	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,301183	82%
	2	Takım Toplantılarının Verimliliği	1	2	3	4	5 x	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,251162	
	3	Takım Üyelerinin Konsantrasyonu	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,140474	
	4	Takım Üyelerinin Motivasyonu	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,14395	
	5	Takımdaki Üye Sayısı	1	2	3 x	4	5	İşlerin tamamlanması için ihtiyaç olan üye sayısı 5 çok az, 1 çok fazla olacak şekilde değerlendirilmelidir.	3	0,039168	
	6	Takımın Tecrübesi	1	2 x	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	2	0,06377	
	7	Takımın Uyumu	1	2	3	4 x	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,060293	

Şekil D.2 : Scrum uygulama değerlendirme sonuçları 2.

EK E

1. MALİYET

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Süreç Maliyetleri	1	Analiz Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	3	0,135988	51%
					x						
	2	Destek Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	2	0,061143	
				x							
	3	Tasarım Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	3	0,103982	
					x						
	4	Test Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	2	0,441308	
				x							
	5	Yaygınlaştırma Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	4	0,042727	
						x					
	6	Yazılım Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	3	0,214852	
					x						
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Dış Kaynak Maliyetleri	1	Dış Kaynak Analiz Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	2	0,136498	62%
				x							
	2	Dış Kaynak Destek Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	4	0,238476	
						x					
	3	Dış Kaynak Yazılım Maliyeti	1	2	3	4	5	1) Maliyet çok yüksek 2) Maliyet yüksek 3) Maliyet normal. 4) Maliyet düşük 5) Maliyet çok düşük	3	0,625026	
					x						

2. KALİTE

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Yazılım Kalitesi	1	Esneklik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	2	0,056037	73%
				x							
	2	Genişletilebilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,037363	
					x						
	3	Güvenilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,103793	
						x					
	4	Kod Kalitesi	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,215941	
					x						
	5	Kullanılabilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,132776	
						x					
	6	Test Kolaylığı	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,230363	
						x					
	7	Çalışabilirlik	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,223727	
						x					
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Analiz Tasarım Kalitesi	1	Bütünlük	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,092257	63%
					x						
	2	Doküman Kalitesi	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,295404	
					x						
	3	Doğruluk	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,130901	
						x					
	4	Kullanım Kolaylığı	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,481438	
					x						

3. ZAMAN

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Teslim Süreleri	1	Analiz Süresi	1	2	3	4	5	1 çok erken, 5 çok geç olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,120782	50%
					x						
	2	Destek Süresi	1	2	3	4	5	1 çok erken, 5 çok geç olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,292905	
					x						
	3	Tasarım Süresi	1	2	3	4	5	1 çok erken, 5 çok geç olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,058002	
					x						
	4	Test Süresi	1	2	3	4	5	1 çok erken, 5 çok geç olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	2	0,302366	
				x							
	5	Yaygınlaştırma Süresi	1	2	3	4	5	1 çok erken, 5 çok geç olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,038224	
					x						
	6	Yazılım Süresi	1	2	3	4	5	1 çok erken, 5 çok geç olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	2	0,187721	
				x							
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Zaman Planlaması	1	Fazla Mesai Süreleri	1	2	3	4	5	1 yüksek sayıda fazla mesai, 5 fazla mesai olmaması skalasına göre değerlendirilmelidir.	2	0,249981	55%
				x							
	2	Öncelikli İşlerin Teslim Süreleri	1	2	3	4	5	1 çok geç teslim , 5 çok erken teslim olacak şekilde değerlendirilmelidir.	3	0,750019	
					x						

Şekil E.1 : Şelale uygulama değerlendirme sonuçları 1.

4. KAPSAM													
	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)		
Dış Etkenler	1	Müşteri ile İletişim	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,065896	58%		
						x							
	2	Müşteri ile Uyum	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,248966			
					x								
	3	Proje İhtiyaçlarındaki Değişimin Sıklığı	1	2	3	4	5	Değişime uyum sağlama süresi ile ilgilidir. 1 çok yavaş, 5 çok hızlı uyum sağlama olacak şekilde değerlendirilmelidir.	3	0,508573			
					x								
	4	Teknolojik Gelişmeler	1	2	3	4	5	Gelişmelere uyum sağlama süresi ile ilgilidir. 1 çok yavaş, 5 çok hızlı uyum sağlama olacak şekilde değerlendirilmelidir.	2	0,176565			
				x									
İç Etkenler	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)		
	1	Doküman İhtiyacı	1	2	3	4	5	1 doküman ihtiyacının çok fazla olması, 5 çok az olması şeklinde değerlendirilmelidir.	1	0,026842	59%		
			x										
	2	Doküman Sayısı	1	2	3	4	5	1 doküman sayısının çok fazla olması, 5 çok az olması şeklinde değerlendirilmelidir.	1	0,027407			
			x										
	3	Karmaşıklık	1	2	3	4	5	1 kapsamın çok karışık olması, 5 anlaşılır olması şeklinde değerlendirilmelidir.	3	0,090316			
					x								
	4	Müşteri Memnuniyeti	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,161326			
					x								
	5	Müşteri Talebinin Doğru Şekilde Karşılanması	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,240513			
						x							
	6	Müşterinin Ek Taleplerinin Karşılanması	1	2	3	4	5	1 çok geç, 5 çok erken olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	2	0,151843			
				x									
	7	Müşterinin Süreç İçerisinde Yer Alması	1	2	3	4	5	1 hiç yer almaması, 5 çok sık yer alması şeklinde değerlendirilmelidir.	2	0,087399			
				x									
	8	Organizasyon Alışkanlıkları	1	2	3	4	5	Organizasyon alışkanlıklarının yazılım metodolojisine uygunluğu 1 hiç uygun değil, 5 çok uygun olacak şekilde değerlendirilmelidir.	4	0,06251			
						x							
	9	Planlama	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,151843			
					x								

	No	Gösterge	Puan					Ölçüm Şekli	Seviye	Ağırlık	Toplam(%)
Takımın Durumu	1	Takım İçi Organize Yapı	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,301183	68%
					x						
	2	Takım Toplantılarının Verimliliği	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	4	0,251162	
						x					
	3	Takım Üyelerinin Konsantrasyonu	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,140474	
					x						
	4	Takım Üyelerinin Motivasyonu	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,14395	
					x						
	5	Takımdaki Üye Sayısı	1	2	3	4	5	İşlerin tamamlanması için ihtiyaç olan üye sayısı 5 çok az, 1 çok fazla olacak şekilde değerlendirilmelidir.	3	0,039168	
					x						
	6	Takımın Tecrübesi	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	5	0,06377	
							x				
	7	Takımın Uyumu	1	2	3	4	5	1 çok kötü, 5 çok iyi olacak şekilde Kriterin düzeyine göre değerlendirilmelidir.	3	0,060293	
					x						

Şekil E.2 : Şelale uygulama değerlendirme sonuçları 2.

ÖZGEÇMİŞ

Ad Soyad: Furkan ANARAL

Doğum Yeri ve Tarihi: İSTANBUL / 08.01.1985

Adres: Beylikdüzü, İSTANBUL

E-Posta: fanaral@gmail.com

Lisans: İstanbul Teknik Üniversitesi Endüstri Mühendisliği