T.C.

HALİÇ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÖNETİM BİLİŞİM SİSTEMLERİ

ITIL(INFORMATION TECHNOLOGY INFRASTRUCTURE LIBRARY) UYGULAMA YÖNETİMİ SÜREÇLERİNİN ORTA/BÜYÜK ÖLÇEKLİ ŞİRKETLERDE UYGULAMASI

YÜKSEK LİSANS TEZİ

Hazırlayan

CEM TOPKAYA

Tez Danışmanı

Prof.Dr. ALİ OKATAN

Ocak 2008

İSTANBUL

İÇİNDEKİLER

Sayfa No.

ÖNSÖZ

*Sevgili Annem **Yıldız TOPKAYA**, babam **Fevzi TOPKAYA**, abim **Cenk TOPKAYA** ve kız kardeşim **Canan TOPKAYA**'ya her zaman benimle oldukları ve olacakları için sonsuz şükranlarımı sunar ve desteklerini hiç esirgemeyen değerli abim **Teoman ALPAY**'a ve biricik dostum **B. Naci ALPAY**'a en içten teşekkürlerimi sunarım.*

Cem Topkaya
Ocak, 2008

ŞEKİL LİSTESİ

KISALTMALAR

ITIL	IT Infrastructure Library-Bilişim Teknolojisi Altyapı Kütüphanesi
BS15000	ISO20000 eskiden BS15000/ BS 15000 olarak bilinirdi.
ISO9000	International Standards Organization 9000-Uluslar arası Standartlar Örgütü 9000
PDA	Personal Digital Asisstant - Kişisel Sayısal Yardımcı
SSL	Secure Socket Layer - Güvenli Soket Tabakası
ERP	Enterprise Resource Planning
XML	Extensible Markup Language - Genişletilebilir İşaretleme Dili
RAD	Rapid Application Development - Hızlı Uygulama Gelişimi
Y2K	Year 2000 Problem - 2000 Yılı Problemi
UML	Unified Modelling Language
CI	Configuration Items - Konfigürasyon Bileşeni
API	Application Program Interface - Yazılım Programlama Arayüzü
WMI	Microsoft Windows Management Instrumentation - Microsoft Windows Yönetim Enstrümantasyonu
CIM	Common Information Model - Yaygın Bilgi Modeli
WQL	WMI Query Language - WMI Sorgu Dili
SQL	Structured Query Language - Yapısal Sorgulama Dili
CPU	Central Processing Unit - Merkezi İşlem Birimi
BİT	Bilgi ve İletişim Teknolojisi (Information and Communications Technology)

TABLO LİSTESİ

ÖZET

Uygulama Yönetim süreçleri, iş süreçlerinin nasıl geliştirildiği, yönetildiği, iyileştirildiği ve ne zaman gerekli oldukları üzerine rehberlik eder. ITIL geleneksel bir yaklaşımla, standart yazılım geliştirme hayat döngüsüne işletme ve iyileştirme katar.

Uygulama Yönetimi, iş birimleri ile fonksiyonel ve iş süreç gereksinimlerini, yeni bir uygulama veya değişim durumunda tanımlamaya çalışır. Bu süreçte, başlangıçtaki hizmet seviyesi ihtiyaçlarını tanımlar.

Uygulama takımı gereksinimleri teknik bir çözüme dönüştürür. Bu noktada Toplam Sahip Olma çalışması, geliştirme ve devam eden destek ücretlerini tanımlamak için yapılır. Bir kere uygulamanın kodlanması ve test süreçleri tamamlandığında, yeni fonksiyonlar üretim ortamına tanıtılır.

Düzenli gereksinimler dâhilindeki değişimler işletim safhasında adreslenir. Fonksiyonel ve performans optimizasyonu iyileştirme aşamasında adreslenir. Performans yönetim takımı, performans iyileştirmesi fırsatları üzerine uygulamayı analiz eder.

Tüm bu süreçler ITIL'ın Uygulama Yönetimi üzerine geliştirdiği en iyi uygulamaların bir ürünüdür.

SUMMARY

Application Management processes guide how business applications are developed, managed, improved, and when necessary, sunset. ITIL takes a traditional approach and adds Operate and Optimize to the standard Software Development Lifecycle.

Work with the business units to identify the functional and business process requirements for the change or new application. During this process, initial service level requirements will be identified.

The application team translates the requirements into a technical solution. At this point a Total Cost of Ownership (TCO) study is performed to identify development and ongoing support costs. Once application coding and test proccesses accomplished new functions are being introduced into the production environment.

Changes in regulatory requirements would be addressed in Operate phase.

Both functional and performance optimization are addressed in this step. Performance Management teams analyze applications to identify any performance improvement opportunities.

All these processes mentioned above are the best practices produced by ITIL on the Application Management.

1 GİRİŞ

ITIL'ın geliştirilmesinin arkasında yatan kültür, organizasyonların ticari başarıyı yakalama ve iş ihtiyaçlarını Bilgi Teknolojilerini kullanarak tatmin etme sürecidir. Bu bağımlılık, Bilgi Teknolojilerinin idare edildiği, Bilgi İşlem bölümlerinde hizmet kalitesinin de doğru orantılı olarak geliştirilmesini gerektirmiştir. İş tarafının ihtiyaçları ve kullanıcı gereksinimlerinin bir araya gelmesi ile Bilgi Teknolojilerinden elde edilecek kaliteli hizmete olan talep, gün geçtikçe daha da artmıştır.

Hangi tipte ya da ölçekte bir organizasyon, ulusal, hükümet, çok uluslu gruplar..vb olursa olsun bu durum, hepsi için yadsınamaz bir gerçektir. Yerel, merkezi, bölgesel, dışarıdan idare edilen, içeride ama dış kaynaklar ile idare edilen ya da başka türlü de olsa organizasyonlar içerisinde Bilgi İşlem, destek ve hizmet vermeye devam edecektir. Her durumda ihtiyaçları karşılayan, ekonomik hizmet sunan, güvenilir ve tutarlı olacak amaçlara cevap verecektir.

Bilgi İşlem, organizasyonun iş tarafının ihtiyaçlarına göre bilgi işlem hizmetlerinin dağıtımı ve desteklenmesi ile ilgilidir. ITIL, Bilgi İşlemin hizmetlerinin dağıtımı ve desteği için en iyi uygulamalar kümesini içerir, böylece iş tarafının hedeflenen verimliliği bilgi teknolojileri ile desteklenerek arttırılır.

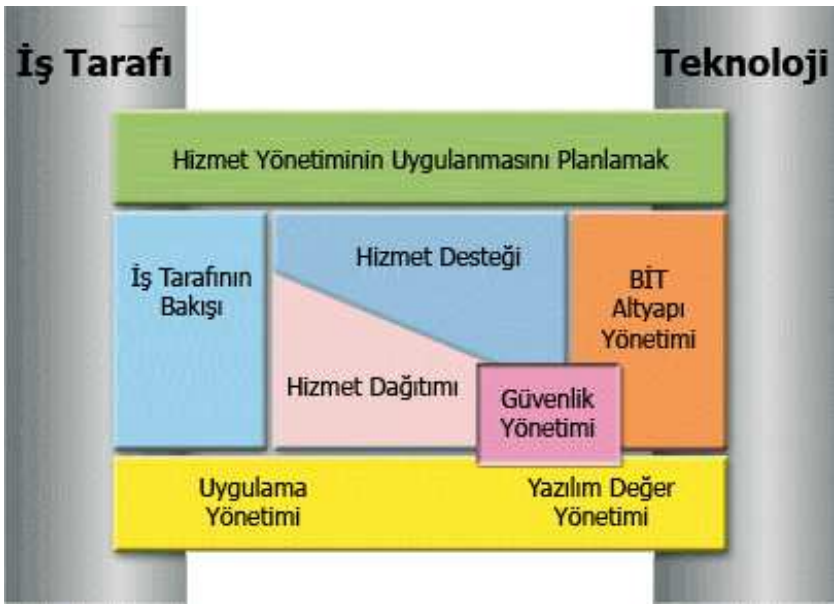
İngiltere Standartlar Enstitüsünün standartları tarafından Bilgi Teknolojileri Hizmet Yönetimi(BS15000) ve ISO kalite standartları ile desteklenir.

Bu tez çalışması, ITIL'ın Uygulama Yönetimi konusunda Türkçe kaynakların neredeyse olmadığı, yurt dışı kaynaklı birçok şirketin ITIL sertifikalı çalışanları istihdam etmesi ya da dış kaynaklı çalışmalarda firmalarında sertifikalı çalışanların bulunmasını öncelikli şart koştukları düşünülerek, Türkçe kaynak oluşturulması amacıyla hazırlanmıştır.

2 ITIL Tarihçe

1980'lerin sonlarında başlatılarak yavaş yavaş geliştirilen ITIL, zaman içinde güncellenmiş ve günümüz rekabet dünyasında tavsiye edilen dağıtık işletmelerin kullanımına sunulmuş, bilgi işlem ve iş operasyonların hedefleri için tavsiyeler bütünüdür.

İş fonksiyonlarının, bilgi işlem hizmetleri konusundaki ilerlemeleri ve yönetimi yeni bir konu değildir. Tüm hizmet yönetimlerinin en iyi pratikler ile birlikte bir çatı altında toplanması hem radikal hem de yeni bir durumdur.



Şekil 0.1 ITIL V2 çatısı

2.1 İş Tarafının Bakışı

İşletmelerde paranın çevriminin yapıldığı birim olan iş tarafı, bilgi işlem ve haberleşme teknolojilerinden operasyonlarının daha hızlı ve hayat çevrimlerine daha iyi destek vermesini ister. İş tarafının bakışı, bilgi işlem ve haberleşme teknolojilerinin en iyi uygulama ile faydalarını anlamasını sağlarken aynı zamanda hizmet sağlayıcısının hizmetinin iş tarafı ile aynı seviyede olmasını sağlar. Bu sayede iş tarafının fayda sağlamak istediği seviye ile hizmet sağlayıcının hizmet seviyesi denkleştirilerek fayda en üst seviyeye getirilir.

2.1.1 Bilgi ve Haberleşme Alt Yapısının Yönetimi

Bilgi ve haberleşme alt yapısının yönetimi, tüm yönleri ile iş tarafının tanımlanması ve ihtiyaçlarının karşılanması esasına göre yapılandırılır.

Yapılacak destek, 4 madde halinde gösterilebilir(OGC,2007,s:22)

- Tasarım ve planlama
- Plana göre yerleşim ve yayılma
- Tüm işlemlerin durumu ve ölçülmesi
- Teknik destek işlemleri

2.1.2 Hizmet Desteği

Hizmet masası olayların günlüğünün tutulduğu ve olay yönetimi sürecinin desteklendiği bir yer olmalıdır. Fakat olaylarla ilgilenmekten çok daha geniş bir rolü olmalıdır. Son kullanıcı açısından birincil arabirim olarak hizmet masasının işlevinin, değişiklik taleplerinin koordinasyonunu sağlamak, problemler ile aktif bir şekilde ilgilenmek, hizmet amaçlarını ve başarılarını anlatmak ve müşteri memnuniyetini izlemek gibi kapsamlı hizmetler grubunu ele almaktadır.

Hizmet masasının işlevinin diğer ITIL süreçlerine bir arabirim tedarik etmesi ve iş akışını otomatik kontroller aracılığıyla kesintisiz sürdürebilmesi çok önemlidir. Birincil rolü ise, Olay Yönetimi süreçleri ile ilintili faaliyetleri ele almaktır.

Organizasyon sadece yardım sağlama düşüncesine takılıp kaldığında ve son kullanıcılara verilen hizmetleri gerçekten iyileştirmediğinde hata ortaya çıkabilir. Hizmet Masası işlevi, işler yanlış gittiğinde tepkisel yardım hizmetleri sunmak için vardır. Ama aktif hizmet geliştirmelerin de kritik rol oynar(Glick, 1991, s:42).

Örneğin; Hizmet Masasına gelen en yaygın talep olan parolanın sıfırlanmasına bakalım. Araştırmalar, Hizmet Masası'na gelen çağrıların en az %25'inin uygulama parolasının sıfırlanması için olduğunu göstermektedir. Standart ya da geleneksel "yardım" masası modelinde, son kullanıcı doğrudan yardım masasını arayacak ve destek analisti kullanıcının parolasını sıfırlamak için bir dizi faaliyet başlatacaktır. Bu, hem verim kaybıdır hem de maliyet açısından aşırı derecede etkisizdir. Gerçek bir "hizmet" masası modelinde, son kullanıcı parola sıfırlama talebini otomatikleştirmek için kendi kendine hizmet işlevine ve bekli de hizmet masasına gelen "nasıl yapmalı" çağrılarını ortadan kaldırmak için kullanıcı eğitimine erişecektir. Bundan ötürü Telekom maliyetlerinde azalma ve son kullanıcı ile etkileşim ortadan kaldırılacaktır.

2.1.2.1 Hizmet Masasının Amaçları

Hizmet masasının birincil amacı; Bilgi İşlem Hizmet Yönetimi arasında merkezi irtibat noktası olarak hareket etmektir ve bu yüzden her destek ihtiyacını ele alacak şekilde donanımlı olması şarttır. Diğer işlevi son kullanıcıların işlerini normal hizmet seviyelerine göre yürütebilmelerini sağlayacak Bilgi İşlem faaliyetlerini aktif bir şekilde anlatmaya hazır ve donanımlı olmaktır. Yani Hizmet Masası nedenlerin ve cevapların aranacağı ilk yer olmalıdır.

Bir başka amacı, bütün olayları ve hizmet taleplerini ele almaktır. Bu işlev olayların kök nedenini bulmak ile ilgili değildir.

Nihai amacı ise Değişiklik ve Sürüm Yönetimi, Konfigürasyon Yönetimi, Hizmet Seviyesi Yönetimi gibi süreç faaliyetleri için bir arabirim sağlamaktır.

2.1.3 Hizmet Sunumu

Hizmet sunumu, Bilgi İşlem hizmetlerinin kendi kendilerine yönetimi ve hizmet sağlayıcı ile müşteri arasında sağlanan Bilgi İşlem hizmetlerinin yönetim uygulamalarını içerir.

Hizmet sunumu 5 disiplinden oluşur.

Bunlar:

- Hizmet Seviyesi Yönetimi
- Kapasite Yönetimi
- Beklenmedik Durum Yönetimi
- Erişebilirlik Yönetimi
- Bilgi Teknolojilerinin finansal yönetimi

2.1.4 Hizmet Yönetiminin Planlanması ve Uyarlanması

Hizmet Yönetimi, müşterinin ihtiyaç ve taleplerinin Bilgi İşlem tarafında karşılanması halidir. Bu başlık altında bilgi işlem hizmetlerinin iyileştirilmesini, iş tarafının ihtiyaçlarının karşılanması konusunda hizmet planlamasının aynı düzleme taşınması işlenmektedir. Hali hazırda yürüyen hizmetlerin değerlendirmesini yapıp, olgunlaşmalarını sağlayabilmek için rehberlik yapılır ve en iyi uygulama sunulur(Vail, 1991, s:278).

2.1.5 Uygulama Yönetimi

ITIL çatısının son parçası Uygulama Yönetimi ile ilgilenir. Uygulama Yönetimi; organizasyonlarda kullanılan ya da kullanılacak yazılımların başlangıcından kaldırılmasına kadar geçecek sürede, iş tarafının karmaşık ihtiyaçlarının karşılanmasına çalışır(Glick, 1991, s:69). Uygulama Yönetimi bu sürede sürekli olarak Bilgi İşlem Hizmet Yönetiminin diğer disiplinleri ile doğrudan temas içinde olur. İş tarafının sıkı bir şekilde uygulamaların hayat çevrimleri etrafında bulunması, bu sürede Bilgi İşlem projelerinin ve stratejilerinin yönetilmesinde büyük önem arz eder. İş tarafı yatırımların düzgün yapılabilmesi açısından Bilgi İşlem ile aralarındaki bu ilişki kritik bir öneme sahiptir.

Bir uygulama(yazılım), iş tarafının işlemleri, prosedürler ve fonksiyonlarının işletebilmesine doğrudan destek verecek şekilde tanımlanabilir. Uygulamalar, veri ve donanım, işletim sistemleri gibi altyapı bileşenleri ile Bilgi İşlem sistemlerini birer hizmete çevirirler. Bunu basitçe taşınabilir pos cihazları ile yapılan satışlarda kullanılan programlar ile örnekleyebiliriz. İş tarafı ürünün satışı ve stok yönetimi ile ilgilenirken, bir donanım olan pos cihazının, sim kart üzerinden ürün, müşteri, adet ve fiyat gibi bilgileri transfer etmesini sağlayan uygulama ile Bilgi İşlem tarafı ilgilenmektedir. Bu örnekte Bilgi İşlem pos cihazı ve sim kartı birer donanım olarak kullanırken, satışa konu olan ürünü, bu donanım ve haberleşme yöntemi ile iş tarafının emrine sunulan hizmete çeviren uygulamadır.

İş tarafı bugünün bilgi dünyasında birçok meydan okuma ile yüz yüze gelmektedir. İş tarafının birçok değişime, isteğe, dinamik olarak değişen pazar şartlarına daha hızlı cevap verebilmesi için Bilgi İşlem organizasyonunda ve bu organizasyonun desteğinde yapılacak değişimler hususunda çevik olması gerekmektedir. Bu değişimler elbetteki birtakım sorunları da yanı sıra getirecektir. Bilgi İşlem, yönetimindeki varlıklar ile iş tarafını ciddi anlamda kapsamaktadır. Varlık, tanım olarak bir insan, yazıcı, bilgisayar, işletim sistemi, faks makinesi, PDA(Personal Digital Asisstant, Kişisel Sayısal Yardımcı), cep telefonu, pos cihazı olabileceği gibi basılmış bir dokuman ya da ağ yapısı da olabilmektedir(OGC,2007,s:89).

Organizasyonun bir parçası ya da tümü e-iş üzerinden işliyor olabilir. Bu durumda müşteri tamamen ya da kısmen Bilgi İşlem alt yapınız ve hizmetleriniz ile ilgileniyor olacaktır. Bu durumda organizasyonun iş yapma stratejisi, sürekliliği, iş akış şeması müşteriyi doğrudan ilgilendiriyor olacaktır ve gereksiz sistem

fonksiyonları ile hedeflerin kötü, yavaş bir sistemle yakalanması mümkün olmayacaktır. Bunu basit bir e-iş örneği ile anlatmak mümkündür.

Örneğin; binlerce ürünü bulunan bir organizasyonun internet üzerinden satış gerçekleştirmek için web formları ile bir uygulama geliştirmiş ve bu uygulama üzerinden tüm ürün bilgisi, fiyat ve stok bilgisini takip ettiğini kabul edelim. Bu uygulama hem organizasyon içerisinden erişime açık hem de müşteriye ulaşabilmek için internet üzerinden açık olsun. Bu erişimlerin hızı içeride farklı, dışarıda daha farklı olacaktır. İçeride yeni ürün girişleri için kullanılırken yaşanılan hız ve alt yapı, müşterinin yaşayacağı hız ve alt yapıdan farklı olacaktır. Müşterinin yapacağı aramalar ve satın alma işlemleri karşısında uygulama alt yapısı yeterince hızlı olmazsa, bu durum müşterinin uygulamayı terk etmesine neden olacaktır. Donanım olarak sisteme verilecek desteği somutlaştıran bu örneği uygulamanın iş akış yönetimde de örnekleyerek genişletmek mümkündür. Yine aynı uygulamada üye kaydı yapılmadan ürünlerin görüntülenmemesi ya da sepetin fiyatlandırılma esnasında müşteri bilgilerinin SSL(Secure Socket Layer) gibi yöntemlerin kullanılmadan yani şifrelenmeden internette dolaştırılması doğrudan müşteri kaybına verilebilecek örneklerdir. Bu yüzden organizasyonun uygulama çevrimini çok ciddi olarak iş tarafının Bilgi İşlem ile birlikte işletilebilir, güvenilebilir, başarımı ve yönetilebilirliği yüksek tasarım için çalışması gerekmektedir. Birliktelik sadece tasarım değil, değişim yönetiminde de önem arz etmektedir.

Gartner Group tarafından yayınlanan bir raporda, uygulamaların başarısızlıkları %40 oranında işletme hatalarından, %40 oranında uygulama hatalarından ve kalan %20'nin ise teknolojik hatalardan meydana geldiğini vurgular. İşletme hataları; unutulan talepler, organizasyon prosedürlerinin olmayışı, yedekleme ve hata bildirimlerinden kaynaklanmaktadır. Uygulama hataları,;doğru ya da hiç yapılmayan testlerden, gerekli değişimlerin düzgün yapılamayışından ve sistem yüklenmelerinden kaynaklanmaktadır. Teknolojik hatalar ise; doğru seçilmeyen donanımlar, ağ hataları ya da uygulama için seçilen teknolojinin yetersiz olmasından kaynaklanmaktadır.

Hedef, bir uygulamanın geliştirilmesi ve dağıtımında Bilgi İşlem ve iş tarafının birlikte stratejilerini karşılıklı çaba harcayarak geliştirmeleridir.

3 UYGULAMA YÖNETİMİNİN KONUMLANMASI

Uygulama Yönetimini tam olarak anlayabilmek için Uygulama Yönetimini, Hizmet Yönetimi ve uygulama geliştirme ile karşılaştırmak gerekir.

3.1 Uygulama Yönetimi

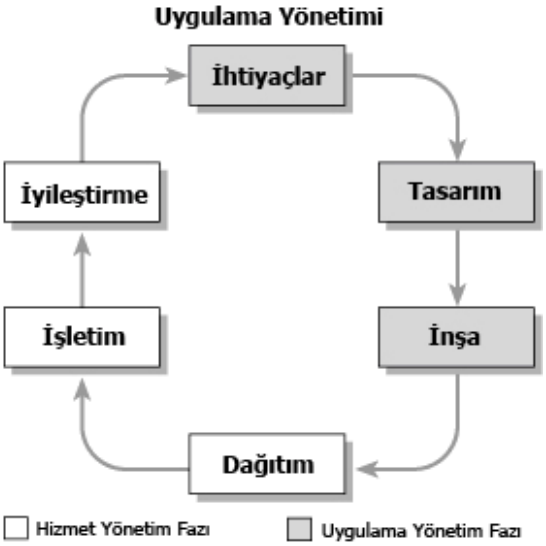
Uygulama Yönetimi, hayat çevrimi boyunca uygulamanın yönetimini ve yeni ihtiyaçlara göre değişimini bakımını tanımlayan, peş peşe devam eden süreç ve aktivitelerin birleşimidir. Bu hayat çevrimi hem Uygulama Geliştirmeyi hem de Hizmet Yönetimi aktivitelerini kapsamaktadır. Bu iki disiplini bir araya getirmek Bilgi İşlem Hizmet Yönetiminin hedeflerini yakalamayı, uygulamanın dağıtımını ve daha fazla işletilebilirliği ile yönetilebilirliğini sağlayacaktır.

3.2 Uygulama Geliştirme

Uygulama Geliştirme, nihayetinde organizasyonun bir veya daha fazla parçası tarafından kullanılacak planlanması, tasarlanması, temin edilmesi ve inşa edilmesi gereken bir uygulamanın aktiviteleri ile ilgilidir. Bu uygulama genel bir geliştirme, satın alma, barındırma ya da bunların herhangi bir birleşimi ile elde edilmiş olabilir. Uygulama Geliştirme uygulamanın dağıtımı, işletimi, rutin yönetimi ya da bakımını kapsamaz.

3.3 Hizmet Yönetimi

Hizmet Yönetimi, uygulamaların işletimi, iyileştirilmesi, desteği ve dağıtımı ile ilgili aktivitelerini kapsar. Ana hedefi, geliştirilmiş ve organizasyona uyarlanmış uygulamanın, geliştirilmesi ve dağıtımının sağlanmasındaki sürekliliktir.



Şekil 0.1 Uygulama geliştirme fazları.

Uygulama Yönetimi, bir yazılımın her aşaması için izlenmesi gereken tüm yönetilebilir işlemleri kapsamlı bir şekilde tanımlar. Uygulama Yönetimi ayrıca tüm Bilgi İşlem profesyonellerinin başarılı bir dağıtım ve var olan uygulamanın ayakta tutulabilmesi için izleyebilecek bir yol haritası da sağlar.

Uygulama Geliştirme, Hizmet Yönetimi ve Uygulama Yönetimi farklı disiplinler gibi davranmalarına rağmen gerçekte bu disiplinler bir birine geçmiş haldedirler. İşte bu disiplinlere bakış elbette ki kurumların kendi işleyişlerine göre farklılıklar gösterecektir. Nihayetinde ise yine birbirine bütünleşmiş olacak şekilde birleşirler.

4 İŞ TARAFININ DEĞERLERİNİ YÖNETMEK

4.1 İş tarafı İle Bilgi İşlemin Hizalanması

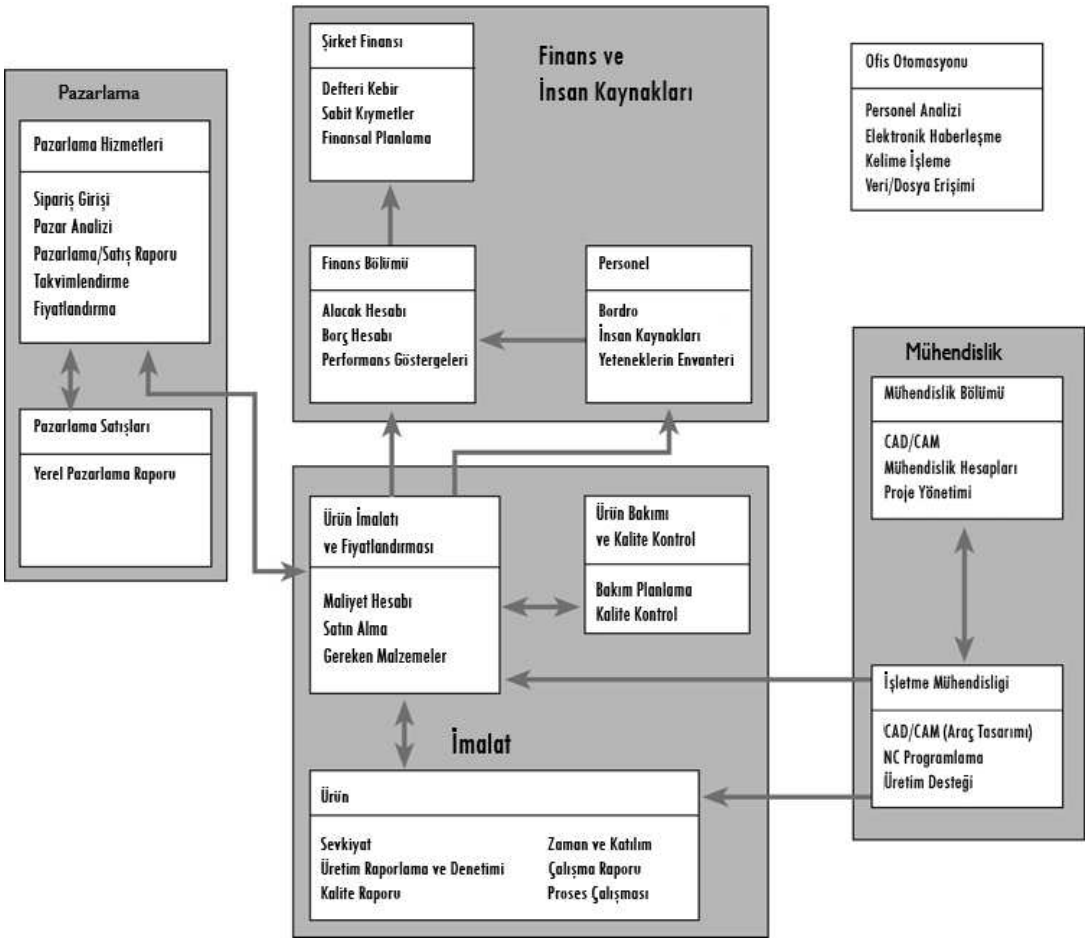
4.1.1 İş Tarafı İle Bilgi İşlemin İhtiyaçlarını Anlayarak Hizalama

“Teknoloji tek başına iş tarafına destek sağlamak için değil, iş tarafının ihtiyaçlarını karşılayabilmek içindir.”

Yukarıdaki cümle Bilgi İşlem yöneticilerinin odaklanması gereken noktanın, iş tarafının ihtiyaçlarını karşılarken en az teknoloji ile yapılması gerekliliğini anlatır. Bunu yaparken iş tarafının gerçek ihtiyaçlarını tam anlamıyla karşılamak gerektiği üzerinde durulmalıdır. Bu sayede yapılan teknoloji yatırımlarının daha yüksek bir geri dönüşümü olduğu görülecektir. Çünkü yapılan yatırımlar kritik düzeydeki ihtiyaçları fonksiyonel olarak cevaplayacaktır. İş ile Bilgi İşlemin aynı hizaya getirilmesi bu yüzden çok daha önemlidir.

Özellikle her şirket iş amaçlarını başarmak için uygulamalar barındırırlar. Bu hedefler sıklıkla müşteriye daha iyi hizmet veya iş akışını daha verimli olarak yapabilmek üzerinedir. İnternet, intranet, geniş bant, kablosuz ağlar, siteler ve mesajlaşma gibi teknolojiler yine işten-işe, işten-tüketiciye internet siteleri müşteriye olan hizmetin daha verimli ve daha iyi olması amacıyla kurulurlar. Örnek olarak kurumların bu tarz isteklerini karşılamak üzere hazırlanan programlardan ERP(Enterprise Resource Planning) paketleri ve finansal sistemler (muhasebe, faturalama sistemleri gibi sistemler) verilebilir. İş tarafının ihtiyaçlarını, Bilgi İşlemin verebilecekleri ile aynı hizaya getirmek, yukarıda verilen genel örneklerde olduğu gibi iş süreçlerinin (tipik iş süreçleri; sipariş almak, satılan ürünler, dağıtım hizmetleri, ürün dağıtımı, fatura hizmetleri, tahsilât gibi işin parçası her aktivitedir) daha iyi ve verimli kılınabilmesi demektir.

Bilgi İşlemin, iş tarafını anlaması ve destek olabilmesi için iş tarafının iyi analiz edilerek küçük fonksiyonlara bölünmesi ve işleyişin çok açık bir halde anlatılabilmesi gerekmektedir. Bunu gerçekleştirmek için en iyi metodun işin öncelikle iş alanlarına, iş kollarına göre bölmek gerekmektedir.

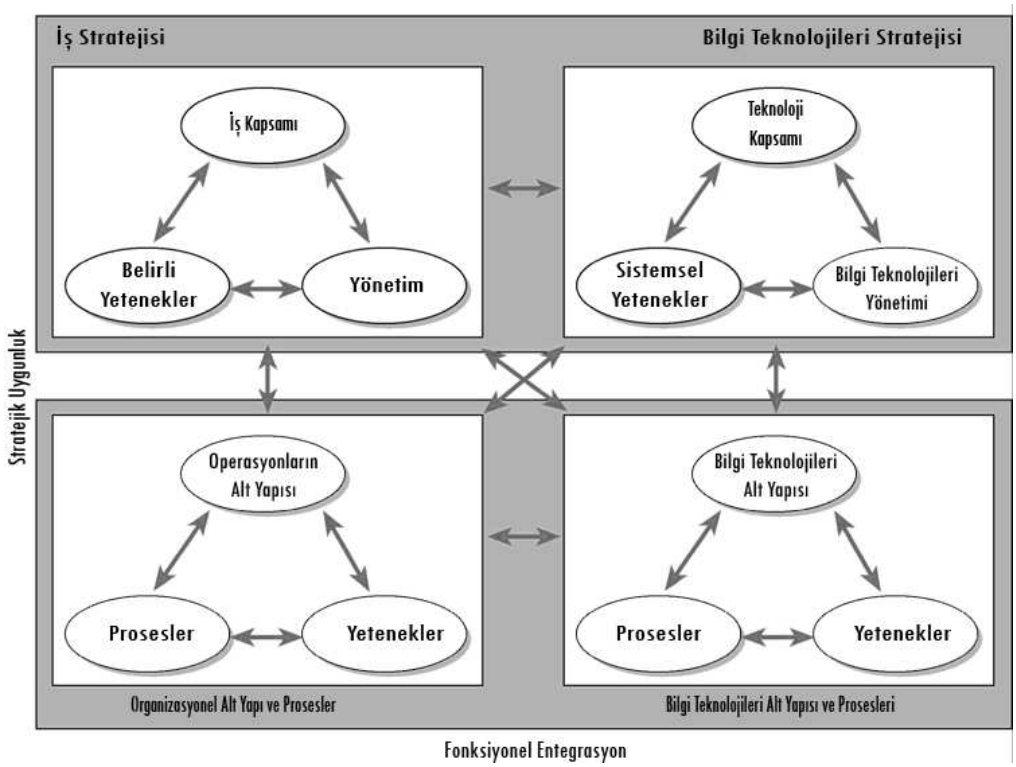


Şekil 0.1 Yüksek seviyeli bir iş mimarisi

Bu resimde esas işi imalat – satış ve pazarlama olan; mühendislik, finans, insan kaynakları ve imalat kollarına sahip bir kurum mevcuttur. Her bir iş alanı dâhilinde, anahtar iş fonksiyonları listelenmiştir. Listelenen bu hizmetler iş tarafının amaçlarını temel alırlar.

4.1.2 Bilgi İşlem Ve İş Tarafının Hizalanması İçin Stratejik Bir Model

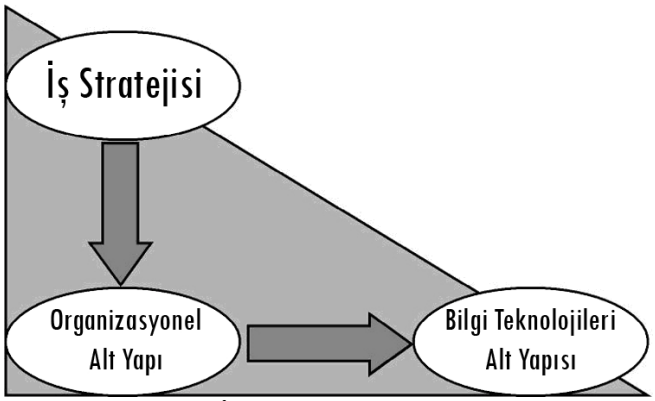
Şekil 4.2 de gösterildiği gibi iş tarafının birçok fonksiyonu şirket tarafından hayata geçirilmiş ve Bilgi İşlem tarafından bu fonksiyonlara destek veren birçok başka Bilgi İşlem fonksiyonları bulunmaktadır. Başarılı bir hizalama yapabilmek için Bilgi İşlem ve iş tarafının farklı açılardan düşünülmesi gerekmektedir. Şekilde bu perspektifler gösterilmiştir.



Şekil 0.2 Stratejik hizalanma modeli

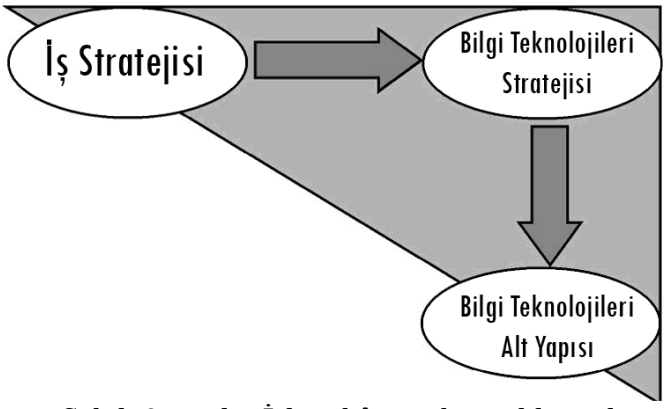
Stratejik hizalanma modeli, stratejik uygunluk ve fonksiyonel bütünleşme diye iki blok üzerine inşa edilmiştir. Stratejik uygunluk, dâhili ve harici alanlardaki tüm ihtiyaçları adreslemeyi tanımlar. Harici alan şirketin rekabet içinde olduğu pazarı gösterirken, dâhili alan ise, iş tarafı veya bilgi işlem operasyonlarını gösterir. İkinci blok ise, fonksiyonel bütünlüğü yani bilgi işlem ile iş tarafının stratejilerinin ihtiyaçları adreslemedeki bütünlüğünü gösterir.

Harici alanlardaki hedef müşteri memnuniyetini sağlamak olduğu için ürün ve/veya hizmetin müşteriye ulaştırılması, müşterinin ihtiyaçlarının doğru analiz edilerek organizasyon içerisinde ticari bir bakışla rakipler ile organizasyonun ayrımını müşteri tarafında sağlayabilecek bir strateji geliştirilmesi ve yönetilmesi gerekmektedir. Bunu da dâhili alanlarda, organizasyonun operasyonlarının alt yapısını, süreçlerini ve kabiliyetlerini değerlendirerek içerideki işleyişi kolaylaştırması ve bunun alt yapısını oluşturması ile sağlayabiliriz. Bilgi İşlem tarafında benzer argümanlar kullanılabilir. Harici Bilgi İşlem alanı ise iş tarafını destekleyecek uygun teknolojilerin adreslendiği yer olarak kullanılabilir.



Şekil 0.3 Bilgi İşlemin maliyet merkezi olduğu durum.

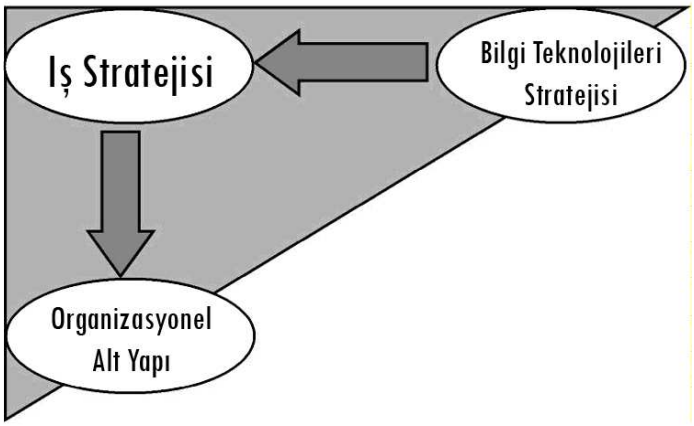
Geleneksel perspektifte bilgi işlem iş tarafının stratejisine uygun olarak hazırlanmış alt yapısal organizasyonlarını destekler durumdadır. Bilgi işlem altyapısının, iş tarafının seçtiği stratejiye uygun olarak iş yönetimine destek vermesi halidir ve bu durumda bilgi işlem bir maliyet merkezi gibi davranır. Bir maliyet merkezinde tüm aktiviteler maliyeti düşürmek yönündedir. Bu durumdaki bilgi işleme sahip olma maliyetini düşürmek için uygulamaların bakımının kolay yapılabilir olarak tasarlanır.



Şekil 0.4 Bilgi İşlem kâr merkezi olduğu durum.

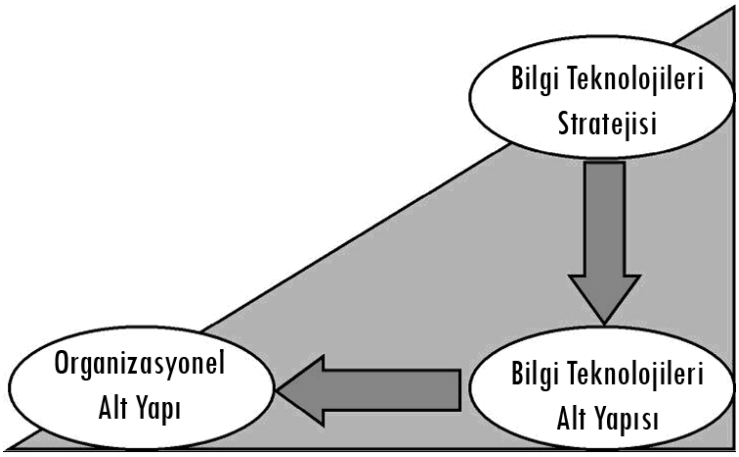
Bilgi işlem ile iş tarafının hizalanması için kullanılabilecek diğer bir yol ise, iş yönetimi ve bilgi işlem yönetiminin rollerini değiştirecek iş stratejisi tanımlamaktır. Bu kez iş tarafının stratejisini destekleyecek bir bilgi işlem alt yapısı oluşturmak yerine, bilgi işlem ile birlikte en uygun teknolojinin kullanılması için bir strateji belirlenir. Bu durumda en uygun maliyet ile en yeterli teknolojiye sahip olma ve bu kez maliyet merkezinden kar elde etme yönünde yeni aktivitelere gidilir.

Bunun sonucunda uygulamalar daha yüksek esnekliye ve iş ihtiyaçlarına göre tekrar değiştirilebilirliğe kavuşulur.



Şekil 0.5 Bilgi işlemin yatırım merkezi olduğu durum.

Bu kez bilgi işlem stratejilerinin başlangıç noktası olduğu durumlar söz konusudur. İş stratejileri, bilgi işlem stratejilerinin üzerine inşa edildiği zaman bilgi işlem bir yatırım merkezi pozisyonuna gelir ve tüm hizmet ve/veya ürün pazara bu kez bilgi işlem stratejileri üzerinden ulaşır.



Şekil 0.6 Bilgi işlemin hizmet merkezi olduğu durum.

Bu son yol ise, bilgi işlem stratejileri ile bilgi işlem hizmetlerinin oluşturduğu organizasyondur. Bu perspektifte bilgi işlem kurumsal kaynakları temsil eder ve tek bir iş stratejisi yoktur.

5 ORGANIZASYONEL KABİLİYETLER VE İŞ ANAHTARLARI İLE SUNUM STRATEJİSİNİN HİZALANMASI

5.1 Değerler İçin Hazır Olma

5.1.1 Risk Yönetimi için Bir Temeli Sağlamak

Birçok yazılım geliştirme projesi daha başlamadan başarısız olur. Bu durumdan uzak durmak için her organizasyonun kendisine bu işi ne kadar başarıp başaramayacağını sorması gereklidir. Demek oluyor ki bir takım sorular sormalıyız kendimize: Şirketimiz bir uzay mekiği için yazılım geliştirme yetisine sahip mi? Sahip olmamız gerekir mi? Verimli bir yazılım geliştirebilir miyiz? İş ve bilgi işlem organizasyonlarımız bunu başarabilir mi? Böyle bir uygulamayı işletebilmemiz için bu tür yeteneklere ihtiyacımız olur mu? Bilgi işlem hizmetlerimiz, anahtar iş tanımlarımızla aynı düzlemde mi? Eğer öyle ise bu düzlemi nasıl genişletmeliyiz?

Birçok organizasyon yeni bir uygulama inşasına karar verirken, bu sorulara aşırı iyimser cevaplar verebilir. Bazı nedenlerden dolayı bu görev bilgi işlem ve iş tarafının olgunlaşma seviyesine göre çok karmaşık gelebilir. Sıklıkla, iş tarafının ihtiyaçlarını yerine getirmek için alternatifler düşünülebilir.

Örneğin;

Birçok ERP paketlerini yürürlüğe koymak ERP paketlerinin organizasyona vaat ettiği başarımı sağlamayabilir. Tam bir iş süreçlerinin ve bütünleşik iş yönetiminin entegrasyonunu kurmak oldukça zor olabilir. Neden? Birçok organizasyon bu ERP paketini parça parça uyarlarken, işletmenin işleyişini ERP paketine göre değiştirmek zorunda kalabilirler. Örneğin; masraf formlarını eskiden yazılı alırken, bu paket ile veri tabanına kaydedilen bilgiler olarak alması gibi.

Bu paketlere uygun bir şekilde iş operasyonu değiştirilse bile yine de bu ERP paketinin başarısız olması bir sebepten dolayı mümkündür. Neden? Birçok nedenden dolayı çalışanlar ya da eski işleyişe uygun giden aktiviteler, bu büyük değişimi absorbe edemeyip direnç göstereceklerdir. Bir ERP paketi tasarımından kodlanmasının sonuna kadar ideal bir işletme göz önüne alınarak tasarlandığı ve organizasyonların bu idealliğe kavuşma arzusunda ya da bu konumda olduklarını

düşünerek üretildiği için, hali hazırdaki organizasyonun işleyişi bu geçişi kolaylaştırmayacaktır.

Hatalı bir proje başlangıcından uzak durmak için riskleri iyi analiz etmek ve bunları yönetebilmek gerekmektedir. İş ihtiyaçlarını ve bilgi işlem organizasyonunu ihtiyaçlara göre hizmet verebilecek duruma getirmek, ideal olarak iş tarafı ile aynı hizada çalışma gayretinde olmak bu tip risklerle karşılaşmamayı sağlayacaktır. Yine de bu ilişkinin olgunlaşma seviyesi ne olursa olsun daima bir gedik bulunacaktır.

Örneğin;

Bir yayın şirketinin pazarlama müdürü en son dijital yayın teknolojini görebileceği bir konferansı ziyaret etmiş ve çeşitli sistemlerden etkilenip şirketin esnekliğini arttıracak bir ürünü; XML tabanlı bir sistemin müthiş avantajlarını müşterilerine ulaştırmak istemiş olabilir. XML üzerinden bu sistemi uyarlayabilmek için ihtiyaç duyacakları şeyleri planlamak ve bu yapı üzerine sistemlerini entegre edebilmek için bilgi işlem bölümünden destek talep etmiş olabilir. XML üzerine pek bir tecrübesi, bilgisi, kullanılabilecek araçlara sahip olmayan bilgi işlem müdürünün saygınlığını korumak için yeni sistemin sözünü vermesi bu sistemin oluşturulması sırasında hem bilgi edinme hem de sistemi yerine getirme çabaları sırasında gecikmelere neden olacaktır.

5.1.2 Bilgi İşlem Kabiliyetlerini Geliştirmek İçin Bir Temel Sağlamak

Yazılımdaki problemleri adreslemede önemli ilk adım, tüm yazılıma kontrol edilebilen, ölçülebilen ve geliştirilebilen bir süreç gibi davranabilmektir. Açık ve tam verimli süreçler, ihtiyaç duyulan görevleri, araçları, yöntemleri kullanan geliştirmeye devam eden, yetenekli ve motivasyonu olan iş gücü ile mümkündür. Bilgi işlem organizasyonlarının yapabilirlik içeriği 6 bakış açısı ile karakterize edilebilir. Bunlar; yöntem, teknoloji, beceri, organizasyon, ölçüm ve kültürdür.

5.1.2.1 Metodoloji

Bir metot bir organizasyonun yazılımın bir parçasını inşa edip dağıtımında kullandığı belirli bir prosedürü tanımlar. Diğerleri, metodu modellenmiş kullanıcı ihtiyaçları, harici ve dâhili modellenmiş sistem tasarımları ve farklı uygun modeller arasındaki şeylerdir.

5.1.2.2 Teknoloji

Teknoloji, yetenekler ve organizasyon yazılımın bir parçasının bir metodunu izlemek ve dağıtmak için ihtiyaç duyduğu malzemedir. Teknoloji araçların kullanımını, ağ protokollerini, donanım platformlarını ve mimari prensiplerini içerir. Yaygın organizasyonlar teknoloji için standartları tanımlamıştır. Uygulama ve alt yapı için standart mimarilere örnek verilebilir.

Organizasyon, bilgi işlem hizmetlerinin yayılması için ihtiyaç duyulan rolleri ve mesuliyetleri tanımlar. Aynı zamanda kaynakların nasıl tahsis edileceğini de tanımlar.

5.1.2.3 Ölçüm

Ölçüm, bilgi işlem hizmetlerinden müşteri memnuniyetini, hizmet kalitesini (müşteri ile servis sağlayıcı arasındaki imzalanmış anlaşma seviyesi), ürünlerin kalitesini, diğer memnuniyetleri ölçümlemeyi sağlayacak niceliklerin yeterlilik sayısını, iyileştirme rehberini ve değişimlerin etkisini ölçebilmeyi kapsar.

5.1.2.4 Kültür

Kültürel bakış açısı, organizasyonun değerlerini, hedeflerini ve iletişim tarzını içerir ve vaat ettiğini gerçekleştirme performansını adresler. Eğer bir organizasyonun kültürü iş tarafının ihtiyaçları ile aynı hizada değişirse, müşteri memnuniyeti ve çalışan morali büyük ihtimal ile düşük olacaktır.

5.2 Sunum Stratejisinin Tanımlanması

5.2.1 Sunum Stratejisi Seçenekleri

Bir organizasyonda, yeni bir uygulamayı geliştirmek ve plana göre yaymak için çok çeşitli yolları vardır. Seçilen yöntem ve strateji, organizasyonun öngördüğü çözümü temel almalıdır. Bu çözüm bilgi işlemin ve iş ortaklarının yetenekleri ile tatmin edici seviyede uygulanmalıdır. Bu düşünce tarzı ile tüm çıkabilecek problemlere işletilebilen yenilikçi çözümlerle müdahale edilmelidir. Tüm bunların sonunda başarı ile bitirilebilen proje ile başarısızlığa uğramış proje arasındaki fark ortaya çıkacaktır. Bu ortaklıkta bir diğer önemli nokta ise, tarafların ya da

organizasyon içindeki grupların avantajlı ve dezavantajlı olduğu durumları ortaya koymasıdır.

Dağıtım stratejisi, iç kaynakların ve dış kaynakların kullanılabilmesi, iç ve dış kaynakların beraber çalışması, ortaklığı ya da birleşmeleri ile de sağlanabilir.

İç kaynakların kullanımı ile yeni, tekrar düzenlenmiş uygulamaların veya veri merkezli operasyonların dağıtımının yapılması sağlanmalıdır.

Dış kaynakların kullanımı ile yazılımın geliştirilmesi, işletilmesi, bakımı ve/veya desteğinin iyi tanımlanmış düzenlemeler ile sağlanmasıdır.

İç ve dış kaynakların birlikte kombinasyonu, genel olarak harici bir firmanın uygulamanın bir kısmını geliştirmesi, bakımı işletilmesi ve/veya desteği için yapılan anlaşmalarla sağlanmasıdır.

Ortaklık, resmi bir düzenlemenin iki veya daha fazla şirketin beraber geliştirme, bakım, işletme veya desteğinin bir uygulama için bilgi işlem veya iş tarafının fonksiyonları ile gerçekleştirilmesi halidir. Bu ortaklığın odak noktası, kritik ve stratejik olarak pazardan veya tecrübelerden istifade edebilmektir.

Birleşmeler, genel olarak bir şirketin diğer bir şirketi nakit para ile satın alması ya da stoklarının değiş tokuşu olarak anlatılabilir. Birleşmeler, pazardaki genişleme veya dolaylı olarak rekabetçi baskıya birer cevap olarak anlamlandırılabilir.

Örneğin;

Orta büyüklükte bir banka ile diğer bir bankanın birleşmesi ürün portföyünü tamamlayıcı nitelikte olabilir. Bu yüzden uygulamaların entegrasyonu basittir. Ancak, bu iki bankanın operasyonlarının pekiştirilmesi niteliğinde birleşme, ekonomik büyümeye yeterli miktarda katkı sağlamayacaktır. Dış kaynakların kullanımı ayrıca bir seçenektir fakat iki bankanın ortak bir firmayı bulması gerekmektedir.

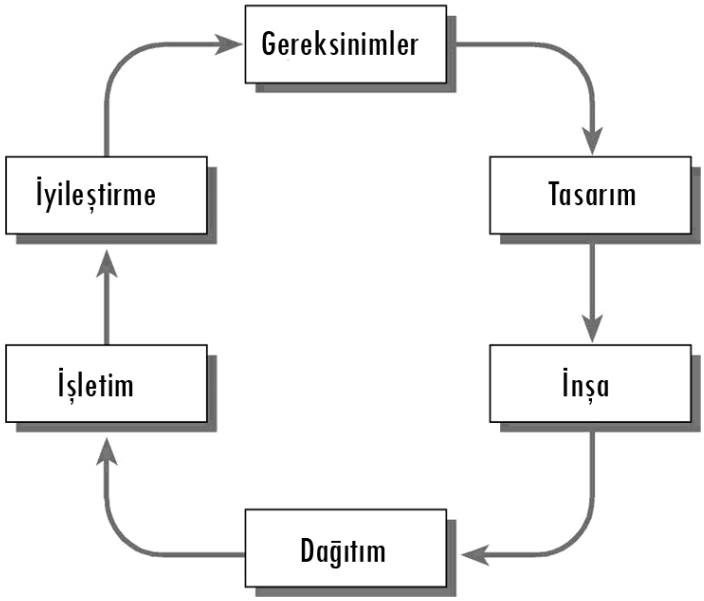
Dağıtım Stratejileri	Avantajları	Dezavantajları
Dahili kaynak kullanımı	Doğrudan kontrol	Boyut sınırları
	Seçimde özgür olma	Maliyet ve hizmet sağlayıcının bulunma süresi

	Hizmetlerde hızlı prototip elde edebilme	Dahili kaynaklara ve onların becerilerine ve yetkilerine bağımlılığı
	Şirkete özgü bilgi	
Dış kaynak kullanımı	Ölçeklenmiş ekonomi	Daha az doğrudan kontrol
	Satın alınmış uzmanlık	Çıkış engeli
	Şirket özüne göre yeteneklere odaklanabilme	Hizmet sağlayıcılara ödemeyebilme riski
	Geçici istekleri destekleyebilme	Bilinmeyen destek sağlayıcı ustalığı ve becerileri
	Yeni hizmetlerin deneme süreli kullanımı	
İç ve dış kaynakların kombinasyonu	Pazara sürme zamanında esneklik	Projenin karmaşıklaşması
	Pazar genişlemesi ya da pazara giriş	Fikir mülkiyeti ve kopya haklarının korunması
	Rekabete dayanan tepki, karşılık	Şirketler arasında kültürel çatışma
	Riskin paylaşımı	Operasyonel entegrasyon
		Operasyonel sistemlerin entegrasyon
		Yan çizmek
Birleşmeler	Rekabete dayanan tepki, karşılık	Finansal kaynak gereksinimleri
	Pazarın genişlemesi ve yeni hizmetler	Şirketler arasında kültürel çatışma
	Doğrudan kontrol	

Tablo 0.1 Dağıtım stratejilerinin avantaj ve dezavantajları

6 UYGULAMA YÖNETİMİ HAYAT ÇEVİRİMİ

6.1 Uygulama Hayat Çevirimi



Şekil 0.1 Uygulama hayat çevirimi

Uygulama hayat çevrimi, bir uygulamaya başlama kararı verilmesinden, uygulamanın hizmetten kaldırılmasına kadar süren zamana denir. Bu zaman içerisinde birçok fazı içerir. Bu zaman zarfında, geleneksel uygulama geliştirme fazı, hizmet yönetim fazı tek bir hayat çevrimi içinde birleştirilir. Tüm bu fazlar, ihtiyaçları, talepleri, tasarım ve inşa sürecini, uygulamanın geliştirme, dağıtım, işletim ve iyileştirme bölümünü oluşturur.

Hayat çevirimi, doğrusal ve şelale tabanlı olabileceği gibi mükemmel bir şekilde modern sistem geliştirme yaklaşımı olan Hızlı Uygulama Geliştirme(RAD, Rapid Application Development) yöntemi ile de yapılmış olabilir.

Bir uygulama elbette ki başladığı gibi işletilmez de. Hem yeni yöntemler ile geliştirebileceği gibi, hem de yeni eklentiler ile hayat çevrimine devam eder. Ayrıca önemli bir kısmı da bir uygulamanın yeterince yaygın kullanılmaması onun kullanımdan kaldırılmasına da engel bir nedendir. Aslında bir uygulamanın hayat çevrimi gerçek bir emeklilik dönemine kadar sürer. Gerçekten emeklilik dönemi; iş ihtiyaçlarını yeterince karşılayıp/karşılamadığı, ihtiyaçların karşılanması için gereken çabanın yazılımın yatırım maliyetini karşılama oranına göre durumu ve yazılımın

hayatını devam ettirmesi için gereken bilgi işlem alt yapısının yine teknolojik alt yapı yatırımını karşılama durumuna göre değerlendirilir.

Bu hayati noktadır. 2000 yılı problemi nedeniyle binlerce yazılım Y2K hazırlığı için hazırlandılar. Bunların büyük bir kısmı da iş fonksiyonlarını yeterince desteklemediler.

6.1.1 Uygulama Geliştirme İşinin Hizmet Yönetimi ile Hizalanması

Bilgi işlem endüstrisi, geleneksel olarak uygulama geliştirme(hizmet yaratma) ve Hizmet Yönetimi(hizmetin ulaştırılması, dağıtılması) işlerini birbirinden ayırmıştır bu her zaman iyi çalışmamış olsa da. Aslında uygulamanın sıklıkla yaratılması ve doğru bakım bağlantısını bu durum kopartmıştır. Uygulama Yönetimi, bilgi işlem tarafında birbiriyle ilgili iki tarafın bir bütün olarak aynı hizada bulunması gerekliliğini işaret eden bir yaklaşım tarzı sergilemiştir.

Birleştirilmiş modelleme dili(UML, Unified Modelling Language) diye adlandırılan tasarım modellerini standart haberleşme dili olarak kabul edilmiş ve bu dil üzerine uygulama üretimini tamamlamış birçok yazılım geliştirilmiştir. Standart modelleme dili, şimdi kendi notasyonlarını üretebileceği araçlarla yaygınlaşmaktadır. Ayrıca entegrasyon araçları gittikçe kolaylaşmakta, farklı araçlarda ama aynı dil üzerinde kullanılabilmektedir.

6.1.2 Uygulama Hayat Çevriminin Organizasyon İçinde Yürütülmesi

Yazılım geliştiricilerin özellikle alınacak kararların hizmetin dağıtımı ve üretilmesi konusundaki etkilerini çok iyi anlaması gerekmektedir. Hizmet Yönetimi ve uygulama gelişimini sağlayan bölümlerin çok yakın işbirliği uygulamanın hayat çevriminin her fazında uygun dikkatin, hizmetin oluşturulması ve dağıtılması açısından büyük önemi vardır.

En iyi uygulamalar bize, hizmet yaratma ve dağıtım işinin hizalanmasının ve organizasyonel değişim çabasının gösterilmesinin uzun zaman aldığını göstermiştir. Tecrübe ile sabittir ki bu yolda birçok tuzak vardır.

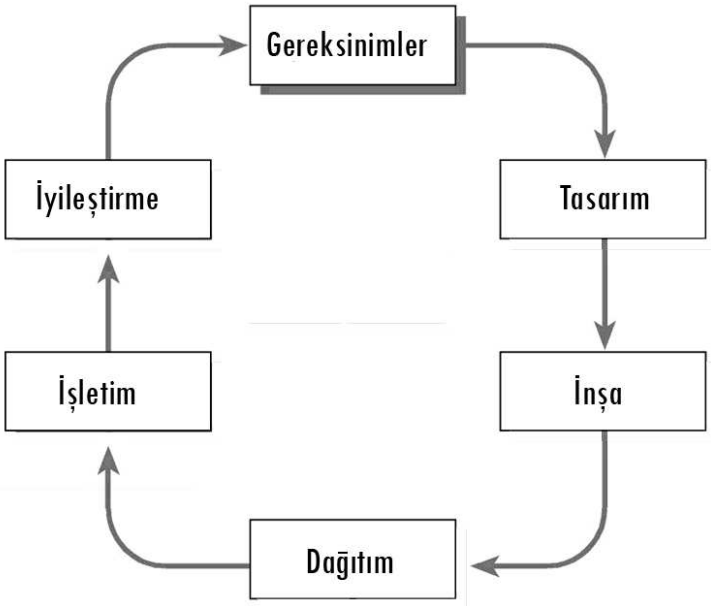
6.1.3 Hayat Çevirimindeki Fazlar

Uygulama Yönetimi hayat çevriminde çok çeşitli perspektifler vardır, fakat iki ayrı kampa bölünmüşlerdir. Bir kısmı uygulama geliştirmeyi vurgularken, diğerleri Hizmet Yönetimini vurgular. Uygulama geliştirme açısından global ve

detaylı tasarım arasında ayrılırlar. Birim test ve sistem test arasında ve sonunda çok genel bir fazda “bakım” olarak isimlendirilir. Hizmet Yönetimi perspektifi, hali hazırda geliştirilmiş bir uygulamanın yükleme ve operasyona hazırlanması ile başlar.

ITIL da organizasyon içinde bu iki perspektifin olması gereken nokta üzerinde yoğunlaşıyor. Uygulama gelişimi 6 fazdan meydana geliyor ve her bir faz içinde bu iki görüşün de hizalanması gereklidir. Uygulama geliştirme ve Hizmet Yönetimini birleştiren bu altı faz şunlardır:

6.1.3.1 İhtiyaçlar



Şekil 0.2 İhtiyaçlar safhasının uygulama hayat çevrimindeki yeri

Organizasyonun iş tarafı ihtiyaçları temel alınmış ve yeni uygulama için tüm gereksinimlerin toparlandığı fazdır. Bilinmesi önemli olan 3 tip gereksinimin olduğudur. Bunlar; fonksiyonel gereksinimler, fonksiyonel olmayan gereksinimler ve gereksinimlerin kullanılabilirliğidir. Bir uygulama seferini bu aşamadan başlar. Geliştirme takımı, iş tarafının karar vericileri ile çok yakın bir iş birliği içinde uygulamadan beklentilerini belirlerler.

6.1.3.1

Fonksiyonel Gereksinimler

Fonksiyonel gereksinimler, iş fonksiyonlarını desteklemeye özgüdürler. Uygulamadan beklenen hizmetler, görevler ve fonksiyonlar olarak ifade edilebilirler. Kullanım diyagramları ya da sistem kontekst diyagramları gibi metotlar etrafında

fonksiyonel ihtiyaçların belirtilmesidir. Sistem kontekst diyagramı, uygulama tarafından kullanılan veri ve kaynaklar üzerinde ve uygulama sistemi üzerindeki bilgilerin değiş tokuşunu gösterir.

Kullanım vakaları modeli, kullanıcı, sınıf ya da rol gibi sisteme müdahalesi olan nesnelerin uygulama ile etkileşimlerinin meydana getirdiği sonuçları izler. Uygulamada amaçlanan sınırlarının tam halini fonksiyonel yeterliliğini görebilmek ve geliştirebilmek için kullanılır. İş tarafındaki kullanıcılar ve uygulama geliştiriciler arasında haberleşmeyi, iletişimi kurmada da ayrıca çok faydası vardır. Gereksinimleri ortaya koymak için çok iyi bir araçtır.

Kullanım vakaları uygulamanın test edilme aşamalarında, çalışması beklenen işlemlerin sırasını ve uygunluğunu deneyebilmek için de önemlidir. Kullanım vakaları ile uygulamanın desteklemek zorunda olduğu tüm durumlar tanımlanabilir.

Kullanım vakaları ve sistem kontekst diyagramları gibi birçok modelleme tekniği uygulamalarda kullanılabilir. Bu modellemelerle uygulamanın statik ve dinamik karakteristiğini belirlenir.

6.1.3.1

Fonksiyonel Olmayan Gereksinimler

Fonksiyonel olmayan gereksinimler, Hizmet Yönetimi perspektifinden bakıldığında; sistem yönetimi, güvenliği, işletimi ve dağıtımı gibi konulara cevap verebilen ve elde edilebilir olmasıdır. Gereksinimlerin kullanılabilirliği, son kullanıcı ihtiyaçlarını adresleyebilen ve sonuçta sistemin kullanımını kolaylaştıran özelliklere sahip olması halidir.

Fonksiyonel olmayan gereksinimlerin kategorileri:

- Yönetilebilirlik (Çalışıyor mu? Başarısız oldu mu? Nasıl başarısız oldu?)
- Verimlilik (Ne kadar kaynak tüketiyor?)
- Etkinlik(Doğru şeyleri yapıyor ve bilgi işlem iş tarafının hedeflerini ve etkin iş planlama süreçlerinin girişlerini kavrıyor)
- Diğer hizmet ve uygulamalar ile çalışabilirliği
- Erişilebilirliliği ve güvenilirliği (Ne kadar güvenilir?)
- Kapasite ve performansı

- Güvenliği
- Doğrulanabilirliği (Ne hızda başarısız olduğunda geri getirilebiliyor? Bir ve birden fazla geri getirme prosedürü veya aracı mevcut mu?)
- Yüklenilebilirliği (Ne kadar çaba ile uygulama yükleniyor? Otomatiğe alınmış yükleme prosedürleri kullanıyor mu?)
- Kontrol edilebilirliği (Standardize edilmiş konfigürasyonları var mıdır? İsteğe bağlı olarak değiştirilebiliyor mu?)
- Bakım yapılabilirliliği (Uygulama bakım yapılabilecek şekilde ayarlanabiliyor mu?)
- İşletilebilirliği (Uygulamalar diğer uygulamaların fonksiyonelliğine mani oluyor mu?)
- Güvenilirliği (Uygulamalar dışarıdaki kullanıcıların sisteme karışmamalarına göre tasarlanmış mı?)

Fonksiyonel olmayan ihtiyaçlar uygulamanın inşasına başlamadan önce kalite değerlerini tanımlamak için kullanılabilir.

6.1.3.1

Kullanabilme gereksinimleri

Kullanıcıların beklentileri ile sistemin sağlayabileceklerini karşılama amacı, kullanılabilme gereksinimlerinin birincil gayesidir.

Ürünün çalışması:

- Kullanılabilirlik değerlendirmesi için başarım standartlarını tespit etmek
- Test planlarının ve testlerin kullanılabilirliliği için test senaryoları tanımlamak

6.1.3.1

Vaka değişimleri

Gereksinimlerin bölgesinde önemli bir geliştirme konusu da vaka değişimleri maddesidir. Gelecekte ihtiyaç duyulacak şekilde uygulamanın fonksiyonelliğinin, değişimlerinin belirtilmesidir. İhtiyaç duyulan diğer dokümanlarla, vaka değişimlerinin açık bir şekilde içeriği ve gidişatı belirtilmelidir. Gelecekte maliyet

getirebilecek durumların mimari ve tasarımlarının çıkartılması bu madde altında gerçekleşir. Uygulamanın sahibi tarafından bu ekstra maliyetin karşılanması gerekir.

Vaka değişimleri; yazılımın ileride ortaya çıkabilecek maliyetleri göz önüne alınarak uygulamanın mimarisinin değiştirilmesi, gelecekteki maliyetlerin düşürülmesi, ortaya çıkacak etkilerin minimize edilmesi için önem arz eder.

6.1.3.1

Test gereksinimleri

İhtiyaçların test edilmesi, genellikle yazılımın geliştirmesi sırasında gözden kaçar. Başka teknik ve araçların kullanımı, genel özet ve tartışmalar, gereksinim modellerinin kalitesini geliştirmeye ve yardımcı olmaya yarar. Bu aşamada atlatılamayan her anlaşmazlık daha sonra zaman ve kalite kaybına neden olur.

6.1.3.1

Gereksinimlerin yönetimi için kontrol listesi

İhtiyaçların belirlenmesi fazında, tüm Hizmet Yönetimi fonksiyonlarının tam olarak düşünülmüş ve adreslenmiş olması önemlidir. Aşağıdaki tablo ihtiyaçların yönetilmesi fazı için örnekler içerir.

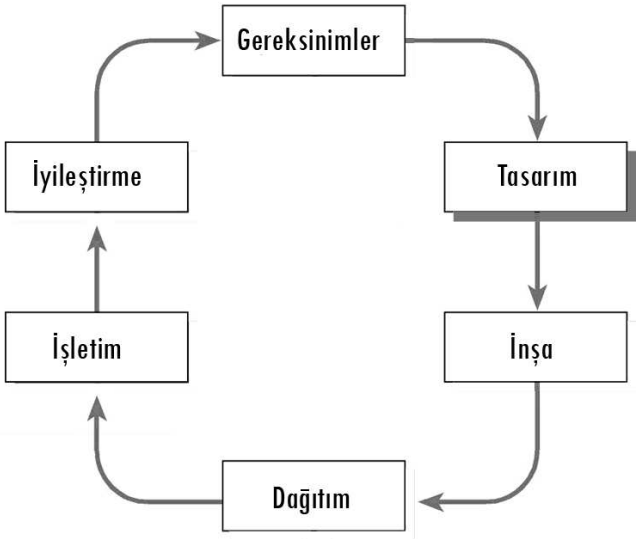
Hizmet Yönetim Fonksiyonları	İhtiyaçlar fazını yönetilebilme örneği
Konfigürasyon Yönetimi (Bir sistem, kaydedilen ve raporlanan konfigürasyon elementlerinin durumu, değişim talepleri ve içindeki konfigürasyon elementlerinin tanımlanması ve konfigürasyon elementlerinin doğruluk ve bütünlüğünün, tanımlanması ve kimliklendirilimesi prosesleridir.)	Uygulamanın çalışmak için ihtiyaç duyduğu çevrenin tanımlanması
Değişim Yönetimi (Hizmet ve altyapının onaylanmış değişimlerden en az bozulmaya uğraması prosesleridir.)	Değişim durumlarının belirtilmesi.
Sürüm Yönetimi	Uygulamaların hangi durumlarda sürümünün yayınlanması gerektiğini tanımla.

	Sürümlerin diğer uygulamaların sürümleri ile ilişkisini tanımla.
Güvenlik Yönetimi	Güvenlik ihtiyaçları ve uygulama gereksinimlerinin neler olduğuna karar ver.
Olay Yönetimi	Uygulamadaki hatalar ile nasıl başa çıkılacağını tanımla. Hata ile başa çıkmak için kurumsal standartların neler olacağına karar ver. Eğer kurumsal standartlarda nasıl çözüleceği belirtilmemiş bir hata meydana gelirse ne olacağına karar ver.
Problem Yönetimi (Harici olayların, müşteri kusurlarının veya insan hatalarının hizmet ve altyapı üzerinde meydana getireceği etkilerininin en aza indirgenmesidir)	Problem yönetim proseslerinin nasıl kullanıldığını öğren.
Kapasite Yönetimi	Çevredeki kullanılabilir kapasite ve kabiliyeti öğren. Çözümler (donanım, ağ gibi) için kapasite ihtiyacını öğren.
Erişilebilirlik Yönetimi	Uygulamanın, iş tarafı taleplerinin karşılanmasında ne kadar erişilebilir olduğunu belirle. İsteklerin 7x24x365 seviyesinde mi karşılayacağına karar ver.
Hizmet Devamlılık Yönetimi	Hizmet olmaksızın iş tarafının operasyonu ne kadar sürdürebileceğine karar ver. Hizmetler olmaksızın iş tarafının operasyonu nasıl sürdüreceğine karar ver.
Hizmet Düzeyi Yönetimi	İş tarafının ne seviyede hizmete ihtiyaç

	duyacağını tanımla.
Finansal Yönetim	Uygulama için finansal durumu tanımla. Bu uygulamanın nasıl bir katkı yaptığını ve nasıl bir değer ve bedelin bu uygulama için idare edildiğine karar ver. Uygulamanın devam eden(işletme ve bakım gibi) ve geliştirme maliyetlerini kim tarafından karşılanacağını tanımla.

Tablo 0.1 İhtiyaçlar fazında yönetim kontrol listesi.

6.1.3.2 Tasarım



Şekil 0.3 Tasarım safhasının uygulama hayat çevrimindeki yeri

Bu faz, gereksinimlerin, ayrıntılı özelliklere çevrildiği fazdır. Hedef uygulama tasarımının bu organizasyonun gereksinimlerini tatmin edecek seviyede olmasıdır. Tasarım, uygulamanın üzerinde çalışmak zorunda olduğu, kendi tasarımını ve çevresel ya da operasyonel modelinin tasarımını içerir. Bu fazın uygulama ve operasyonel modellerinin içeriğini ve yapısını etkileyen en önemli yönü mimari düşüncelerdir. Mimari düşünceler uygulama ve operasyonel modelin güçlü ilişkilerinin kurulmasında ve aynı seviyeye getirilmesinde büyük önem arz eder.

Tasarım safhası uygulama hayat çevriminin en önemli safhalarından birisidir. Akılda tasarlanmış, işletilebilir bir uygulamanın Uygulama Yönetimini sağlar. Bu safha ihtiyaçlar safhasındaki çıktıları girdi olarak alır ve bunları inşa edilecek uygulamanın şartnamesi olarak kabul eder. Uygulamanın inşası tamamlandıktan

sonra bir değişiklik yapılması hem çok zor, hem de çok pahalıya mal olacağı yaygın bir kabuldür. Bu değişimin sonrasında başarı şansı da çok düşüktür. Örneğin, bir uygulama geliştirmeyi, bina inşaatı ile kıyaslayalım. Binanın yapı analizi, statik ölçümleri, çizimleri gibi inşaatında temeli, bulunduğu yere uygun yapılanması tamamlandıktan sonra; bir kapının yerinin değiştirilmesi ihtiyacı meydana gelsin. Bir kapının yer değiştirmesi sadece bir duvarın yıkılması ve diğerinin düzenlenmesinden fazla olarak, binanın statik yapısından iç ve dış ortama göre birçok şeyi değiştirmiş olacaktır.

Genel olarak tasarım safhasında, ihtiyaçlar safhasında toparlanmış gereksinimlere daha fazla detay eklenir. Mimari modelin bir başka bakış açısı da var olan ortamına uygulamanın gömülmesidir. Hali hazırdaki alt yapının hangi parçası, istenilen yeni fonksiyonlara destek verecek? Hali hazırdaki sunucu veya ağ kullanılabilir mi? Hangi etkililik ile kullanılabilir? İhtiyaç duyulan fonksiyonlara hali hazırdaki uygulamadan erişilebilir ve yararlanılabilir mi?

Sistemin karakteristiğini, uygulamanın düzgün dağıtımı ve işletimini yapabilmek için, bunları açık şekilde ifade eden dokümantasyona ihtiyaç vardır. Ancak bu sıklıkla ihmal edilmiş ya da yok sayılmıştır.

Tasarım safhası çözüm için tüm ihtiyaçları göz önüne almaya ve bir tasarım başlangıcı içinde birleştirir. Bu sadece temel bir başlangıç noktasını yazılım geliştiricilere verdik demek değildir. Ayrıca müşterilerin de ihtiyaçlarını öğrenmek için sorulan soruların cevaplanmış olmasıdır.

Bir çözümün tasarımı her zaman her açıdan öngörülemez. Bir çözüm geliştirildikçe, bazı şeylerin nasıl yapılıp, nasıl yapılmayacağı öğrenilmiş olur. Ortaya çıkabilecek planlanamamış engeller, genellikle müşteri ihtiyaçlarının anlaşılmasını ve azaltılmasını gerektirir.

Anahtar olan, esnek bir tasarım yaratabilmektir, böylece bir değişiklik yapmak geliştiricileri tasarım evresinin en başına göndermez. Bunun olma şansını azaltan çok sayıda yaklaşım vardır.

- Fonksiyonel olmayan gereksinimler/yönetilebilirlik için tasarım
- Sürdürülebilir risk takvimleme teknikleri kullanma
- Kazanç kayıp idaresi (managing trade-off)
- Uygulama kullanımı(tasarım rehberinden bağımsız)
- Tasarım takımının rollerini organize etmek

- Fonksiyonel olmaya ihtiyaçlar için tasarım

6.1.3.2

Fonksiyonel Olmayan Gereksinimler/Yönetilebilirlik İçin Tasarım

Fonksiyonel olmayan gereksinimlere verilen önem fonksiyonel ihtiyaçlar ile aynıdır ve tasarım safhası için aynı öneme sahiptir. Fonksiyonel olmayan gereksinimlerine örnek verelim. Bunlar; erişilebilirlik, bakılabilirlik ve güvenilirliktir . Kullanıcı ara yüzünün tasarımının atlanması modern uygulama geliştirme projelerinde kabul edilemez bir durumdur. Ancak, birçok organizasyon yönetilebilirliği unutur ya da ihmal eder.

6.1.3.2

Sürdürülebilir Risk Takvimlemesi

Müşteri ihtiyaçlarını karşılamak için yüksek risk görevleri, yüksek öncelikli ve risk öncelikli olarak takvimler. Bu demektir ki, en yüksek risk aktiviteleri önce yapılır. Böylece eğer bazı şeyler kötü giderse veya planlandığı gibi gitmezse, kurtarmak ve projenin kritik sürece girmesini önlemiş oluruz.

6.1.3.2

Kazanç Kayıp İdaresi (Managing Trade Off)

Kazanç kayıp idaresi, kaynaklar ile proje takvimi arasındaki dengenin kurulmasını amaç edinir. Geliştirme takımları bu dengeyi genellikle fonksiyonel olmayan gereksinimler pahasına tamamlamaya çalışırlar.

6.1.3.2

Uygulama Bağımsız Tasarım İlkeleri ve Uygulama Çatısı

Uygulama bağımsız tasarım ilkelerinin kullanımı, geliştirme içinde yönetilebilirliği destekler. Verimli işletilebilirlik ve yönetilebilirlik tüm tasarım süreçlerinin standardı olmaktadır.

6.1.3.2

Tasarım Yönetimi Kontrol Listesi

Hizmet Yönetim Fonksiyonları	Tasarım safhası yönetim kontrolleri örneği
Konfigürasyon Yönetimi	Konfigürasyon yönetimi için şirketin kullandığı standartlardan tasarımcılar haberdar mı?
	Nasıl bir tasarım organizasyonel standartları kabul edilebilir konfigürasyonlar için karşılar?
	Versiyon kontrolünü tasarımı destekliyor mu?
Değişim Yönetimi	Uygulamadaki değişimler ile bu tasarım başa çıkabilecek mi?
	Organizasyon tarafından kullanılan değişim yönetim proseslerini tasarımcılar anlıyorlar mı?
Sürüm Yönetimi	Uygulama tasarımcıları, uygulama sürümü yapabilmek için standartları ve araçları biliyorlar mı?
	Tasarım, uygulamanın basit ve verimli bir yoldan sürümün yapılmasını nasıl sağlayacak?
Güvenlik Yönetimi	Dizayn, uygulamanın güvenlikle tasarlanmasını nasıl sağlar?
Olay Yönetimi	Bir şeyler hatalı gittiğinde tasarım, olayların gidişatını hafifletiyor mu?
	Tasarım, organizasyonun olay yönetim sistemi ile uyumlu mu?
	Tasarım, olayların kayıt altında tutulması ve algılanmasını otomatik sağlıyor mu?
Kapasite Yönetimi	Tasarımcılar, organizasyon içinde kullanılan kapasite yönetiminden haberdarlar mı?
	Operasyonlar ve performans nasıl ölçülür?

	Kapasite ihtiyaçlarını karşılayan tasarımı sağlayan bir modelleme mi kullanılıyor?
Erişilebilirlik Yönetimi	Tasarım, uygulamanın erişilebilirlik gereksinimlerini adresliyor mu?
	Uygulama, organizasyonun kurtarma ve yedeklemesine uygun bir tasarıma mı sahip?
Hizmet Devamlılığı Yönetimi	Tasarım, organizasyonun hizmet sürekliliği gereksinimlerini nasıl karşılıyor?
Hizmet Seviyesi Yönetimi	Tasarım, organizasyonun hizmet seviyesi anlaşmasının gereksinimlerini nasıl karşılıyor?
Finansal Yönetim	Tasarım, uygulama için finansal gereksinimleri karşılıyor mu?
	Tasarım, uygulamanın son hali ile yatırımın geri dönüşünü nasıl sağlıyor?

Tablo 0.2 Tasarım fazının yönetim kontrol listesi

Tasarım İlkeleri İle Problemler

Tasarım fazı sırasında, tasarımcılar tasarım ilkelerinin fonksiyonel olmayan gereksinimleri örtmeye nerede başarısızlığa uğrayacağını tanımlamalıdırlar.Bu uygulamanın değiştirme ve güncellemelerinde, gelecekte ortaya çıkabilecekleri yansıtabilmek adına bir geri besleme sağlar.

Gereksinimlerin Test Edilmesi

Fonksiyonel olan ve olmayan gereksinimlerin düzgün belirtilip dokümante edilmesi gerekir. Bu test takımının statik test yöntemini kullanarak fonksiyonel olan ya da olmayan gereksinimlerin uygulamanın tasarım kısmında düzgün adreslendiğinin doğrulanmasını sağlar. Birkaç teknik bu fonksiyonel olan veya olmayan gereksinimlerini uyumunu doğrulamak için kullanılabilir. Örneğin:

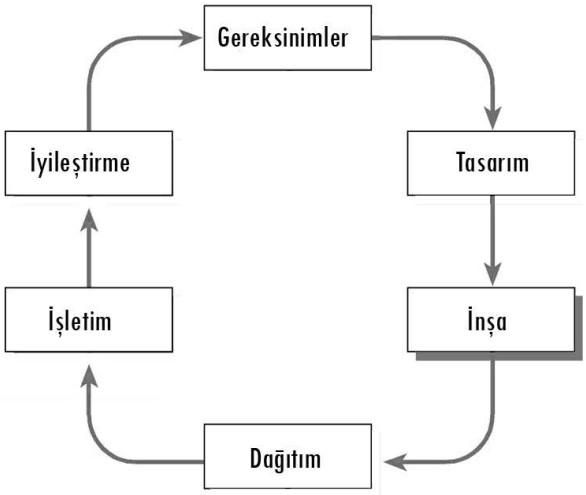
Teftiş Etme: en çok resmi gözden geçirme metodudur.

Masa başı Denetimi: en az resmi süreçtir. Bir ürün sahibinin kendi işinin kusurlarını tanımlamasıdır.

Adım Adım: Yaratıcısına ek olarak, ürünün bir ya da daha fazla kişi tarafından adım adım okunarak ve/veya görsel inceleme ile teftiş edilmesi.

Gözden Geçirme: Proje, ürün ya da geliştirme sürecine ve etkilenme konularına, projenin kontrol altında, kalite güvencesi sağlayacak şekilde odaklanılması. Gözden geçirme; inşa edilmeye devam edilen şeyin karakteristikleri ile, nasıl inşa edildiği ve inşasında kullanılan kaynaklar ile ilgilenir.

6.1.3.3 İnşa



Şekil 0.4 İnşa safhasının uygulama hayat çevrimindeki yeri

İnşa fazı, uygulama ve operasyonel modellerin dağıtımı için hazırlanma safhasıdır. Uygulama bileşenleri kodlanır, entegre edilir ve sınanır. Genellikle sürüm ve test ortamı ayrıdır. Deneme ortamı uygulama ve operasyonel modelin kombinasyonu içindir. İnşa safhası aşağıdaki başlıkları içerir:

- Tutarlı kodlama düzeni
- Uygulama bağımsız inşa ilkeleri
- İşletilebilirlik denemesi
- İnşa safhasının yönetimi için kontrol listesi
- İnşa takımının rollerinin organize edilmesi

6.1.3.3.1 Tutarlı Kodlama Düzeni

Uygulamanın geliştirme safhasında herkesin kolaylıkla okuyabileceği, anlayabileceği ve idare edebileceği kodlama tarzında tüm uygulamanın standardize edilmesidir. Organizasyonun kendine has kodlama standardı ile daha okunaklı, kesin sonuçlu açıkça anlaşılan kaynak koduna sahip olmalıdır. Ancak bu standart ile kodun ve uygulamanın idare edilmesi sağlanabilir. Organizasyon içindeki yazılımcıların ortak çalışma projelerinde birbirilerinin kodlarını anlayabilmeleri, hata düzeltme

döneminde tekrar kod ile yapılmak istenenin, amacı kavrayıp, tekrar kodu düzenlemeleri için kodlamanın tutarlı ve düzenli olması şarttır. Bu sayede uygulamanın hayat çevrimi aksamadan sürecektir.

6.1.3.3.2 Uygulama Bağımsız İnşa İlkeleri

Birçok geliştirme aracı çeşitli örnek kodlama bileşenleri, uygulama geliştirmede sağlamaktadır. Bu örnek kodlar gibi, yazılımcının kendine ait tekrar kullanmaya hazır çeşitli örnek kodları mevcuttur.

Başka uygulamalarda iskeleti verilmiş modellerden, kod üretimini gerçekleştirebilmek için örnek kodlardan ve uygulama çatısından faydalanılır. Bu üretilen kodlarda yazılımcının girebileceği bölümler ayrıca belirtilmiş olarak verilir. Örnek kodlar ve uygulama çatısı, uygulamanın ilerlemesi hızlı ve önceki tecrübelerin devam olması nedeniyle güvenli sürdürmeyi sağlar.

Üretiminde uygulamaya gömülü olarak gelen içeriklere gömülü uygulamalar denir. Gömülü uygulamaları sadece konu ile ilgili yazılımcı kullanılır. Örneğin, Microsoft Windows© API(Application Program Interface, yüksek düzeyli dilde yazılmış bir uygulama programının, işletim sisteminin fonksiyon ve verilerini kullanmasına imkan sağlayan arayüz.) ya da Microsoft Windows© Management Instrumentation (WMI, tüm sistem bilgilerinin saklandığı CIM(Common Information Model, yaygın bilgi modeli) adı verilen bir veri tabanına sahiptir. Tüm sorgular bu veri tabanı üzerinden yapılır. Sorgu dilinin adı WQL(WMI Query Language, WMI sorgulama dili)'dir ve SQL(Structured Query Language, yapısal sorgulama dili)'e çok benzer. .NET ' in System.Management isim alanındaki sınıflar ile bu veritabanını sorgulayan programlar yapılabilir. System.Management.Instrumentation isim alanındaki ise CIM içindeki bilgileri değiştirmeye yönelik sınıflar mevcuttur.) gömülü uygulamalara en güzel örneklerdir. API yi ele alalım:

Aslında her yazılım dönüşümü bir API’dir, sadece görüntüleme birimleri için kullanım alanları daha geniştir. Şöyle bir örnek ile inceleyelim:

1. Kâğıda "deneme" yazalım. d,e,n,e,m,e Harflerini yan yana yazmamız gerek.

2. Bu harfleri ayrı ayrı inceleyip, 10x10’luk bir matriste siyah ve beyaz olarak nokta nokta ayrıştıralım.

3. Merkezi işlem birimimizin(CPU, Central Processing Unit) bu matrisleri okuyup görüntüleme aygıtımıza göndermesi için bir program yapmış olalım.

4. Ekran kartının bu görüntü önbelleğindeki taranmış objeleri, doğru şekilde tarayıp, doğru sinyaller ile monitöre iletme işi bir API ’dir.

Bir uygulama en başta iş tarafının ihtiyaçlarını karşılamak için yapıldığına göre, bittiğinde ya da inşanın her safhasında bu iş ihtiyaçlarını karşılayıp/karşılamadığını da test etmek gerekmektedir. Bu testlerin bir düzen içinde ve dokümante edilmesi gerekmektedir. İşletilebilirlik testi uygulamanın kullanımından önce gördüğü en ciddi sınamaları içerir.

En basit seviyede denemeler yazılımcıların kodlama esnasında yaptıkları küçük denemeler olup, bunların daha kapsamlısının inşa sürecinin ana safhaları bitiminde ve tüm inşa safhasının bitiminde iş ihtiyaçlarını, fonksiyonel olmayan gereksinimleri karşılama miktarını da daha profesyonel ve inşa sürecinin dışında kalan, ama işletme sürecinde bulunacak kişilerin yapması en uygunudur.

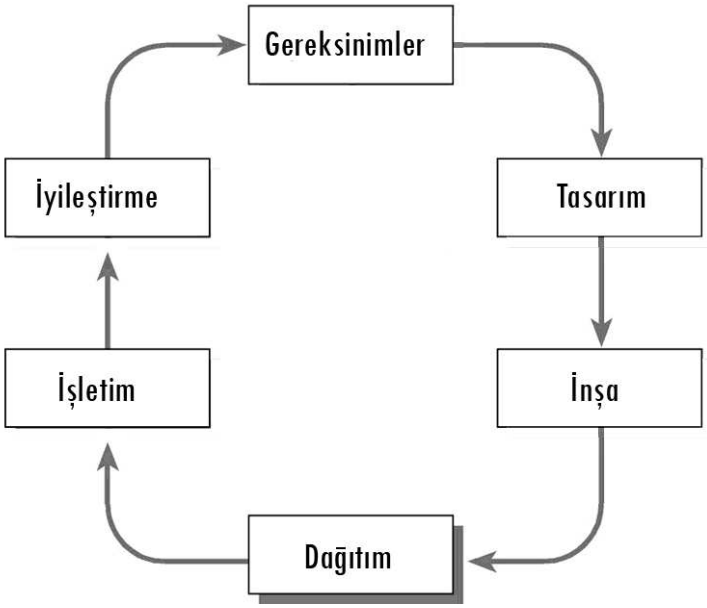
Uygulama birçok revizyondan geçecek ve birçok sürüm sayısına ulaşacaktır. Her sürümde yine test döngüsü tekrar edilmelidir. Uygulamada meydana gelecek bir değişimin Konfigürasyon Yönetim Sisteminde değişimlerin tamamlanması ile Sürüm Yönetimine ve oradan da Olay Yönetimine hareketi oluşacaktır. Bu döngüde oluşabilecek hataların giderilmesi en elzem olanıdır çünkü sistem canlıdır ve operasyonel faaliyetlerin aksamaması için hatasız olması gerekmektedir.

Hizmet Yönetimi Fonksiyonları	İnşa safhası yönetilebilirlik kontrol listesi örnekleri
Konfigürasyon Yönetimi	Kullanılan konfigürasyon yönetimi standartlarına uygun olarak uygulama, geliştiriciler tarafından inşa edildi mi? Ürün katalogu dahilinde uygulama, programlar ve araçları mı kullanıyor? Uygulama sürüm kontrol ve yönetimi için bir destek içeriyor mu?
Değişim Yönetimi	Şirketin Değişim Yönetim proseslerine göre uygulama inşa edilip test edildi mi?
Sürüm Yönetimi	Basit ve verimli bir yol ile ortama aktarımı inşa ve test edildi mi?

Güvenlik Yönetimi	Bu aktivite için inşa prosesleri en iyi güvenlik uygulamalarını izliyor mu?
Olay Yönetimi	Uygulamanın inşasında ve testlerinde bir şeyler ters gittiğinde basit olay yönetimi prosesleri var mı? Yapılan testler, organizasyonel olay yönetimi sistemi ile uyumlu mudur?
Problem Yönetimi	Program, analiz ve problem yönetiminin kök nedenlerini kolaylaştıracak bilgiyi bize sağlıyor mu?
Kapasite Yönetimi	Uygulama, kapasite ihtiyaçlarını karşılayacak şekilde inşa edildi mi? Kapasite bilgisinin temini, uygulamada test edilerek ve onaylanarak sağlandı mı? Uygulama içinde stres ve hacim karakteristikleri belirtilerek inşa edildi mi?
Erişilebilirlik Yönetimi	Uygulamanın erişilebilirlik gereksinimleri nasıl adreslenmiş ve test edilmiş? Uygulamada, organizasyonun yedekleme ve kurtarma kabiliyetlerini karşılayan hangi testleri yapılmıştır? Uygulama stres altında kaldığı zaman nasıl bir tepki veriyor?
Hizmet Sürekliliği Yönetimi	Uygulama iş tarafının kurtarma proseslerine destek verecek şekilde mi inşa edilmiş ve bu yeteneği test edilmiş mi?
Hizmet Seviyesi Yönetimi	Organizasyon Hizmet Seviye Anlaşması gereksinimleri karşılan bir uygulama mı ve bu test edilmiş mi?
Finansal Yönetim	Uygulama, finansal bilginin dağıtımını sağlar mı ve bu konuda test edilmiş mi?

Tablo 0.3 İnşa safhasının yönetim kontrol listesi.

6.1.3.4 Dağıtımı



Şekil 0.5 Dağıtım safhasının uygulama hayat çevrimindeki yeri

Bu fazda, operasyonel ve uygulama modeli yayılmış, dağıtılmıştır. Operasyonel model en üst operasyonel modele yüklenmiş ve var olan bilgi işlem ortamında ITIL'da tanımlı dağıtım süreçleri kullanılarak yayılmıştır. Uygulama inşasının tamamlanması ile dağıtımına geçilmesi ayrıca önemli bir safhadır. Bu evrede dağıtımı yapabilmek için yeterli bir düşünce ve planlamaya sahip olmak bir sorun çıkmasına mani olacaktır.

Dağıtım safhası aşağıdaki başlıkları içerir:

- Dağıtımın planlanması
- Bir dağıtımın onaylanması
- Uygulamanın yayılması
- Pilot Uygulama
- Dağıtım evresi için yönetim kontrol listesi
- Dağıtım takımının rollerinin organize edilmesi

6.1.3.4.1 Dağıtımın Planlanması

Organizasyon içinde doğru planlanmış ve uyarlanmış dağıtımlar belirgin bir fark yaratır. Tam tersine, iyi dağıtımı yapılamayan uygulamalar her ne kadar başarılı olursa olsun sürekli problem meydana getirirler ve sürekli problem yönetimini tetiklerler. En kötüsü de ortamı felakete uğratırlar.

Küçük şirketler çok karmaşık bir düzene sahip olmadıkları için uygulama dağıtımı, daha büyük ve hatta çok uluslu şirketlerin sahip oldukları karmaşık yapılara nispeten daha kolaydır. Bu aktiviteyi tamamlamak için planlayıcıların aşağıdaki sorulara cevap vermesi gerekir:

Dağıtım için ne gereklidir?	Dağıtımı yapılacak uygulama iyi anlaşılmış mı? Dağıtımı yapılacak uygulama hangi bileşenlerden meydana geliyor? Kritik iş ihtiyaçlarını karşılamak için gerekli mi?
Kullanıcılar kimlerdir?	Uygulamanın dağıtımdan hangi kullanıcı ve bölümler etkilenecek? Özel bir eğitime ihtiyaç duyuluyor mu?
Kullanıcılar nerededirler?	Tüm yerel kullanıcılara ve sistemlere, yerel ya da uzak dağıtımlar mı yapılacak ve bu durum lojistiği nasıl etkileyecek?
Dağıtımın tamamlanmasına ne zaman ihtiyaç duyulur?	Bu dağıtım tam tarih ve saatinde mi tamamlanmalı veya esnek bir takvimlemeye göre dağıtımın tamamlanmasında bir sakınca var mı?
Neden dağıtım yapılıyor?	Bu dağıtım için bir problem çözmeye gerek var mı veya kullanıcıların ne ile karşılacaklarını anlatan bazı yeni fonksiyonelliklere gerek var mı?
Kritik başarı nedir?	Dağıtımın başarı ile tamamlandığını nasıl anlayacağız? Dağıtımın bittiğini nasıl anlayacağız?

Tablo 0.4 Dağıtım safhasında cevaplanması gereken örnek sorular.

Yukarıdaki sorular ve cevapları uygulama dağıtımının nasıl yapılması ve yapıldığını öğrenebilmemizi sağlayacaktır. Dağıtımda kontrol listesi ve dağıtımın takvimi iyi dokümante edilmelidir. Dağıtımın geri alınması durumlarında dağıtımı yapacak personelin ya da kullanıcıların eğitilmesi iyi dokümante edilmelidir.

6.1.3.4.2 Dağıtımın Onaylanması

Yukarıdaki bilgiler bir kez kullanılabilir olduğunda, bir sonraki adım, yol haritasına uygun dağıtımın kimler tarafından, değişim yönetimi süreçlerinin kimler tarafından yapılacağına karar verilmesi gerekir.

Bu gerçekte kimlerin yeni isteklerde bulunabileceği ve isteklerin değişim yönetiminde kimler tarafından karşılanacağının sonucunu ve cevabını doğurur.

6.1.3.4.3 Uygulamaların Dağıtımı

Uygulama için başarılı dağıtım stratejilerinden en önemlilerinden biri de Kritik Başarı Etkeni'dir. Bir uygulama, CD, disk, ağ, internet ya da intranet üzerinden aşağıdaki kriterler düşünülerek dağıtılabilir:

Paketleme: Uygulamanın bir veya daha fazla dosyasının bir kurulum dosyası ile dağıtılmasıdır. İdeal olarak bu işlemleri yapan insan gücünün araya girmesine gerek duyulmayan programlar vardır.

Yayılması: Uygulamanın bileşenleri açık bir şekilde yayılma esnasında belirtilmelidir. Bu diğer uygulamalar veya sistemle çelişip çelişmeyeceğini anlayabilmeyi sağlayacaktır.

Yazılımın esnek dağıtımı: Doğru kullanıcılara, kullanıcı gruplarına ya da sistemlere dağıtıldığını görebilme kriteridir. Birçok sistem yönetim araçları, hedef kuralların bir kümesini sağlar. Bu da uygulamanın yazılım ya da donanım envanter özelliklerinin değiştirilmesi veya konfigürasyonunu yöneticilere ya da kurulumu gerçekleştiren kişilere sağlar.

Yayılmanın fonksiyonelliği: Araçların kullanımı ayrıca yayılmanın bir takvime göre yapılmasını sağlar. Uygulamanın doğru yer ve doğru zamanda daha az sistem kaynağı tüketerek yayılmasını, güncellenmesini sağlar.

Çekme teknolojileri: Sunucu üzerindeki uygulamanın ihtiyaç hâsıl olduğu zamanlarda çekilebilmesini sağlayan teknolojilerdir. Bu istemci tarafında kullanıcının gerekli bileşenleri yükleyip yüklememesini bilmemesi durumlarında çok kullanışlıdır.

Kendi kendini iyileştirme: Yazılım kimi zaman yaşabilecek sorunlarda, örneğin ağ bağlantılarından kaynaklanan, disk okumadan kaynaklanan problemlerde kendi kendine tekrar hata giderilmesi için yükleme ya da güncellemeyi yapabilmesidir.

Yamalama: Dağıtımın başarısı ile yazılımın boyutu değişime uğrayacaktır ve bu yavaş ağlarda ciddi bir zaman alacaktır. Tüm uygulama yerine dağıtımdaki değişikliğin ağ üzerinden gönderilmesi daha iyi ağ başarımına neden olacaktır.

Geri besleme ve raporlama: Dağıtım tamamlandığında yapılan değişikliklerin listesinin veya oluşmuş hataların raporlanmasıdır.

6.1.3.4.4 Pilot Uygulama

Laboratuarda uygulamanın test edilmesi elbette ki yeterli değildir. Uygulamanın gerçek bir ortamda denenmesi ve sonuçlarının görülebilmesi için organizasyon içinde bir bölümde uygulamanın kurulması gerekmektedir. Böylece uygulamanın gerçek iş verileri ile çalışması görülebilecek ve ihtiyaçların ilk bakışta ya da küçük ölçekte hangi kısımlarına cevap verip veremediği görülebilecektir.

Pilot alanda yapılacak denemelerde kullanılacak bilgisayar sayısından, sunucu, ağ ekipmanları, kullanıcılara kadar organizasyonun yapılması ve önceliklere göre, çoklu çalışmanın sağlanması gerekmektedir. Bu pilot uygulamanın da başarılı olabilmesi kullanıcılar için verilecek eğitimin, haftada kaç kişiye ve ne sürede verileceği, eğitim metotlarına kadar belirlenmesi ve planlanmasına bağlıdır. Buradan elde edilen bilgiler doğrultusunda yeniden uygulamanın değişime gitmesi, tüm yayılımın gerçekleşip değişime gitmesinden daha iyi olacaktır.

6.1.3.4.5 Dağıtım Yönetimi Kontrol Listesi

Hizmet Yönetimi fonksiyonları	Dağıtım fazının yönetilebilirlik kontrol listesi örneği
Konfigürasyon Yönetimi	Dağıtım takımı, güncel envanteri kullanarak dağıtımı planlayıp tamamlayabiliyor mu?
Değişim Yönetimi	Uygulama kurumu Değişim Yönetim proses ve standartlarını kullanan bir dağıtıma mı sahip?
Sürüm Yönetimi	Dağıtım evresi gibi bir safhada riskleri minimize edecek bir yol var mı? Uygulamanın sürüm prosesinde geri alma seçeneği var mı?
Güvenlik Yönetimi	Uygulamanın dağıtımında organizasyonel güvenlik standartları ve gereksinimleri tam anlamı ile karşılanıyor mu?
Olay Yönetimi	Dağıtım, Olay Yönetimi sistemini mesele ve problemleri yayınlamak için kullanıyor mu? Dağıtım takımı üyelerinin Olay Yönetimi sistemine giriş yetkisi var mı? Eğer girebiliyorlarsa yeni kayıt yetkisi ve ilişkili önceki olayları görüntüleme yetkisi var mı?

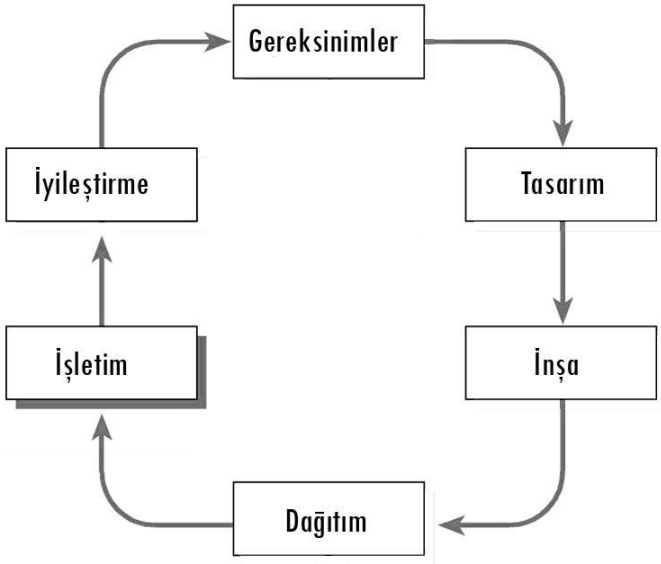
Problem Yönetimi	Bir problem meydana geldiğinde konu ile ilgili bir müdür atanmış mıve bu kişinin varlığından dağıtım takımı haberdar mı?
Kapasite Yönetimi	Dağıtım prosesinde Kapasite Yönetimi ilişkilendirilmiş mi, dağıtımla ilişkilendirilmiş kaynakların görüntülenmesi mümkün mü?
Erişilebilirlik Yönetimi	Dağıtım sırasında kurtarılmış ya da yedeklenmiş olanlar için uygulamanın kabiliyeti nasıl?
Hizmet Devamlılığı Yönetimi	Uygulamanın dağıtımı sırasında ya da sonrasında meydana gelebilecek bir felaketi takip eden iş kurtarma prosesi, herhangi bir değişiklik gerektirecek mi?
Hizmet Seviyesi Yönetimi	Hizmet Seviyesi Yönetimi uygulamanın dağıtımından haberdar mı? Dağıtım safhası için uygulama bir Hizmet Seviyesi Anlaşmasına sahip mi? Dağıtım sırasında organizasyonun Hizmet Seviyesi Anlaşma gereksinimlerini etkiliyor mu?
Finans Yönetimi	Yazılımın dağıtımı, toplam maliyetine dahil mi?

Tablo 0.5 Dağıtım fazının yönetilebilirlik kontrol listesi.

6.1.3.4.6 Dağıtım takımının organize edilmesi

Tüm uygulamanın dağıtımında da diğer fazlarda olduğu gibi yetkilendirme mevcuttur. Rollerin takım içindeki üyelere atanması ve bu rollere göre dağıtım planın tamamlanması gerekir.

6.1.3.5 İşletimi



Şekil 0.6 İşletimi safhasının uygulama hayat çevrimindeki yeri

Bu fazda bilgi işlem hizmetleri organizasyonu, hizmetin iş tarafında kullanılmasını sağlar. Hizmetin başarımı devamlı ve tekrar eden hizmet seviyelerine göre ölçülür.

6.1.3.5.1 Günü gününe hizmet seviyesinin sürdürülmesi için bakım aktiviteleri

En iyi bakımı yapılan sistemlerde dahi elbette hata meydana gelmesi, sunucunun durması ya da donması, ağ yapılarında sorunun oluşması mümkündür. Yazılım geliştirme problemleri, araçların başarısız olması, iş yöneticisinin ayrılması gibi sorunlara rağmen fonksiyonun işlemesi sağlanmalıdır.

Her uygulama kendi çalışmasını iç süreçleri ile sürdürebilmeyi bir miktar da olsa sağlar. Ama bu süreçler bazen günlük, haftalık, aylık ya da plansız olarak bozulur. Operasyon takımının bu bakımların akıbetini ve sürekliliğini kontrol edecek kimi personele ihtiyacı vardır.

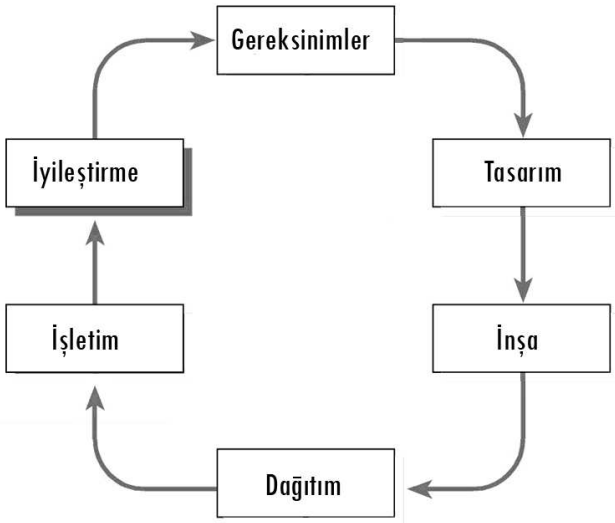
Örneğin;

- Günlük:
  - Çalışanların sayısını kontrol et
  - Günlük sorunları gözden geçir
  - Acil değişim ihtiyaçlarını gözden geçir
  - Veri tabanı tutarlılığını kontrol eden uygulamaları çalıştır
  - Veri tabanı hata kayıtlarını kontrol et

    - Hata ya da ağ ilişkili şartları başarım için görüntüle
    - İstemcilerin durumunu görüntüle
    - Sunucu bileşenleri ve hizmet durumunu görüntüle
    - Ana sunucular üzerindeki olay kayıtlarını görüntüle
    - Sistem başarımını görüntüle
    - Sistem dizinlerini görüntüle
    - Her sistem yönetim sunucusu üzerinde güvenli yedekleme yap
- Haftalık:
    - Değişim Yönetimi tahtasına dikkat et
    - Takım toplantıları düzenle
    - Veri tabanı performansını iyileştir
    - Sistem dizinlerini denetle
    - Yönetim raporu üret
    - Dosya sistemini yönet
- Aylık:
    - Aylık durum raporu derle
    - Takım üyeleri ile bire bir tamamla
    - Sistem sağlığını ve başarımını gözden geçir
    - Sistem performansını arttır
    - Sistem hesaplarını güvenli kıl
    - Sunucu fonksiyonlarına erişimi gözden geçir
    - Yedeklemelerin tutarlılığını onayla
- Plansız:
    - Takım ile beyin fırtınası estir
    - Geri bildirim anketini tamamla
    - Raporlanmış problemleri düzelt
    - Süreçlerin iyileştirilmesini gözden geçir
    - Güvenlik çakışmaları için durum mesajlarını gözden geçir
    - Güvenlik fonksiyonuna erişimi düzenle

Tablo 6.7 – Bir uygulama için bakım görevleri örneği

6.1.3.6 İyileştirme



Şekil 0.7 İyileştirme safhasının uygulama hayat çevrimindeki yeri

İyileştirme fazında, hizmet seviyesinin başarım ölçümleri, sonucu analiz ederek etkiler. Mümkün olan iyileştirmeler ve geliştirmeler gerektikçe tartışılır ve yapılır. İki ana önemli strateji bulunmaktadır bu fazda; hizmet seviyelerinin geliştirilmesi ve maliyetin düşürülmesi. Bu faz, hayat çeviriminde veya uygulamanın emekliliğinin ötelenmesinde esas kısımdır.

Bu bölüm aşağıdaki başlıkları içerir:

- Uygulama gözden geçirme süreçleri
  - Değiştirmeden
  - Modifiye etmek veya değiştirmek
  - Uygulamanın emekliliğe ayrılması
- İyileştirme evresi için yönetim kontrol listesi
- Organizasyonda iyileştirme takımının rolleri

6.1.3.6.1 Uygulama Gözden Geçirme Süreçleri

Uygulamanın ihtiyaçları sürekli karşılayabilmesi onun periyodik olarak incelenmesi ve yeniliklere göre düzenlenmesi ile mümkündür. Operasyonlar daima pazardaki rekabete göre sürekli olarak değişir ve bu değişimin operasyona destek olacak uygulamada da olması gerekmektedir. Sadece operasyonlar değil, uygulamanın çalıştığı alt yapı, iş tarafı veya teknik ihtiyaçlar da değişimin kaynaklarıdır. Kurumda değişimi gerektiren etmenler için sorular sorarak değişimin gerekliliğini, gerekiyor ise ihtiyaçların listesini ya da değişimleri uygulamanın

çözümleyip çözümleyemeyeceği göz önüne alınarak eğer çözümleyemiyorsa, uygulamanın emekliliğini cevaplamalı.

Bir değişimin olmaması hali en hızlı nihayetlendirilecek yanıttır. Bu durumda sadece son halin raporlanması ve değişimin gerekmediğinin vurgulanması yeterlidir.

Eğer bir değişime ihtiyaç olursa, bir önerinin yapılan değerlendirmeye karşılık sunulması gerekir. Uygulamadaki değişiklikler aşağıdaki başlıkları içermelidir:

- Değişimin gerekliliği
- İş tarafının ihtiyaçları ve mazeretleri
- Değişimden kimin sorumlu olacağı
- Değişimin tamamlanması için gerekli çabanın miktarı
- Değişim için ön istekler
- Değişim ihtiyaçları tamamlandığında yapılacaklar
- Değişim için başarı ölçütü

Uygulamanın artık işe yaramadığı anlaşılınca emekliliğine karar verilir. Bu kez kıymetli olan hâlâ yaşayan veya sistemde bulunan bu uygulamanın hizmetten kaldırılmasıdır. Eğer kaldırılması yeterince karmaşık ise, yazılım takımından yazılımı kaldırmak için bir araç geliştirmeleri istenir.

İyileştirme evresinde tüm ITIL Hizmet Yönetimi fonksiyonları düşünülmelidir. Bu evrede bir kontrol listesi ile sürecin devam ettirilmesi ve doğru analiz ve kararlarının alınması elzemdir.

Hizmet Yönetimi fonksiyonları	İyileştirme fazı yönetim kontrol listesi örnekleri
Konfigürasyon Yönetimi	İyileştirme evresinde uygulama gözden geçirildikçe sonuçları için Değişim Yönetimi Veri Tabanı kullanılıyor mu? Konfigürasyon yönetimi çalışanı, iyileştirme prosesi ile ilgili, envanterin güncellenmesi ve kullanımı ile tavsiye veriyor mu?
Değişim Yönetimi	Modifikasyonlar bu fazda tanımlandı mı, takım Değişim Yönetimi sistemini değişiklikleri koordine etmek için kullanıyor mu? İyileştirme takımı, Değişim Yönetimi proseslerini anlıyor mu?

Sürüm Yönetimi	İyileştirme takımının üyeleri, Sürüm proseslerini anlıyor ve dağıtım planlaması için kullanıyorlar mı?
Olay Yönetimi	İyileştirme takımının üyeleri, Olay Yönetimi sistemine erişime sahip mi, böylece olayları kaydedip iyileştirmeleri adresleyebiliyorlar mı?
Problem Yönetimi	İyileştirme prosesleri, düzeltme prosesleri içinde Problem Yönetimi tarafından bilgilendiriliyor mu?
Kapasite Yönetimi	Kapasite yönetimi, iyileştirme proseslerine doğru geri bildirimde bulunuyor mu?
Erişilebilirlik Yönetimi	Uygulamanın yedeklenmesi ve kurtarılması yetenekleri için bir geliştirmeye ihtiyaç var mı?
Hizmet Devamlılığı Yönetimi	İş tarafının ihtiyaçlarını karşılamak için iş tarafı kurtarma proseslerinde iyileştirme gerekiyor mu? Gerekiyorsa nelerdir?
Hizmet Seviyesi Yönetimi	Hizmet seviyesi yönetim bilgisi, iyileştirme prosesleri içinde yer alıyor mu?
Finansal Yönetim	Finansal bilgi düzeltme ya da ayarlama proseslerinde yer alıyor mu?

Tablo 0.6 İyileştirme fazı yönetimi için kontrol listesi

6.1.4 Hayat Çevrimini Bir Uçtan Diğer Uca Yaymak.

Uygulama hayat çevirimi hakkında hatırlanması gereken bir başka şeyde, bu çevirimin dolambaçlı olduğu, aynı uygulamanın hayat çevriminde farklı fazlarında ve aynı zamanda olabileceğidir. Örneğin; uygulamanın bir sonraki sürümü tasarlanmaya başladığında ve hali hazırdaki versiyonu yüklendiğinde, önceki sürüm hâlâ organizasyon içerisinde hala işletiliyor olabilir. Bu açıkça güçlü bir sürüm ve konfigürasyon kontrolü gerektiren durumdur.

Fazların yapısına özgü olarak diğer fazlardan daha fazla zaman alan veya daha fazla önem arz eden durumlar oluşturabilir. Örneğin; geliştirme fazı çoğu zaman dağıtım fazından daha fazla zaman alabiliyorken, tasarım fazı da geliştirme fazına göre daha fazla önem arz edebilir. Elbette zaman ve yazılımın durumuna göre

değişiklik arz edebilir. Uygulamanın tasarım kısmı çok zaman almasına rağmen yazılım kısmı, tasarım kısmının aldığı zamana karşın çok daha az sürede tamamlanabilir. Bu durumlar bazı uygulamaların ilk zamanları için bu şekilde olabileceği gibi, yazılımın dağıtımından sonra iyileştirme kademesinde operasyonel istekleri yazılıma yansıtmak için yeni tasarımlar gerekebilir. İşte yeni tasarımların yazılımda olması gereken hali, çoğu zamanda kodlamadaki zamandan daha kısa sürebilir. Nitekim yazılımın sürümü gerçekleştirilmişti ama iyileştirmenin ele alınıp, yeni tasarımın planlanması yazılımı destekleyen programcı grup tarafından eski ve konfigüre edilmiş halinden elbette tekrar haberdar olunarak, tekrar sisteme yeni özelliklerin eklenmesi programcılar tarafında daha fazla süre geliştirmeye ayırmalarına neden olabilir. Yine de tüm fazların tek tek geçilmesi ve her birinin peşi sıra işletilmesi kritik bir öneme haizdir. Bu hayat çevriminin doğasıdır ve en az bir kez tamamlanmalıdır. Tamamlanmayan hayat çevriminin neticesinde oluşabilecek ya da oluşmuş yazılım başarısız olur.

İyi haberleşme, uygulamanın hayat çevirimi boyunca bir anahtar gibi çalışır. Yazılımın organizasyon içerisinde geliştirilme nedenlerine tekrar bakarak iç haberleşme ve koordinasyonun önemini bir kez daha anlamış olabiliriz.

Bilgi işlem organizasyonunda işin ilerlemesi konusunda edinilmesi gereken bilginin iş tarafının daha iyi ve daha hızlı hareket edebilmesini, müşteriye ulaşırken ve ihtiyaçlarını karşılarken daha esnek ve daha güvenilir yaklaşmasını sağlayabileceğini, bu yüzden bilgi işlemin iş operasyonun ayrılmaz bir parçası olduğunu vurgulamak, bu noktadan sonra yapacağı hamleleri anlayabilmek için önemlidir. İş tarafının daha aktif ya da daha verimli olabilmesini sağlayabilecek teknolojik önerileri sunabilmesi ve bu teknolojinin yazılım ya da donanım olarak iş tarafının hizmetine sunulabilmesi için bir hayat çevrimini başlatması gerekmektedir. Öncelikle iş tarafının, bilgi işlem tarafından iyileştirilebilecek bir sürecinin başlangıç fikrinin tartışılması ve olgunlaştırılarak fikir aşamasından faaliyet noktasına taşınması için atılan ilk adımı organizasyonda karar mercileri ve uygulayıcılar, kullanıcılar dâhil olmak üzere her birimin kabul etmesi şarttır.

Ortak bir amaç oluşturulup yazılımın gerçek ihtiyacı fark edildiğinde, projeye katkısı olabilecek tüm birimlerin proje yöneticisinin oluşturacağı takvime uygun periyotlarla gereksinim bildirimlerini ve işleyişi taraflarında olabileceği hali ile aktarmaları gerekmektedir. Bu safha yukarıda belirtildiği gibi ihtiyaçlar safhasıdır ve beklentileri ortaya koymak ve projenin boyutlarını çizebilmek için önemlidir. Tüm

operasyona dâhil olan dolaylı, dolaysız her kullanıcının istek ve önerilerinin toparlanması projenin kaynak kullanımını belirlemede, maliyet analizinde ilk bilgilerin ortaya çıkmasını sağlayacaktır.

7 ÖRNEK UYGULAMA

Örneğin; proje bir intranet eksiğinin hissedilmesi üzerine düşünülmüş ve bu ihtiyacın etrafında şirketin tüm birimleri taleplerini oluşturmuş olabilir. Bu tip bir proje şirkette her birimi doğrudan etkileyecektir. İş tarafını ciddi ölçüde etkileyen bu projeye iç haberleşmelerden doküman yönetiminin entegrasyonuna, satış görüşmelerinin ve sonuçlarının takibinden, takımların kendi projelerini izleyip, görevlerin güncellemelerine kadar önemli olabilir. Yine aynı proje üzerinden yönetici pozisyonundaki kişilerin iş planlamasından tutun da projedeki personellerin performans analizine değin, raporlamalar alabilmelerini sağlayabilecek bir havuz oluşturmak mümkündür. İntranet üzerinde projelerdeki gelişmeler olabileceği gibi duyurular, şirket içi haberler, atamalar, görev değişiklikleri hatta yemekhanede günün yemeğine değin bilgiler bulunabilir. Bu durumda görülüyor ki örneğimizdeki proje ciddi bir kapsam ve etki alanına sahiptir ve tüm organizasyon personellerini etkilemektedir. Bu da büyük bir uzlaşı kümesi oluşturmak ve gereksinimleri karşılamak için ciddi bir kaynak planlaması için iyi hareket planına sahip olmak gerekir.

Gereksinimlerin belirlenmesi aşamasından sonraki safha tasarım fazıdır. Tasarım safhasında yazılımdan beklentilerin toparlandığı gereksinim safhasından elde edilenler doğru bilgilendirme ile bu safhaya aktarılarak uygun teknoloji donanım altyapısı ve yazılım altyapısı seçimi yapılır. Projenin işleyişini, inşa ve sonraki iyileştirme dönemlerinde, yeni katılan yazılımcılardan, eğitim dokümanlarının oluşturulmasına kadar UML ile dokümante etmek gerekir. Bu doküman zaman içinde yazılımdaki tüm değişimlerle tekrar düzenlenmeli ve güncel halini korumalı. Tasarım aşamasında gereksinimleri yazılımda olması gereken halini düşünerek ve oluşan halinin sürekli olarak iyileştirilmesi sağlanarak, UML diyagramları güncellenir ve tasarım aşaması nihayetlendirilir. Tasarım aşaması yazılımın sadece arayüzü değil aynı zamanda sınıf diyagramları, nesne diyagramları, kullanım vakalarının diyagramları da hazırlanmalıdır. Yazılım için hazırlanmış UML diyagramları her zaman incelenerek yazılımın çalışma metodolojisi ve fonksiyonları tekrar anlaşılabilir olur.

Hali hazırdaki örneğimizde organizasyon içerisinde dâhili ağ donanım altyapısı olmazsa olmaz gereksinimler için kullanılacaktır. Bu ağ yapısında çalışacak

tüm kullanıcıların ya da istenilen yetkideki kullanıcıları erişimine müsait bir sunucunun bulunması gerekmektedir. Projeyi barındıracak sunucunun üzerinde koşacak yazılımlar ve bu yazılımlara uygun çalışacak uygulamamız bulunacaktır. Tam bu noktada kritik bir kararın daha eşiğindedir bilgi işlem organizasyonu. Talep edilen uygulamanın teknolojisi, hem beklenen sistem yükünü, hem geliştirme hızını, hem geliştiricilerimizin bilgisi dâhilinde olan bir kodlamayı, hem de bakım maliyetlerini göz önüne alarak karar verilmesi gereken bir konudur. Yazılım geliştirme takımımızın Java programlama diline hâkim olduğunu düşünelim. Java piyasada maliyeti yüksek yazılımcıların bildiği programlama dilidir. Bu yazılımcılara sahip olan organizasyonumuz adam saat ücreti bu iş için maliyeti belirleyecektir. Ama aynı seviyede geliştiricilere sahip bir başka şirkete yazılımın çekirdeğinin yaptırılması yazılım maliyetini yukarı çekecektir. Çekirdek kısmını eğer Java ile yazdırmak istersek, bulmamız gereken firmanın bizim için çıkaracağı fiyatlandırma en az bizim personellerimizin(kalitesi ile doğru orantılı olarak) maliyetine kâr marjı eklenmiş olan rakama ulaşacaktır. Firmanın istediğimiz şartlarda çalışması ve açık kaynak olarak yazılımı bize vermeleri operasyonel olarak yüksek bir rakama ulaşacaktır. Bu duruma alternatif olarak piyasada yaygın olarak bilinen bir başka güçlü programlama dili olan Microsoft C# dilidir. C# programlama dili istihdam ettiğimiz Java bilen personelin zaman içerisinde uyum sağlaması kolay olacak yazılım dilidir. Bu da bize sisteme daha sonra yazılım takımımızın entegrasyonunu kolaylaştıracak bir artı olur.

Microsoft C# yazılımını bilen yazılımcılara sahip bir başka firmaya çekirdek kısmını, Java ile kodlama yapan bir firmaya yazdırmaktan daha ucuza gelecektir, bu da maliyetlerin bu kez yazılımcı kadromuzda C# bilen programcı istihdamı yerine bu teknolojik altyapının Java diline yakınlığı nedeniyle teknolojik bilginin çalışanlarımıza transferi, maliyetleri tekrar değiştiren bir etmendir. İki yöntemle C# ile yazılmış çekirdek yazılımını bünyemize dâhil edebiliriz; birincisi, yeni bir istihdam oluştururuz. Bu istihdam C# bilen yazılımcıları kadromuza katmak olarak yapılabilir. İkincisi ise, kendi kadromuzda bulunan ve projenin devamı için yeterli sayının yazılım mimarı ve teknik liderin planlamasına uygun olarak yazılımcılarımızın eğitimi olarak gerçekleşmesidir.

Kaynak planlaması daha öncede anlatıldığı üzere birçok stratejik seçenekten bir veya bir kaçının bir arada kullanılması ile mümkündür. Yine örneğimiz üzerinden giderek kaynak planlaması hakkında bir kaç şey söyleyebiliriz. Projenin ihtiyaçlar

safhasında elde ettiğimiz veriler ile ne büyüklükte olacağını görebilmemiz ile bu yazılımı iç, dış, ortaklı, iş ortaklı stratejik planlamalardan hangisi ile yapacağımıza karar verebiliriz. Projenin organizasyon içinde hali hazırdaki zihniyet ve yaklaşımlardan dolayı çok uzun sürelerde kullanılacağı aşikârdır. Neredeyse her organizasyon altyapı olarak bir iç ağa sahiptir ve bu iç ağı en verimli şekilde kullanabilmenin planlarını yapmaktadır. Değerli olan, var olan fiziksel kaynakların tam anlamı ile verimli kullanımı ve iş tarafına sağlanabilecek en yüksek faydayı verebilmektir. İşte bu maksatla bakıldığında kullanıcıların organizasyonda masaüstü, dizüstü, cep bilgisayarlarını, şahsi dijital yardımcılarını(PDA, Personal Digital Assistant) gibi ağ üzerinden bilgi alıp verebilen cihazlarını ağa kablolu ya da kablosuz ağlar ile dâhil ettiklerinde işleri ile alakalı tüm bilgilerini transfer edebilme düşüncesindedirler. Bu altyapı yatırımında gerçek faydayı ancak üzerinde yapılan işin hacmini arttırarak sağlayabiliriz. İş hacminin arttırılması adına intranet proje örneğimizin maliyetlendirilmesi, projenin hayata hangi kaynakların kullanımı ile daha düşük maliyet ve yüksek hızda yapılabileceği, işletimi sırasında yapılacak iyileştirmelerin ve uygulamanın bakım maliyetlerinin kâr ve zararlarının ortaya konması gerekmektedir. Yeni projeler, yazılım geliştirme ekibine sahip organizasyonlarda daima heyecan ile karşılanır ve iç kaynakların kullanımı ile yapılması yönünde eğilim sergilenir. Oysa gerçek anlamda kaynak yönetimi isteklerin, heveslerin yerine akılcı politikaların dikkate alınması ile gerçekleşir. İşte tam bu noktada projenin süresi, tasarım ve iyileştirmenin sıklığı, bakımı dikkatle hesaba katılmalıdır. Örneğimizdeki projenin uzun soluklu bir kullanım süresine sahip olacağı için aşikârdır. Bu süre zarfında değişimi ve bakımını bilgi işlem tarafından yüksek sıklıkla karşılaşılacağı, isteklerin zaman için değişeceği, kimi zaman yeni eklentilerle, kimi zaman da gereksinimlerin ortadan kalkacağı, çoğunlukla yeni isteklerin oluşacağı düşünülünce, iç kaynak kullanımı akla yatkın bir çözüm olarak karşımıza gelmektedir. Projenin beklenen teslim süresi ise, iç dinamiklerin heyecanını soldurmadan süratle ve projenin güvenilir yapısını temin ederek sürmelidir. Bu durumda iç kaynakların bu projeyi istenilen zamanda faaliyete geçirebilmesi için ihtiyaç duyduğu süre dikkate alınmalıdır. Eğer iç kaynaklarla tamamlanması için gereken süre (teknolojinin bilinmesi, yazılımcı sayısı ve iç entegrasyon bu süreyi doğrudan etkileyen etmenlerden sadece bir kısmıdır) dış kaynaklarla tamamlanacak süreden az ise ve eldeki diğer projelerin önem ve bu projelerdeki insan kaynağının dağılımında aşılamayacak bir problem teşkil

etmeyecek ise, kabul edilecek kaynak kullanımı dahili kaynaklar olacaktır. Bilgi işlemin hali hazırda sürdürmesi gereken projelerin yanı sıra intranet projesinin istenilen zamanlarda yetişmemesi durumunda, dış kaynak kullanımının maliyeti göz önüne alınabilir. Bu maliyet bir kaç yolla düşürülebilir. Projede ihtiyaçlar fazının bölümlenmesi ve bu bölümlerden dış kaynaktaki tecrübenin bilgi işlem organizasyonuna aktarılması için, çekirdek kısmının yaptırılması ve bu çekirdeğin üzerine organizasyon bünyesinde başlanabilecek zamana kadar, dış kaynak kullanılarak zamanın değerlendirilmesi ihtimal dâhilinde olabilir. Bu sayede sürekli itibarda sürecek bu projenin şirket bünyesinde devam edilebilmesi için hazır bir nüveye sahip olunabilecektir. Önemli olan şey ise dış kaynaktan alınacak bu bilginin açık kaynak olmasıdır. Projenin tümü dış kaynak tarafından yapılmayacağı için hem maliyet olarak kazançlı olunacak, hem de projenin çekirdeğinin dış kaynaktaki tecrübe ve bilgi birikimleri ile yapılacağı için bu bilgi birikimini organizasyona kazandırılacaktır. Organizasyon dış kaynak tarafında hazırlanan çekirdek yazılım üzerine bina edeceği diğer fonksiyonlar için, yeterli bilgi seviyesinde yazılım geliştiricileri projeye dâhil ederek ya da yazılım takımında bu proje ile ilgilenmesini istediği yazılımcılara C# eğitimi vererek bilgi işlem bünyesinde projenin geliştirme sürecine devam edilmiş olunur. Bu yazılımcıların projenin tasarım aşamasında, teknik lider ve yazılım mimarları tarafından hazırlanmış tasarım kısmında belirtilmiş modülleri ayağa kaldırmaları ve dağıtım safhasına götürmeleri elbette tüm projenin dış kaynak kullanımının maliyetinden daha düşük ve sistemin bakım ve iyileştirmelerinin hızlı, güvenilir ve esnek olmasını sağlayacaktır.

Projenin bilgi işlem organizasyonuna harici firmadan geçmesi ile birlikte, yazılımın inşa fazı devam edecektir. UML diyagramlarına bakarak yazılım ekibi çekirdek yazılımın sistematiğini, uygulama içindeki metot ve prosedürlerin sebeplerini anlayabilecek ve çok kısa bir sürede, kalınan yerden inşa sürecine devam edebileceklerdir. Bu faz yazılımın ilk kez inşa edilmesi nedeniyle uzun süren fazlarından birisidir. Ama tasarım fazının ihtiyaçlar fazında elde edilen bilgileri ne kadar iyi derlediği, bu fazın tamamlanma süresini tayin eden en önemli noktadır. Elbette ki her fazın ehemmiyeti yüksektir ve elbette ki tüm bu fazların arasındaki geçişlerde bilginin aktarımı ayrı bir önem taşımaktadır. Son hali ile bilginin transferi projenin bundan sonra geliştiricilerini, kendi yazılım takımımıza aktarımı sağlandıktan sonra, yazılımın ilk sürümü teste girmek üzere hazırlanır. İnşa safhasının son noktası olan testlere girmek üzere hazırlanmış yazılım, öncelikle

geliştiricilerin yazılımı geliştirirken yaptıkları testlerden geçmiş, sonrasında sürüm öncesi profesyonel testçiler tarafından test edilmek üzere hazır hale gelmiştir.

Sürüm öncesi test safhasında esas olan, kullanıcı gözü ile yazılımı denemelerden geçirebilmektir. Bu konuda varsa yazılım geliştirme takımından yoksa ve özellikle harici firmalardan denemeleri yapabilecek personellerden faydalanmak tavsiye edilir. Yazılım takımında test edecek personel, kodlama sırasında yazılımın ara sürümlerinde denemeler gerçekleştirdiği için bir süre sonra kullanıcının bakış açısını yitirmiş olacaktır. Bu durumda yazılımın gerçek anlamda test edilmesi mümkün olmayacaktır. İşte bu sebeple yazılıma ilk kez bakan göz olabilmesi açısından profesyonel ama harici bir kişinin testi yapması gerekmektedir. Bu testler yazılımın sınırlarını zorlayıcı ve kullanıcı hatalarını ve onların bakışına uygun olacak testlerdir. Örneğin; ürün adı alanına 8000 karakterlik bir ürün ismi yazıp, kaydetmeye çalışması ya da aşağı açılır kutunun düşünülen boyutunun 200 piksel genişliğinde içeriğini değiştirerek kullanılırlığını test etmesi gibi.

Uygulamanın operasyonel Yüksek kalitede bilgilinin uygulamanın hayat çevriminde bir fazdan diğer faza geçişteki önemi kritiktir. Yine organizasyonun Uygulama Yönetimi hayat çevriminin kalitesini sürekli itibarda gözlemlemesi ayrıca önemlidir.

Hayat çevrimindeki değişiklikler, örneğin bilginin organizasyon içinde farklı fazlara geçişi, kaliteyi etkileyecektir. Uygulama Yönetimi hayat çevirimindeki her fazın karakteristiğini anlamak, toplam kalitenin iyileştirilmesinde kritik önemdedir. Bir fazda kullanılan yöntemler ve araçlar diğerlerini tekrar iyileştirirken etkileyecektir.

Uygulama, emeklilik kararına kadar bu hayat çevirimini yaşayacaktır.

Çalıştığım şirket içinde ITIL Uygulama Yönetimine uygun olarak bitirme projemi temel aldığım bir uygulamayı ve süreçlerini örnek olarak göstereyim.

Uygulama talepleri gereksinimlerden doğar. Bu ihtiyaçlar ne derecede bir uygulamaya gerek olduğunu ancak iyi bir ön analiz ile netleştirebiliriz. Uygulama Yönetimi süreçlerini uygulayabileceğimi düşündüğüm uygulama, Sosyal Güvenlik Kurumu'nun e-devlet projesi altında geliştirdiği MEDULA(Medikal ULAk) sisteminden doğdu. MEDULA'nın amacı, sigorta kapsamındaki herkesin sağlık harcamalarını WEB Hizmetleri üzerinden klinik, hastane, polikliniklerden alıp ödeme yapılıp yapılamayacağını ve faturalama hizmetleri çevrimiçi olarak muhataba bildirmektir.

Çalıştığım firma, diyaliz hizmetlerini 52 klinik üzerinden, tıbbi ilaç ve araçların ithalat, üretim ve satışını iki anonim şirket üzerinden sağlamaktadır. 4500'ün üzerinde hastası olan firmada aylık ortalama 60.000 seans yaparken, pazarın %60'ına hâkim durumdadır. Böylesi bir çapta olan şirketin faturalamadaki bu değişimden (bugünün rakamları ile aylık 8.000.000 YTL civarında ciro) eğer zamanında bir uyum süreci geçirmezse ne derecede etkileneceği aşikârdır. Görülüyor ki gereksinimin doğduğu konu, doğrudan diyaliz hizmetlerinin gerçekleştirildiği şirketleri ve dolaylı olarak da bu şirketlere ürün satan diğer tedarikçi firmaları etkilemektedir.

Yukarıda ki ihtiyacın ön analizi ile ihtiyacın karşılanması için bir uygulama ile gereksinimin karşılanması sonucuna kolayca varılmaktadır. İhtiyacın karşılanması için öncelikle şirket içinde hazırlanması ya da hazırlatılması talep edilen uygulamanın hudutlarını çizebilmek gerekmektedir. Öncelikle proje tanımı başlangıçta yapıldı(bu tanım ilerleyen safhalarda genişleyebileceği gibi daralabilen bir çerçevede oldu). Proje tanımına uygun olarak konu ile ilgili şirket bölümlerinin taleplerini alabilmek için bir dizi toplantı düzenlenmesine karar verildi. Sıra ile konuyu operasyon, muhasebe, medikal destek bölümleri ve uygulamanın finansal yönetiminin çerçevesinin çizilebilmesi için yönetim ile görüşüldü. Bu görüşmelerin neticesinde projenin planı ortaya çıkarıldı. Amaç, faaliyet alanı, hedefler, kabuller, kısıtlar, riskler, dağıtım, takvim, bütçe planı, planın evrimi gibi başlıkları içeren ve detaylandıran bir proje planı hazırlandı.

Proje için ayrılabilecek bütçe, dâhili bilginin oluşturulması ve gizliliği, ilerleyen zamanda gelişimi ve bu gelişimin maliyeti, muhasebe bölümünde kullanılan SAP uygulaması ile birlikte çalışması gerekliliği, şirket içi uygulamalara veri ihraç edeceği yanı sıra çalışılabilecek düzeydeki yazılım firmalardan alınan teklifler de hesaba katılarak, yazılımın şirket içi yazılım geliştirme kaynakları kullanılarak üretilmesine karar verildi.

Proje planında uygulamanın, bilginin oluşturulması takımının yanı sıra yazılım geliştirme ekibi ve ekibin iç koordinasyonu oluşturularak yazıldı. Ön analiz yapılmış ve plan hazırlanmıştı. İlk analizin daha detaylandırılması ve diyagramların oluşturularak yazılımın geliştirme takviminin oluşturulması sağlandı.

Yazılım takımının kısıtlı sayısından dolayı analiz, kodlama, test, eğitim başlıkları farklı takımlara değil, yazılım takımı üyelerine dağıtıldı. Analiz, kodlama gibi konularda tüm takım başlangıçta çalışmaya katıldı. Yazılımı öncelikle

dağıtımının hızlı ve minimum adam saat ile sağlayabilmek, hayat çevriminde ki değişimleri hızlıca hizmete sunabilmek için WEB tabanlı olabileceği teknoloji üzerine inşa kararı verildi. Kodlama da Microsofot.NET teknolojilerinin, veritabanı olarak da Microsoft SQL 2005 kullanılması yönünde karar alındı. İnternet üzerinden çalışması planlanan uygulamanın, hem hat kullanımında en verimli seviyeye ulaşmayı hem de kullanıcı rahatlığının sağlanması için AJAX teknolojisi kullanılmasına karar verildi. Bu teknolojiler üzerinde çalışacak projenin yazılım analizi ve diyagramlarının hazırlanması için proje planında temas edilecek ya da rehberliğine karar verilmiş kişilerle toplantılar yapıldı ve bu toplantılarda ki görüşmeler kayıt altına alındı. Sektörde konu ile ilgili yapılan yazılımların ara yüzleri ve fonksiyonellikleri de daima incelenerek şirketin oluşturduğu iç bilginin gizliliği sağlanarak tüm bu görüşmelerin neticesinde yazılımın UML diyagramları hazırlanarak kodlama aşamasına geçildi. Kullanım, sınıf, nesne diyagramları, veritabanının yapısı yanı sıra uygulama formlarının ara yüzleri tasarlandı.

Kodlama yani inşa safhası süresince hali hazırdaki analizin gözden geçirilmesine devam edildi. Bu süreçler döngü içerisinde ve sürekli kontrol altında tutularak geriye dönüşleri en aza indirmek, hataları ya da geleceğe dönük yapılabilecek iyileştirmeleri şimdiden sağlayabilmek sürekli kontrol edilip düzeltildi.

Kodlama süresince haftalık toplantılar gerek toplantı salonları kullanılarak, gerek ayakta 15 er dakika içinde tamamlanmak üzere gerekse de internet üzerinden yapılarak durumumuz ve hedeflere ulaşırken yaşadıklarımızı değerlendirdik. İlk sürüm her uygulama için elzem ve rehberdir. Haftalık toplantılarımız en az bir ay içinde büyük hedeflerimizden birini bitirmek üzere kurgulanmaktaydı. Ama esas olan ilk sürümün çıkmasıdır ve hızlıca testlerden geçirilip kullanıcılara gösterilmeliydi. Bunu sağladığımızda projenin gidişinin ne derece doğru olduğunu görebilmiş ve küçük değişiklikler ile diğer sürümleri daha iyi hale getirebildik.

Uygulama sürümleri son halini alınca eğitiminin verilmesi için ülke üzerindeki klinikler belirli noktalarda bir araya getirildi. Kullanıma başlayan uygulamamız, veritabanında gittikçe değeri artan verileri topluyor, şirket içinde ölçülmesi zor sonuçlara çok kısa sürede kesin doğruluklarla ulaşılabilmeyi sağlıyordu. Tüm bu anlatılan ITIL Uygulama Yönetimi süreçlerini tezimi yazdığım bu zamanda dahi yazığımız bu yazılıma uyguluyor, sonuçlarını her yeni sürümde daha verimli hale getiriyoruz.

8 SONUÇ

Tüm yukarıda anlatılanlar ışığında ITIL, süreçlerin prosedürler ile bir düzen ve ilişkiler dizisinin belirtilmesine, bu süreçlerin sıralı ve sürekli olarak iyileştirilmesini sağlayan en iyi uygulamalar bütünüdür. Tezimde inceleme konum olan Uygulama Yönetimi kısmı içinde bir yazılım, ihtiyacının büyük ya da küçük şirket ya da organizasyonlar içinde doğru tespitinin yapılması ve eldeki imkânların karşılayamayacağı durumlarda yeni bir başlangıcın yapılması ve nihayetlendirilmesine kadar süren yolculuğu ITIL süreçlerine göre anlatılmıştır. Bulunduğum tüm yazılım projelerinde, ITIL süreçlerinin üzerinde durduğu proseslerin, iyi idare edilmediği sürece problemlere neden olduğunu ve bu problemlerinde iyi yönetilmemesi sonucunda projenin doğmadan ölümü ile sonuçlandığına şahit oldum.

Bir uygulamanın bir hayat çeviriminin olduğunu, bu çevirimin her evresinin doğru işletilmesi gerekliliği ne kadar aşikârsa, bu safhaların iyi idare edilmesinin ancak eski tecrübelerden elde edilen bilgilerle başarılabileceği de o kadar açıktır. Bu yüzden uygulamanın doğru karar verilmiş bir çatı altında(ki ITIL bu çatılar içinde en yaygın ve başarısını kanıtlamış olanıdır) hayat çevrimini tamamlaması en doğru karar olacaktır.

Yukarıda, hem iş tecrübem hem de tez çalışmam sırasında okuduğum kaynakları göz önüne alarak söyleyebilirim ki; ITIL gibi bir çatı, işletme içerisinde kalite artırımını ve organizasyon yapısının daha iyileştirilmesini desteklemektedir. Gerçek hayat tecrübeleriyle, yapılması gerekenlerin düzenli ifade edilmesi ve kullanımımıza açık bir kaynak haline getirilmesini sağlayan bilgi teknolojileri uzmanları, ITIL çatısı ile şirketlerin bilgi teknolojilerine bakışını değiştirerek, kâr odaklı operasyonlarında eski ama çok güçlü bir silah olarak kullanmalarını sağlamak için şirketlere ideal bakış açısı katar.

KAYNAKÇA

1. Vail, P. (1991). "*Management as a Performing Art; New Ideas for a World of Chaotic Change"*.
2. Jossey-Bass. Weick, K. E. (1995). "Organisational redesing as improvisation".
3. H. A. Glick, (1991).*Organisational Change and Redesing: Ideas and Insights for improving performance"*.
4. Office of Government Commerce. (2007). "*The Official Introduction to the ITIL Service Lifecycle"*.