5.16. Sorular

5.1) Proje plan türleri nelerdir?

5.2) Plan türlerinin ortak özellikleri nelerdir?

5.3) İşlerin bölünmesinde takip edilecek yolları açıklayınız.

5.4) Yazılım projelerinin planlanmasında karşılaşılan zorluklar nelerdir? Bunların çözüm yolları ne olabilir?

5.5) Projenin ilk aşamalarında planın çok ayrıntılı yapılmasının olumlu ve olumsuz yönlerini açıklayınız.

5.6) İşlerin bölünmesi ile yazılımın mimari yapısı arasındaki ilişki nedir? Yazılım mimarisi doğrudan planda kullanılabilir mi? Açıklayınız.

5.7) *Kritik yol* nedir? Açıklayınız.

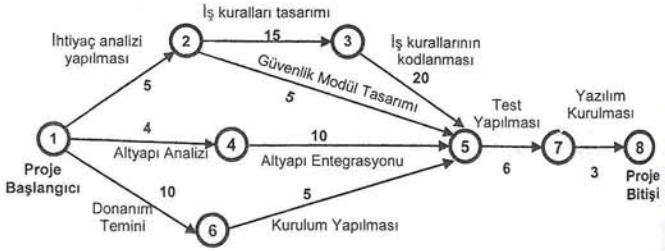
5.8) Görevler arası ilişki türleri örneklerle açıklayınız.

5.9) *Kaynak dengeleme* nedir? Nasıl gerçekleştirilir?

5.10) *Ağ* ve *Gantt Şemalarından* hangisi daha esnektir? Sebebini açıklayınız.

5.11) Proje zaman planının hedefi nedir? Açıklayınız.

5.12) Ağ şeması verilen plan için kritik yol hesabını yapınız; bkz. Şekil 5.13.



Şekil 5.12. Örnek Proje Planı

"*Altyapı Analizi*", "*Güvenlik Modül Tasarımı*" ve "*Kurulum Yapılması*" görevleri için Toplam ve özgür bolluk süresi hesabını yapınız. En erken bitiş ve başlangıç tarihlerini belirleyiniz.

5.13) Kaynak takvimlerinin belirlenmesinde hatalar türlerini açıklayınız.

5.14) Yazılım proje maliyetini oluşturan kalemleri açıklayınız.

5.15) Planlama sürecinde ortak katılım ve onayın yeri ve hedefini açıklayınız.

5.16) *Kaynak seviyelendirme* nedir? Projenin toplam süresine etkisini açıklayınız.

5.17) Proje yöneticisi görevlerdeki aşırı yüklemeleri düzenlerken öncelikle *özgür bolluk süresini* mi yoksa *toplam bolluk süresi* mi sıfırdan büyük olan görevleri kaydırır? Sebebini açıklayınız.

Bölüm 6

Plan Kalitesi

Bu bölümde öğrenilecek kavramlar:

Plan kalite faktörleri	*Modülerlik ve hiyerarşi*
Görev takibinde 0-100 kuralı	*Kapalı çevrim planlama*
Planlamada modülerlik ve süreç	*Planlama hataları*

Kalite[1], bir ürünün bilinen en iyi özellikleri bünyesinde taşıması durumudur. Bir yazılımın kalitesi geliştirme sürecinin kalitesine, geliştirme sürecinin kalitesi de planlama sürecinin kalitesine bağlıdır. Oluşturulan plan sonrasında yapılacak işlerin kalitesi için belirleyicidir. Elbette planlama yanında oluşturulan ekip ve ekibe sunulan imkânlar ölçüsünde kaliteli bir yazılım ortaya çıkar.

Planın yüksek kaliteye sahip olması planın anlaşılmasını kolaylaştırır, izlenebilme ve gerçekleştirilebilmesine imkân verir. Kişi planı kolayca anlayabilir çünkü kendisine atanan görevler tutarlı ve açıktır. Diğer ekip üyeleriyle birlikte çalışma şekli de açıkça tanımlanmıştır. Plan kolayca izlenebilir çünkü kişinin işi bitirme şartları ve iş çıktıları açıkça tanımlıdır. Yapılacakların tümü plana dâhildir. Plan gerçekleştirilebilir, çünkü her görev gerçekleştirilebilir çıktılara sahiptir ve süresi makuldür.

Planın kalitesi tüm paydaşların planlama sürecine dâhil olması, işlerin tek kişinin yapabileceği ve anlayabileceği düzeye kadar bölümlenmesi, yapılacak tüm işlerin plana dâhil edildiğinin kontrolü gibi yöntemlerle güvenceye alınır. Bunları gerçekleştirmek için proje alt projelere bölünmeli, böylece çok uzun binlerce adımlık planlar yerine kısa ve basit planlar kullanılmalıdır. Bu her açıdan kolaylık sağlar.

Plan kalite faktörleri, bu bölümde bir bütün olarak incelenmiştir. Planlamada ortak doğrular olduğu gibi maalesef hatalar da birbirine benzemektedir. Bölüm sonunda konuyu farklı bir açıdan incelemek amacıyla genel hata türlerinden de bahsedilmiştir.

---

[1] http://tdkterim.gov.tr/bts/ (Erişim: Şubat, 2012)

6.1. Görevlerle İlgili Kalite Faktörleri

Plandaki görevleri, anlama ve izlenmeyi kolaylaştıran özellikler planın kalitesini de arttırır. Bir *görev* gerçekleştirilebilir, bitiş noktası ve çıktıları net tanımlı, süresi makul ve kısa, mesai içini kapsayacak ve tek kişi tarafından yapılacak şekilde planlanmış olmalıdır. Benzer şekilde projenin kontrol noktaları olan kilometre taşlarının da tamamlanma kriter ve çıktıları net olmalıdır. Kontrol süresi kısa tutulmalıdır.

6.1.1. *Gerçekleştirilebilme*

Planın en önemli özelliği eldeki kaynak ve teknik imkânlar kullanılarak istenilen sürelerde gerçekleştirilebilir olmasıdır. Tüm kaynaklar makul ve çıkabilecek olumsuzluklara belli bir pay ayrılarak planlanmalıdır.

Plan bir çerçevedir. Planın gerçekleştirilebilmesi, planı oluşturan temel bileşenlerin yani görevlerin gerçekleştirilebilir olmasıyla sağlanır. Bu da analizdeki isteklerin makul ve tutarlı olmasını gerektirir. Fonksiyonel olarak gerçekleştirilmesi mümkün olmayan ve sık karşılaşılan birkaç örnek aşağıda listelenmiştir.

- **Girilmemiş bilginin işlemlerde kullanılması:** Yazılım, kendi başına bilgi üretemez. Girilen bilgilerden yola çıkarak belli çıktılar üretebilir. Sisteme girilmeyen bir bilginin işlemlerde kullanılması zaman zaman kullanıcı tarafından istenen bir durumdur. Örneğin şirket içi birim yapısının değişmesi gereksin. İnsan kaynakları yazılımında eski ve yeni birim yapısı eşleştirilmeden *Birimde çalışan personel* raporu alınamaz.
- **Formatsız alanların raporlanmada değişken olarak kullanımı:** Bilginin kodlanmadan sisteme girildiği, formatsız bir alan üzerinden sağlıklı rapor almak zordur. Örneğin personel türü alanı formatsız olursa müdür unvanı için *"Müdür"*, *"Mdr."*, *"M."* gibi birçok farklı giriş yapılabilir.

Mevcut teknolojiyle yapılması zor veya proje vizyonunu aşan talepler de plana eklenmemelidir. Örneğin internet sayfası üzerindeki basit bir metin alanı ile *Microsoft® Word* veya *Open Office* gibi metin uygulamalarının biçimlendirme özellikleri aynı seviyede değildir. Aynı özellikleri beklemek gerçekçi olmaz.

Aşırı performans beklentileri gerçekleştirilmesi zor olan diğer bir konudur. İnternet teknolojilerindeki ekran performans sorunları bunun en yaygın örneğidir. Kullanıcı haklı olarak internetteki ekrandan istemci sunucu mimarisindeki performansı ve kullanım kolaylığını beklemektedir. Ancak teknolojik kısıtlar henüz buna tam olarak müsaade etmediğinden birçok memnuniyetsizlik oluşmaktadır.

Yazılım projelerinde bir işin gerçekleşip gerçekleşmediğinin son kararını verecek müşteri ve kullanıcıdır. Kullanıcı memnun etmeyecek bir çıktı görevin tamamlanmasını sağlamaz. Proje ekibi alternatif çözümleri kullanıcıya göstermelidir. Kullanıcının seçeceği en uygun alternatifin plana eklenmesi birçok problemin önüne geçer.

6.1.2. *Görevlerin Bitti veya Bitmedi Şeklinde Net Takibi*

Görevlerin izlenmesinde yüzde oranlar yerine bitti bitmedi şeklinde iki temel seçeneği kullanmak kolaylık sağlar. Bir görev tamamlanma kriterlerini yerine getirilene kadar %0 bitmiş sayılır. Kriterler tamamıyla yerine getirilince görev %100 tamamlanmış olur. Buna ikili izleme denir [DeMarco-1986].

İkili izleme planlamadan ziyade bir takip konusudur. Ancak içerisinde iki farklı iş olan veya birden fazla aşamada tamamlanan görevlerin ikili izlemeyle yönetilmesi güç olacaktır. Proje takibinde ikili izleme kullanabilmek için plandaki görevler, tamamlanma oranı %0-100 olmasına imkân verecek şekilde bölümlenmelidir.

6.1.3. *Tutarlılık*

Birbiriyle tutarlı olmayan ihtiyaçlar gerçekleştirilemez. Planlanan görevler arasında da tutarlılık olmalıdır. Örneğin ihtiyaç analizinde *"personele sadece tek unvan verilebilir"*, *"personel farklı birimlerde farklı unvanlarla çalışabilir"* şeklinde iki ayrı madde bulunsun. Görevlendirme yapılmadan önce iki madde arasında bir uzlaşma sağlanmalıdır. Bu amaçla analizdeki madde *"personel tek birimde tek unvanla çalışabilir. Ancak farklı birimlerde farklı unvana sahip olabilir."* şeklinde değiştirilebilir. Bu yaklaşımda planlama, analiz aşamasının testi ve doğrulaması gibi bir işlev görmektedir. Tutarsızlık, ihtiyaç analizinden planlamaya miras kalabileceği gibi analizin yanlış anlaşılmasıyla doğrudan planlama aşamasında da oluşabilir.

6.1.4. *Görev Sürelerinin ve Maliyetlerin Gerçekçi Olması*

Plandaki her görevin süre ve maliyeti gerçekçi olarak belirlemelidir. Bu önceki projelerdeki tecrübeleri dikkate alan ayrıntılı bir tahmin çalışmasıyla mümkün olur. Süre ve maliyetin *gerçekçi planlanmama sebepleri* üst yönetim baskısı, aşırı iyimserlik ve önceki projelerle ilgili ölçümlerin sağlıklı yapılmaması şeklinde sayılabilir.

Gerçekçi tahminler için proje büyüklük artışı ve teknoloji değişimine doğru mana verilmelidir. Örneğin 5 ekranın geliştirilmesi 1 hafta sürüyorsa 10 ekranın 2 hafta tutması beklenebilir. Oysa büyüklük artışı, geliştirme süresini çok daha fazla arttırır. Ayrıca görünümleri aynı bile olsa, arka planda sunucu istemci veya web gibi farklı teknoloji kullanılan iki ekranın geliştirme süresi arasında kat kat fark olabilmektedir.

6.1.5. *Görev Sürelerinin Kısa Tutulması*

Proje planındaki görev süreleri mümkün olduğunca kısa tutulmalıdır. Sürenin kısa tutulması, plandaki sapmaların daha önce fark edilmesini sağlar. Kontrol aralığı azalacağından kişilerin işlerine olan konsantrasyonları da artar. Kesin bir kural olmamakla birlikte en uzun görev süresi bir haftayı aşmamalıdır.

Görev sürelerinin uzun olması, yapılacak işlerin temel görevlere bölünmesinde hata olduğuna işaret ede(bili)r. Bu tür işlerin içlerinde genellikle birden fazla iş gizlidir.

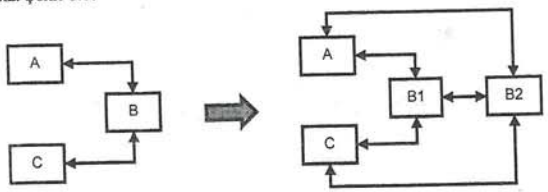
6.1.6. *Görevlerin Tek Bir Kişinin Yapılacağı Seviyeye Kadar Bölünmesi*

Proje planındaki işlemler tek bir kişinin yapacağı görevlere kadar bölünmelidir. İki kişiye tek bir iş atanması durumunda işin kontrolü zorlaşır. Kimin işin ne kadarını yaptığı, kişisel performans ve işin zamanında bitmemesi durumunda sorumluluğun kimde olduğu belirsizleşir.

İşlerin bölümlenmesi, karmaşıklığı azaltır ve sistemin anlaşılmasını kolaylaştırır. Karmaşık işler daha fazla bölümlenmelidir. Örneğin kurum için yeni bir teknolojide yazılacak projede atılacak her adım tek tek ele alınmalıdır. Böylesi bir durumda, tek bir ekranla ilgili işler dahi birçok alt göreve bölünebilir. Basit ve bilinen işleri ise aşırı bölmeye gerek yoktur. Mevcut teknoloji kullanılarak ilave bir ekran yazılması, bilinen bir işlem olduğundan alt işlere bölmeye gerek yoktur.

6.1.7. *Kişiler Arası İş Paylaşımında Aşırı Ayrıntıdan Kaçınma*

Bölümlemede aşırıya kaçılmaması tavsiye edilir. Basit ve tek kişinin yapabileceği bir görevi birden fazla kişiye paylaştırmak, gereksiz iletişim problemlerine ve yönetim zorluklarına yol açar. İşler kişiler arasında paylaştırılırken yapılacak her yeni bölümlemenin iletişim ihtiyacını neredeyse iki katına çıkartacağını unutmamak gereklidir; bkz. Şekil 6.1.



Şekil 6.1. İşlerin aşırı bölümlenmesi

Görev bölünmesinde bütünleşik çalışan yazılım bileşenleri yol göstericidir. Örneğin nesne, ekran ve raporlar bütünleşik yazılım bileşenleridir. Bu bileşenler, yapıları itibarıyla temel bir amaca sahiptir ve herbiri için ayrı bir görev tanımlanabilir. Tek bir yazılımcı tarafından geliştirilmeleri daha uygun olacaktır. Bir ekranın farklı kısımlarının iki ayrı kişi tarafından geliştirilmesi, çok doğru bir tercih değildir.

6.2. Planın Bütünüyle İlgili Kalite Faktörleri

Kaliteli bir planlama için, proje planının bütünü ilgilendiren faktörler de göz önüne alınmalıdır. Planın süresi çok uzun olmamalı, kontrol aralıkları tanımlanmalı ve çalışmalar mesai içini kapsayacak şekilde planlanmalıdır. Ayrıca planın her aşaması yakın seviyede ayrıntı içermelidir.

6.2.1. *Plan Süresinin Kısa Tutulması*

Proje, ilk ürün kısa sürede ortaya çıkacak şekilde planlanmalıdır. Elbette proje süresi proje büyüklüğüne göre değişir. Ancak beş yılda bir önemli teknolojik gelişmeler olan yazılım alanında toplam süresi iki, hatta bir yıldan uzun olan projeler anlamını kaybedebilir. Üç yıl olarak planlanan bir yazılım projesi hedefe ulaştığında, şirketin bambaşka bir hedefi olabilir. Dikkate edilirse *Microsoft*® ve *Oracle*® gibi belli başlı yazılım şirketlerinin ana sürüm çıkarma aralığı genellikle 12 ila 18 ay arasında değişir. Bu süre nadiren 2 yılı aşar.

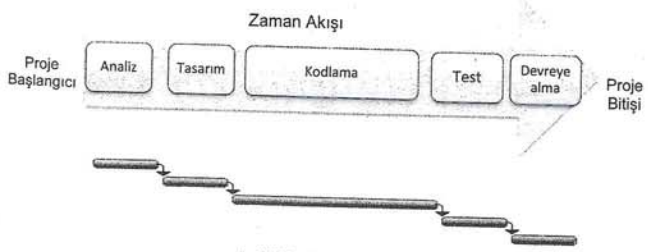
Proje süresinin uzun olması halinde, proje alt projelere bölünerek ilk ürün yine kısa sürede ortaya çıkartılabilir. Projenin amacının dışına çıkmasının önündeki en büyük engel, çalışan bir ürün olmasıdır. Aksi halde teorik çalışmalar, farklı teorik çalışmalarla yer değiştirir. Bitmeyen analiz ve planlama süreçleri kısır döngüye dönüşür.

6.2.2. *Kısa Süreli Kilometre Taşları*

Kısa süreli kilometre taşı veya kontrol noktaları tanımlamak, yapılanlar ile planın kısa aralıklarla karşılaştırılmasını sağlayarak, plandan sapılmasına engel olur. Şelale modeli kullanılarak geliştirilen klasik bir plan örnek olarak verilmiştir; bkz. Şekil 6.2. Böylesi bir planda kısa süreli kilometre taşları kullanımı zordur.

Görev İsmi	Süre	Başlangıç Tarihi	Bitiş Tarihi
Analiz	1 ay	03.01.2011	28.01.2011
Tasarım	1 ay	31.01.2011	25.02.2011
Kodlama	3 ay	28.02.2011	20.05.2011
Test	1 ay	23.05.2011	17.06.2011
Devreye alma	1 ay	20.06.2011	15.07.2011

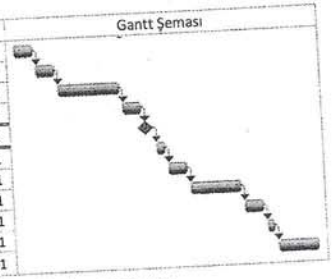
(Süreler mesai günlerine göre hesaplanmıştır)



Şekil 6.2. Klasik bir plan

Artımlı geliştirme veya evrimsel geliştirme gibi yinelemeli geliştirme yöntemleri kullanılarak, plana kısa süreli kilometre taşları eklenebilir. Böylece proje süresinin daha verimli kullanılması sağlanır; bkz. Şekil 6.3.

Görev İsmi	Süre	Başlangıç Trh.	Bitiş Tarihi
Analiz aşaması	10 gün	03.01.2011	14.01.2011
Tasarım aşaması	10 gün	17.01.2011	28.01.2011
Kodlama	30 gün	31.01.2011	11.03.2011
Test	15 gün	14.03.2011	01.04.2011
Sürüm 1	0 gün	01.03.2011	01.04.2011
Analiz aşaması	5 gün	04.04.2011	08.04.2011
Tasarım aşaması	10 gün	11.04.2011	22.04.2011
Kodlama	25 gün	25.04.2011	27.05.2011
Test	10 gün	30.05.2011	10.06.2011
Prototip sunum	5 gün	13.06.2011	17.06.2011
Devreye alma	20 gün	20.06.2011	15.07.2011



Şekil 6.3. Artımlı geliştirme yönteminin plana yansıması

Sürüm 1 kilometre taşı ve onay noktası olarak seçilmiştir. Eğer *Sürüm 1* yüksek kalitede planlanır ve özellikleri dikkatli seçilirse devreye alınacak bir ürün de olabilir. Bu durumda projenin ilk sürümü 3,5 ay önce devreye alınmış olacaktır.

6.2.3. *Dengelenmiş Kaynaklar*

Plan içerisinde, bir kişiye aynı anda birden fazla görev atanması veya görev süresinin ancak fazla mesaiyle tamamlanabilecek şeklinde kısa tutulması gibi kaynağın aşırı yüklendiği durumlar olmamalıdır. Bu durum birçok belirsizliğe yol açar ve planı izlemek zorlaşır. Görevler arası ilişkiler *kaynak takvimleri, Gantt ve Ağ şemaları* ve üzerinden incelerek bu tür tutarsızlıklar giderilebilir; bkz. Kısım 5.7.4.

6.2.4. *Mesai İçi Çalışmaların Kapsanması*

Kişinin fazla mesai yapacağı düşünülerek hazırlanan bir plan gerçek dışıdır! Elbette ekibin yoğun çalıştığı zamanlar olacaktır. Ancak bunlar doğrudan plana eklenirse, sorun çıkması durumunda kullanılabilecek yedek zaman ve güç bulunamaz. Ekip zaten günde 16 saat çalışıyorsa bu süre 20 saate çıkartılamaz. Yedekleme olmadığından acil durumlarda yapılacak her çalışma, proje süresini otomatik olarak uzatır.

Dikkat edilecek bir nokta yöneticilerin çok çalışmasıdır. Proje yöneticisi veya üst yönetimin çok çalışması psikolojik olarak diğer kişilerin az çalıştığını düşündürmektedir. Sebepsiz yere ekibini fazla mesaiye zorlayan yöneticilerle karşılaşmak mümkündür. Yönetici, kendi çalışmasını da makul sınırlar içerisinde planlamalıdır.

6.2.5. *Planın Tüm Faaliyetleri İçermesi*

Önemli görünsün görünmesin kişilerin yapacağı tüm faaliyetler proje planında yer almalıdır. Yazılım geliştirmeyle dolaylı ilgili ancak sistem altyapısı için yapılması zorunlu olan program kurulum, belgeleme ve kullanıcı eğitimi gibi faaliyetler proje planında unutulmamalıdır. Ayrıca tatil ve izin gibi süreler de dikkate alınmalıdır. Aksi halde bu işlemler proje süresinde önceden hesaplanamayan gecikmelere yol açar.

Planlanmayan bir faaliyet sadece süreyi uzatmaz, bu tür faaliyetleri yapmak tamamen unutulabilir. Bu birçok hatayı da beraberinde getirir. Örneğin kullanıcıya eğitim verilmezse, birçok veri hatalı girilir. Bu tür sorunlar da yazılım ekibine yansır.

6.2.6. *Hiyerarşideki Yayılım ve Derinlik Sayılarının Küçük Tutulması*

Kişinin tek seferde en fazla 7 (en fazla 10) maddeyi aynı anda görebileceği ve inceleyebileceği genel bir psikolojik kabuldür. Bu yüzden hiyerarşinin aşırı genişlemesi uygun değildir. Bölümleme yapılırken bir özet görev için 10'dan fazla alt görev kullanılmamalıdır. Bunu sağlamak için gerekirse farklı modül veya yazılım paketleri tasarlanabilir. Aynı şekilde hiyerarşinin aşırı derinleştirilmesi de karmaşıklığı artırarak, anlamayı zorlaştırır. Burada da 7 rakamı sınır olarak kabul edilir.

6.2.7. *Tüm Aşamaları Ayrıntılı Planlamak-Planın Homojen Olması*

Yazılım geliştirme süreci planlama, analiz ve tasarım gibi temel aşamalardan oluşur. Her aşama aynı titizlikle planlanmalıdır. Plandaki hiyerarşik yapı, plan kalitesi hakkında fikir verir. Planın farklı kısımlarında hiyerarşideki genişlik ve derinliğin farklı olması, aynı ayrıntıda inceleme yapılmadığını gösterir. Ayrıntılı olmayan kısımlardaki işlerin başarısız olma riski daha fazladır. Aşağıda bu tür bir plan örneklenmiştir.

Görev İsmi
Analiz aşaması
Tasarım aşaması
Veri model tasarımı
Nesne Model tasarımı
Ekranların tasarımı
Yeni personel giriş ekranı
Personel listeleme
Personel atama ekranı
Personel gözlem ekranı
Personel izin bilgileri girişi
Raporların tasarımı
...

Yukarıdaki plan birkaç açıdan eksiktir. Aşamalar ve aşamalarda yapılacaklar aynı derinlikte incelenmemiştir. Örneğin analiz ve tasarım aynı seviyede ayrıntılı değildir. Birçok planda görülen bu hatanın belirtisi planda çok uzun süreli işler olmasıdır. Bu hata "*Analiz etmeden ayrıntılarını bilmek zordur.*" düşüncesinin bir neticesidir. Ancak planda tasarım ayrıntıları olduğuna göre proje ekibi belli kararlara varmış demektir. Analiz de temel noktalara ayrılarak aşama aşama izlenebilir.

Yukarıdaki planda tasarımdaki hiyerarşik bölümlemede de bir sorun vardır. Ekran tasarımındaki ayrıntı düzeyi ile veri model ve rapor tasarımlarındaki düzey farklıdır. Her ekran için bir görev tanımlıyken raporlar sadece tek bir madde ile gösterilmiştir. Oysa ekran ve rapor benzer arayüzlerdir. Planlanma şekilleri de benzer olmalıdır.

Planın aşağıdaki şekilde düzenlenmesi kalite artışı yönünden faydalı olur. Yeni düzenlemede analiz ve tasarım aşamaları ayrıntılı hale getirilmiştir. Tasalanacak raporlar da plana eklenmiştir. Böylece plan daha homojen ve sağlıklı bir yapı kazanmıştır.

Görev İsmi
Analiz Aşaması
Genel personel bilgileri analizi
Personel izin bilgileri analizi
Ortak kullanılabilecek bilgilerin ve bütünleşme yöntemlerinin analizi
Tasarım aşaması
Veri model tasarımı
Genel personel bilgileri tasarımı
Personel izin bilgileri tasarımı
Nesne Modeli
Personel üzerindeki genel nesne ve senaryoların tasarımı
İzin işlemi nesne ve senaryoların tasarımı
Sistemin ortak kullanılacak nesnelerinin tasarımı
Ekranların tasarımı
Yeni personel giriş ekranı
Personel listeleme
Personel atama ekranı
Personel gözlem ekranı
Personel izin bilgileri girişi
Raporların tasarımı
Personel listesi
Personel Kartı (matbu rapor)
İşyeri listesi
...

6.3. Modüler Planlama

Yazılımın birbiriyle bütünleşik çalışan alt modüllere bölünmesi, planlama açısından da faydalıdır. Binlerce satırdan oluşan bir proje planı profesyonel görülebilir. Ancak nadiren faydalıdır!

Planı basitleştirmek ve izlenebilir hale getirmek için anlamlı parçalara bölmek gerekir. Hiyerarşik bölümleme dikey bir parçalama sağlar, dolayısıyla plan yine bir bütündür. Buna karşın planı modüler şekilde tasarlamak, sistemin bir kısmını tamamen ayrı düşünmeyi sağlar. Plan, birbiriyle bütünleşik birçok yarı bağımsız parçadan oluşacak şekilde basitleşir.

Aşağıda bölümlere ayrılmadan geliştirilen bir sistem planı mevcuttur. Burada personel ve mali işler birimleri için geliştirilecek bir yazılım projesi örneklenmiştir. Ancak plan sanki tek bir modül varmış gibi geliştirildiğinden, personel sistemini veya mali işler sistemini kendi başlarına bir bütün olarak görmek ilave çaba gerektirir. Dolayısıyla bu iki sistemle ilgili yapılanları ayrı ayrı olarak incelemek de zorlaşır. Bu durum karmaşayı arttırır ve projenin takibini zorlaştırır.

Görev İsmi
Analiz aşaması
Personel Sistemi analizi
Mali İşler Sistemi analizi
Tasarım aşaması
Personel ekranı tasarımı
Muhasebe fiş giriş ekranı tasarımı
Personel izin bilgileri girişi
Fatura Ekranı tasarımı
Kodlama
Personel ekranı yazılması
Muhasebe fiş giriş ekranı yazılması
Personel izin bilgileri girişi ekranı yazılması
Fatura giriş ekranı yazılması
Test
Personel ekranı testi
Muhasebe fiş giriş ekranı testi
Personel izin bilgileri girişi ekranı testi
Fatura giriş ekranı testi
Bileşen test
Sistem bütünleşme testi
Kullanıcı kabul testi
Devreye alma

Yukarıdaki bütünleşik sistem alt sistemlere bölünerek tekrar planlanabilir:

Görev İsmi
Personel Modülü
Analiz aşaması
Personel Sistemi Analizi
Tasarım aşaması
Personel ekranı tasarımı
Personel izin bilgileri girişi
Kodlama
Personel ekranı yazılması
Personel izin bilgi giriş ekranı yazılması
Test
Personel ekranı testi
Personel izin bilgi giriş ekranı testi
Bileşen test
Sistem bütünleşme testi
Kullanıcı kabul testi
Devreye alma

Görev İsmi
Mali İşler Modülü
Analiz aşaması
Mali İşler Sistemi analizi
Tasarım aşaması
Muhasebe fiş giriş ekranı tasarımı
Fatura Ekranı tasarımı
Kodlama
Muhasebe fiş giriş ekranı yazılması
Fatura giriş ekranı yazılması
Test
Muhasebe fiş giriş ekranı yazılması
Fatura giriş ekranı yazılması
Bileşen test
Sistem bütünleşme testi
Kullanıcı kabul testi
Devreye alma

Dikkat edilirse yukarıdaki yeni tasarımda görev sayısı neredeyse aynıdır. Sadece sistem modüllere ayrılarak görevler bu modüllere taşınmıştır. *Personel Sistemiyle* ilgili görevler personel sistemine, *Mali İşlerle* ilgili işlemler mali işler modülüne taşınmıştır. Bu sayede gözle dahi kolayca fark edilebilen bir basitleşme sağlanmıştır.

Yazılım geliştirme bir ekip işidir. Ekipte görev paylaşımının sağlıklı yapılması için görevler arasındaki sınırlar açık bir şekilde tanımlanmalıdır. Alt sistemler, görevin mahiyet ve sınırını belirginleştirir. Böylece farklı kişilerin aynı projede çalışması ve görev takibi kolaylaşır. Örneğin iş akış ve yetki sistemini farklı birer kişi tasarlayabilir.

Sistemin alt sistemlere ayrılması, sadece planlama için değil tasarımı basitleştirmek, testleri planlamak ve tekrar kullanım gibi birçok konuda fayda sağlar. Ortak ihtiyaçları ayrı modüller olarak tasarlamak, bunların farklı projelerde tekrar kullanımını sağlar. Veritabanı ve güvenlik gibi özel modüllerde, farklı uzman ekipler oluşabilir.

6.3.1. *Modülerlik ve Hiyerarşik Ayrım Farkı*

Modüler yapı, hiyerarşik ayrımdan farklıdır. Bir modülün özellikleri planın farklı seviyelerde dağılmış olabilir. Planın bir kısmını hiyerarşinin en üst kademesinden bölüp ayrı takip etmek, modüler geliştirme yapmak manasına gelmez. Aşağıda bu konuda örnek olarak personel sistemiyle ilgili bir ekran geliştirme planı verilmiştir.

Görev İsmi	Modül
Personel Modülü	
Analiz aşaması	
Personel Sistemi Analizi	
Güvenlik analizi	Güvenlik
Tasarım aşaması	
Personel ekranı tasarımı	
Personel ekran güvenliğinin tasarımı	Güvenlik
Personel izin bilgileri girişi ekranı tasarımı	
Personel izin bilgileri girişi ekran güvenliğinin tasarımı	Güvenlik
Kodlama	
Personel ekranı yazılması	
Güvenlik kodlarının eklenmesi	Güvenlik
Personel izin bilgileri girişi ekranı yazılması	
Güvenlik kodlarının eklenmesi	Güvenlik
Test	
Personel ekranı testi	
Personel izin bilgileri girişi ekranı yazılması	
Ekran güvenlik testi	Güvenlik
Bileşen test	
Sistem bütünleşme testi	
Kullanıcı kabul testi	
Devreye alma	

Yukarıdaki planda güvenlikle ilgili yapılacaklar birçok seviyeye dağılmıştır. Bu dağınıklık, bir standart oluşturmayı zorlaştırır. Tasarım ve kodlamada güvenlikle ilgili görevler ekranların altına yazılmıştır. Oysa test aşamasında güvenlik sadece tek bir adımdır. Görevlerin hiyerarşide farklı katmanlara dağılması, ayrı bir güvenlik modülü geliştirmeyi güçleştirir. Çünkü modülün bileşenleri diğer yapılacakların arasında kaybolur ve bütünleşik olarak görülemez. Aynı planın modülere ayrılmış hali aşağıdadır.

Görev İsmi
Güvenlik Modülü
Analiz aşaması
Güvenlik analizi
Tasarım aşaması
Güvenlik kütüphanelerinin tasarlanması
Güvenlik kütüphanesinin diğer sistemlerle bütünleşmesinin tasarımı
Kodlama
Güvenlik kütüphanelerinin kodlanması
Güvenlik kütüphanelerinin ekranlardan çağıran kodların yazılımı
Güvenlik kodlarının ekranlara ilavesi
Test
Güvenlik kütüphane bütünleşme testi
Devreye alma

Görev İsmi
Personel Modülü
Analiz aşaması
Personel Sistemi Analizi
Tasarım aşaması
Personel ekranı tasarımı
Personel izin bilgileri girişi ekranı tasarımı
Kodlama
Personel ekranı yazılması
Personel izin bilgileri girişi ekranı yazılması
Test
Personel ekranı testi
Personel izin bilgileri girişi ekranı yazılması
Ekran güvenlik testi
Bileşen test
Sistem bütünleşme testi
Kullanıcı kabul testi
Devreye alma

Planın modüllere ayrılması güvenlik ve personel sistemlerini ayrı ayrı geliştirilmeye imkân verir. Sistemler arası bütünleşmeyi sağlamak için gerekli görevler her iki planda da yer almıştır. Bu görevler, ekip yapısına göre tek bir planda da gösterilebilir.

6.3.2. *Modül Öncelikli veya Süreç Öncelikli Planlama*

Ekibin tercihine göre planlama süreç veya modül öncelikli yapılabilir. Ancak sistemi önce modüllere ayrılmak sonrada süreç yaklaşımı kullanmak, alt planları basitleştirir ve yönetimi kolaylaştırır.

Görev İsmi
Analiz
Personel Sistemi analizi
Mali İşler Sistemi analizi
Tasarım
Personel Sistemi tasarımı
Mali İşler Sistemi tasarımı
Kodlama
Personel Sistemi kodlanması
Mali İşler Sistemi kodlanması
Test
Personel Sistemi testi
Mali İşler Sistemi testi
Devreye alma
Personel Sistemi devreye alma
Mali İşler Sistemi devreye alma

Süreç öncelikli planlamada, süreç aşamaları hiyerarşinin üst katmanlarına yazılır. Daha sonra her aşamada yapılacak görevler hiyerarşik olarak alt katmanlara yerleştirilir. Yukarıda süreç öncelikli bir plan örneği verilmiştir. Böylesi tasarımlarda modülleri tespit etmek güçtür. Orta ölçekli bir planda dahi yüzlerce adım olabileceği düşünüldüğünde, süreç odaklı bir yaklaşımda modüllerin tespiti neredeyse imkânsızdır.

Öncelikle modülleri belirlemek, daha sonra bunların geliştirme ve entegrasyon süreçlerinin ayrıntılı planlanması birçok problemin çözümüdür. Bu yapıda görevler modüllerin altına hiyerarşik olarak oluşturulur. Yukarıda verilen ilk planın modül öncelikli hali aşağıda gösterilmiştir.

Görev İsmi
Personel Sistemi
Analiz
Tasarım
Kodlama
Test
Devreye alma
Mali İşler Sistemi
Analiz
Tasarım
Kodlama
Test
Devreye alma

Yukarıdaki modül odaklı planda geliştirilecek tüm modüller açıkça görülebilmektedir. Yeni durumda her modül için ayrı bir devreye alma tarihi belirlemek mümkündür. Planlarda çok fazla ayrıntıya girilmediğinden, modüller arası bütünleşme gösterilmemiştir. Bu işlem analiz tasarım gibi aşamaların alt görevlerinde yer almalıdır.

6.4. Yapılacak İşleri Paralel Hale Getirmek

Yazılım projesinin toplam süresini, arka arkaya seri olarak yapılan işlerin süreleri toplamı belirler. İşleri paralel hale getirmek proje süresini kısaltır. Bu işleme *hızlı yol alma (fast track)* ismi verilir. Seri planlarda kişiler diğer işlerin bitmesini bekler. İşler paralel planlanarak ekibin projeye erken dâhil olması sağlanır ve çalışma verimi artar.

Paralel hale getirilecek görevleri belirlemekte yapılacak işlemleri farklı bakış açılarına göre ayıran yöntemler kullanılabilir. İhtiyaçların sınıflanması bu tür bir yöntemdir. Yazılımda ihtiyaç, fonksiyonel ve fonksiyonel olmayan ihtiyaç olarak kabaca ikiye ayrılır. Bu ayrımdan yola çıkarak donanım temini ve işletim sistemi kurulumu gibi fonksiyonel olmayan ihtiyaçlarla ilgili çalışmalar projenin erken aşamalarında, diğer işlerden kısmen bağımsız olarak planlanabilir.

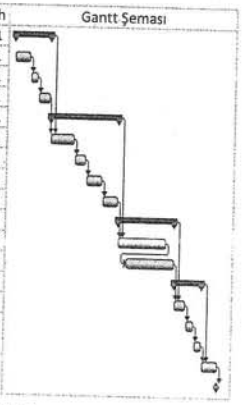
Teknik araştırma ve müşteriye yönelik iş analizi de paralel hale getirilebilir. Örneğin sistemler arasında yapılacak bütünleşmenin teknik altyapı analiz ve tasarımı, iş analizine paralel olarak başlayabilir. Ayrıca önceki projedekine benzer bir işlem farklı platformda yapılacaksa, ihtiyaç analizi ve mimari tasarıma paralel incelenebilir.

Sistemler arası entegrasyon birçok teknik ayrıntı barındırır. Örneğin bir belge yönetim sistemi ile personel sistemi entegre edilmek istensin. Personel sistemi *java* platformu kullanarak geliştirilmiş ilişkisel veritabanı uygulaması olsun. Bu durumda bir personel evrakının, *java* tabanlı veritabanı uygulaması kullanılarak, belge yönetim sistemine güvenli şekilde eklenmesi, silinmesi, değiştirilmesi kesinlikle çözülmesi gereken *teknik konulardır*. Kodlama ekibi bu konudaki çalışmalara iş analizi bitmeden başlayabilir.

Yeni bir proje geldiğinde önceki tecrübelerin bu projede nasıl kullanılabileceği yazılım ekibi tarafından araştırılmalıdır. Örneğin yazılım dilleriyle gerçekleştirilen benzer işler projeden bağımsız araştırılabilir. Sürekli *Microsoft® Visual Studio* ile proje geliştiren bir ekibe *Java* tabanlı bir yazılım isteği gelsin ve bir proje planı oluşturulsun; bkz. Şekil 6.4. Analiz ekibi proje ihtiyaç analizine başlarken, geliştirme ekibi de *Microsoft® Visual Studio* ile yapılan işlerin *Java* ortamındaki karşılıkları incelemeye başlayabilir. Bu çalışmalara birkaç örnek:

- Ortak fonksiyon kullanımı, nesne tasarımı ve kütüphane yapısı
- Kullanıcı ara yüzündeki temel fonksiyonlar
- Veri erişim ve veritabanı işlem fonksiyonları
- İşletim sistemindeki dosyalara erişim, yazma ve okuma gibi işlemler

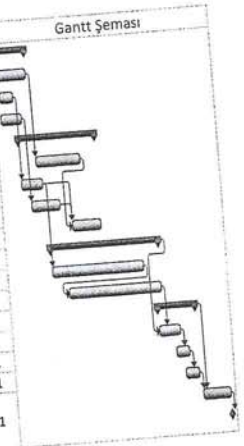
Görev İsmi	Süre	Başl. Trh	Bitiş Trh
Analiz aşaması	**45gün**	**03.01.11**	**04.03.11**
İş analizi	20gün	03.01.11	28.01.11
Önceki projelerle benzerlik inceleme	10gün	31.01.11	11.02.11
Sistemler arası bütünleşme analizi	15gün	14.02.11	04.03.11
Tasarım aşaması	**85gün**	**07.03.11**	**01.07.11**
Kullanıcı sistem tasarımı	30gün	07.03.11	15.04.11
Önceki projelere benzerliklerin tasarımı	15gün	18.04.11	06.05.11
Sistemler arası bütünleşme tasarımı	20gün	09.05.11	03.06.11
Test senaryolarının yazılması	20gün	06.06.11	01.07.10
Kodlama	70gün	04.07.11	07.10.11
Kod geliştirme	60gün	04.07.11	23.09.11
Birim test	60gün	18.07.11	07.10.11
Test	35gün	10.10.11	25.11.11
Bileşen test	15gün	10.10.11	28.10.11
Sistem bütünleşme testi	10gün	31.10.11	11.11.11
Kullanıcı kabul testi	10gün	14.11.11	25.11.11
Devreye alma	**20gün**	**28.11.11**	**23.12.11**
Proje teslimatının tamamlanması	**0 gün**	**23.12.11**	**23.12.11**



Şekil 6.4 Paralel hale gelmeden önceki plan

Plandaki işler paralel hale getirilirken analiz ve tasarım gibi aşamalardaki görevler türüne göre alt görevlere bölünerek paralel hale getirilebilir; bkz. Şekil 6.5.

Görev İsmi	Süre	Başl. trh.	Bitiş Trh.
Analiz aşaması	**20gün**	**03.01.11**	**28.01.11**
İş analizi	20gün	03.01.11	28.01.11
Önceki projelerle benzerlik inceleme	10gün	03.01.11	14.01.11
Sistemler arası bütünleşme analizi	15gün	03.01.11	21.01.11
Tasarım aşaması	**50gün**	**17.01.11**	**25.03.11**
Kullanıcı sistem tasarımı	30gün	31.01.11	11.03.11
Önceki projelere benzerliklerin tasarımı	15gün	17.01.11	04.02.11
Sistemler arası bütünleşme tasarımı	20gün	24.01.11	18.02.11
Test senaryolarının yazılması	20gün	28.02.11	25.03.11
Kodlama	**70gün**	**07.02.11**	**13.05.11**
Kod geliştirme	60gün	07.02.11	29.04.11
Birim test	60gün	21.02.11	13.05.11
Test	**25gün**	**09.05.11**	**10.06.11**
Bileşen test	15gün	09.05.11	27.05.11
Sistem bütünleşme testi	10gün	23.05.11	03.06.11
Kullanıcı kabul testi	10gün	30.05.11	10.06.11
Devreye alma	20gün	13.06.11	08.07.11
Proje teslimatının tamamlanması	0gün	08.07.11	08.07.11



Şekil 6.5 Paralel hale getirilmiş plan

Görev süreleri aynı kalmasına rağmen örneklenen planın paralel hale getirilmesi yaklaşık 5,5 aylık yani %50 civarında bir zaman kazandırmıştır. Burada sadece kaçınılmaz olarak ilişkili işler birbirini izlemektedir. Örneğin projenin teknik analiz ve tasarımı yine birbirine bağlı şekilde yapılmaktadır. Paralel planlamanın yararları yanında karmaşıklığı ve dolayısıyla riskleri arttırdığına da dikkat edilmelidir.

6.5. Kapalı Çevrim Planlama

Bir projede diğer işlerden tamamen bağımsız bir görev olması projenin bütünlüğüne zıt bir durum oluşturur; bkz. Şekil 6.6. Bu yüzden kaliteli bir planda tüm görevlerin birbirine bağlı olması önerilmektedir. Bu kapalı ağ veya kapalı çevrim planlama (*closed network or closed loop planning*) olarak isimlendirilir[Harris-2010]. Bu şekilde oluşturulan planın canlı uygulamayı daha iyi yansıtması sayesinde görevler ve görevler arası bağlantılar daha kolay izlenebilir.

Şekil 6.6. Açık çevrim planlama

Planda kopukluk olmaması için araya gerçek veya sanal işlemler eklenebilir. Bazı işlemler de alt işlemlere bölünebilir. Örneklenen projede, test işlemindeki sistem testi, devreye alınma aşamasıyla birlikte ele alınarak plan kapalı çevrim olacak şekle getirilmiştir; bkz. Şekil 6.7.

Şekil 6.7. Kapalı çevrim planlama

Kapalı çevrim planlama, daha çok malzeme üretim hatlarında kullanılan ve bu hatlardaki kayıpları azaltmayı hedefleyen bir modeldir. Yazılım proje planlarında tüm görevleri birbirine bağlamak her zaman kolay bir işlem olmayabilir. Bu yüzden *kapalı çevrim planlamanın* yazılım projelerine uygulanmasında çok katı davranmak, planın aşırı karışık hale gelmesine yol açar. Bunu önlemek için sadece aynı seviyedeki görevler arasındaki bağlantıların doğrudan gösterilmesi diğer bağlantıların ise ayrıntıya inildiğinde görünür olması tercih edilebilir. Örneğin analiz özet görevi, tasarım özet görevine bağlı çizilebilir. Alt bir analiz görevi ile alt bir tasarım görevi arasındaki bağlantı ise ancak ayrıntıya inince görülmelidir.

6.6. Planlamada Yapılan Hatalar ve Sebepleri

Planlama yapılan hataların birçok ortak noktası ve benzerlikleri mevcuttur. Bu hataların arka planındaki psikolojik etkenler de benzerdir ve hatalar bu etkenler dikkate alarak incelenmelidir. *Planlamada* kurum ve kişiler tecrübe eksikliği, mali şartlar, müşteri ve üst yönetim zorlaması gibi sebeplerle hataya düşülebilmektedir.

6.6.1. *Hataların sebepleri*

Planlama hataları işi basit görmek, proje büyüklüğünü tahmin edememek, önceki başarıları yanlış değerlendirmek ve dış baskılara boyun eğme gibi yaklaşımlar neticesinde oluşabilir. Hatanın birçok sebebi olmakla birlikte ana sebep plan yapmanın önemine inanmamaktır. Belli bir çalışma standardı olan bir proje yöneticisi, şartlar ne kadar zorlasa da plansız bir projeye başlamayı reddedecektir.

Acelecilik

Plansızlığın altında yatan önemli bir sebep aceleciliktir. Yazılım tamamlanmadan ortaya çıkacak ürünü tam olarak gösteren bir teknik henüz mevcut değildir! Kısa vadede de olmayacaktır. Öte yandan insan tabiatında, biran önce gösterilecek bir şeyler ortaya koyma isteği vardır. Ancak bu istek, projenin istenen kalitede olması için yapılması gereken analiz ve tasarım gibi ön çalışmalara engel olmamalıdır.

İşi Basit Görmek

En basit bir problemi çözmek için bile, bir model, yönteme ve plana ihtiyaç vardır. Ayrıca bir işi modellenmeden, basit olup olmadığını tam olarak anlamak da zordur. Burada işten kastedilen yazılım geliştirme, analiz veya veri modelleme olabilir.

Önceki Başarılı Projeler

Şirketlerin ilk projeleri genellikle küçük ve pratik zekâya dayanır. Birçok şirketin başlangıç noktası, piyasada daha önce fark edilmemiş bir yenilik veya önemli bir soruna yeni ve farklı bir çözümdür. İlk anda ekip küçüktür; hızlı ve yoğun bir çalışmayla ürün ortaya çıkar. Eğer beklentileri gerçekten karşılıyor ve pazarlama konusunda da başarılıysa, ürün hemen parlar ve yaygınlaşır. Özellikle internet, ürün yaygınlaşması konusunda bulunmaz fırsatlar sunmaktadır.

Firmanın, ilk andaki başarısını ve bunu başarmak için izlediği yöntemi her proje için geçerli zannetmesi önemli sorunlara yol açar. Çünkü yazılım kullanıcı sayısı arttıkça analiz, planlanma, kalite ve testlere daha fazla yatırım yapmak gerekir. Bu yatırımın yapılmaması yüzünden bilgi işlem sektöründe saman alevi gibi parlayıp sönen birçok şirket görülmektedir. Proje ekipleri kendi başarılarının esiri olmadan, proje için doğru olan ne ise onu yapmalıdır.

Planlamayı Ertelemek

Planı ertelemenin bir sebebi de "*Şu an çok yoğunuz, işler bitsin ayrıntılı plan yapacağız*" anlayışında gizlidir. Hakikatte *plansızlık ve yoğunluk* kardeştir. Plansız işler sağlıklı yürütülemeyeceğinden maalesef kapsamlı bir plana bir türlü sıra gelmez. Planlamaya ayrılamayan zamanın kat kat fazlası, projede ortaya çıkan sorunları gidermek için harcanır.

Dış Baskılara Boyun Eğmek

Dış baskılara boyun eğmek, proje ekibinin yapılacaklar ve planla ilgili gerçekleri bilmesine rağmen, piyasa şartları veya üst yönetim zorlaması sebebiyle altından kalkamayacakları bir plana imza atmasıdır. Ürün çıkış tarihlerine yetişmek, rekabetçi bir ortamda rakip ürünlerin yeni özelliklerine cevap vermek veya kurumdaki idari yöneticilerin teknik konularda aşırı talepleri bu tür baskılara örnek gösterilebilir.

Proje Büyüklüğünü Tahmin Edememek

İşi basit görmenin bir yansıması olan bu durum kapsamı doğru anlayamamanın neticesidir. Bazı durumlarda sadece birkaç ekrandan oluşan bir yazılıma ihtiyaç duyulur. Bu tür projeler küçük görülüp birkaç günde bitecek şekilde planlanır. Bu tür projeler, süre hesabında en çok hata yapılan projelerdir. Yazılım ne kadar küçük olursa olsun özellikle kullanıcı sayısı yüksekse bütünleşme, güvenlik ve iş süreçlerinin getirdiği ihtiyaçlar ile proje kapsamı genişler ve geliştirme en az birkaç ay sürer.

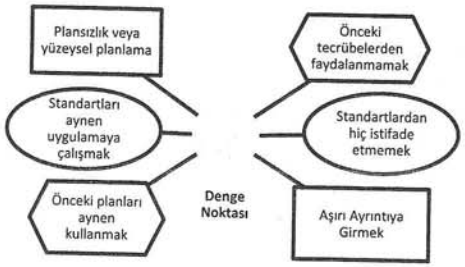
> Büyük kurumların (çok) küçük projesi olmaz!

Büyük kurumlar için geliştirilen bir proje ne kadar basit ve kolay görünürse görünsün süre tahminin ayrıntılı bir plan ve analiz yaptıktan sonra söylemek en iyisidir.

> **Senaryo:** ***Çok kullanıcılı basit(!) bir proje***
>
> Kurum içi bir yazılım ekibinden bilgi toplama amaçlı tek bir ekran hazırlaması istenir. Kullanıcı sayısı 5000 civarındandır. Proje basit göründüğünden proje yöneticisi sadece bir hafta süre ister. Yazılım birkaç günde hazırdır. Bu projede kullanılan teknoloji kurum için oldukça yenidir ve daha önce sadece az kullanıcılı birkaç pilot projeler yapılmıştır. Test fonksiyonları kontrol amaçlı yapılır. Performans ise kısaca gözden geçirilir.
>
> Proje devreye alınır. Ancak sistemi eşzamanlı kullanan kullanıcı sayısı bir anda artar ve performans düşer. Yazılım, arka plandaki kaynakları kilitlemekte ancak zamanla doğru şekilde bırakmamaktadır. Kurum içi ekip birkaç günlük bir çalışma yapar ancak sorun azalmasına rağmen tam çözülemez. Proje için ana yazılım firmasından destek istenir. Ortak çalışmalarla makul bir çözüm bulunması bir haftayı alır.
>
> *Sonuç* kullanıcı memnuniyetsizliği, yeni teknolojiye duyulan güvenin azalması ve yazılım ekibi için ancak uzun sürede telafi edilebilen itibar kaybıdır.

6.6.2. *Planlama Hataları*

Planlama hataları aşırı uçlarda gözlemlenir. Hiç plan yapmamak ve her şeyin planlamayabileceği zannıyla planlamada aşırıya kaçmak iki zıt kutuptur; bkz. Şekil 6.8. Ortak planlama hataları konusunda çeşitli çalışmalar yapılmıştır [McConnell-2001].



Şekil 6.8. Planlama Hataları

Hiç Plan Yapmamak

Hiç plan yapmamak planlama hatalarının en uç noktasıdır. Aylarca sürecek bir proje plansız yapılabilir mi? Bu sorunun cevabı kesinlikle "Hayır!" olmalıdır. Plan yapmamak açık bir hatadır. Kişisel tecrübe eksikliği, piyasa şartları veya üst yönetim zorlamasına boyun eğmek bu hataya yol açabilir. Projenin başarısızlığını neredeyse kesindir. Bu yüzden yol açtığı sorunları saymaya dahi gerek yoktur!

Yetersiz Planlama

Yetersiz planlama, yapılaması gereken işlemlerin bir kısmının unutulması şeklinde ortaya çıkar. Ekibin projeye uyum süreci, test, belgeleme, altyapı kurulumu, son kullanıcı eğitimi ve ürün kurulumu planlamada sıklıkla unutulan konulardır. Ayrıca kişilerin izin, hastalık ve tatil süreleri de unutulabilmektedir.

Riskleri Planlamamak

Projeyi düz bir çizgi şeklinde ilerleyecek şekilde düşünmek ve çıkabilecek olumsuzlukları dikkate almadan planlama yapmak başarısızlıkla neticelenebilir. Her projede az ya da çok risk vardır. Alınması gerekli tedbirler, olumsuzluk ortaya çıkmadan önce planlanmış olmalıdır.

Her Projede Aynı Planı Kullanmak

Bir şirket genellikle benzer projeler geliştirir. Kurumsal kültür ve standartlar, geliştirilen projelere göre şekillenir. Kurum, belli bir proje türünü planlamaya ve yönetmeye alışır. Farklı türde bir proje geldiğinde ilk tepki mevcut yöntemleri kullanarak planlama yapmak şeklindedir. Ancak yeni bir alana yönelik projeler, bu alana özel iş ve durumları dikkate alarak planlanmalıdır. Önceki tecrübeleri kullanırken, yeni ortaya çıkan ihtiyacı fark edebilecek ve buna çözüm üretebilecek esneklik de gösterilmelidir.

Başkalarının Planını Kullanmak

Proje planlama konunda birçok standart ve bu kitap da dâhil birçok çalışma mevcut. Ancak bu önerileri üzerine düşünmeden ve projeye özel ihtiyaçları dikkate almadan, aynen yeni projelerde kullanmaya çalışılmak doğru değildir. Çünkü her proje özeldir ve proje planı projeye özel çözümler düşünülerek hazırlanmalıdır.

Planı Güncellememek

Proje planını ilk aşamalarda yapıp sonraki aşamalarda unutulmaya terk etmek, plan yapmamak gibidir. Sadece plan için ilk aşamada harcanan zaman kaybedilmiş olur. Proje sürecinde netleşen bilgilerle plan sürekli güncellenerek, yapılan ile planlanan arasında uçurum oluşması ve kontrolün kaybedilmesi engellenmelidir.

Erken Aşamalarda Aşırı Ayrıntıya Girmek

Yazılım projesinin erken aşamalarda belirsizlik çok fazladır. Bu aşamada tüm yapılacakları ayrıntılı planlamaya çalışmak gereksiz zaman kaybına yol açar. Analiz sonunda yapılmayacağına karar verilecek bir özellik için plan yapmak faydasız bir çabadır.

Bir binanın mimari tasarımı nasıl bina kadar büyük değilse, plan da yapılacakların tüm ayrıntılarını değil sadece temel görev tanımlarını içermelidir. Tüm ayrıntıları eklemek, planı aşırı uzatır ve karmaşıklaştırır. Görev yapan kişinin şahsi girişimciliğini köreltir ve tüm sorunlara proje yöneticisinin çözüm aramasına sebep olur.

Hiç kimse çalışmasının boşa gitmesini istemez. Erken aşamalarda yapılsa da kişi, kendi yaptığı planı savunur. Bu durum gerçeğe hiç uymayan bir planı uygulamaya çağırmak gibi ilave bir problem doğurur.

Sonlara Doğru Plana Yetişebileceğini Sanmak

Yapılanlar planlanın gerisinde kaldığında, genellikle daha sonraki aşamalarda işlerin yetişeceği düşünülür. Ancak yapılan çalışmalar yazılım konusunda genellikle işlerin daha kötüye gittiğini göstermiştir [Genuchten-1991]. "*Bugün dünün yarını. Bugün ne yaptığımız yarın ne yapacağımızı belirler.*" denilmiştir. Gecikme sebeplerini tam olarak ortadan kaldırılmadan oluşan açığı kapatmak mümkün değildir.

Önceki Proje Tecrübelerinden İstifade Etmemek

Önceki projelerden ders çıkartmamak, hep benzer hataların tekrar etmesine sebep olur. Bu diğer hataların da temel kaynağıdır. Proje sürecini ölçmek için faydalı tüm değişkenler ölçülüp kaydedilmelidir. Her proje sonunda, henüz bilgiler tazeyken alınan neticeler değerlendirilmelidir.

6.7. Özet

Proje planının kalitesi, geliştirme süreci ve ortaya çıkacak ürünün kalitesi için belirleyicidir. Yüksek kalite planın anlaşılmasını, izlenmesini ve gerçekleştirilmesini kolaylaştırır. Kaliteli bir planda görevler gerçekleştirilebilir, başarı kriter ve çıktıları tanımlı, tutarlı ve tek kişinin yapabileceği seviyede bölümlenmiştir. Görev süreleri kısa ve takibi kolaydır. Plan, projeyle ilgili tüm faaliyetleri içerir. Mesai dışı çalışmalara değil, ekibin makul iş sürelerine göre yapılmıştır.

Plandaki işlemleri paralel hale getirmek, proje süresi kısaltır. Özellikle sıralı yapılması gerekmeyen işler paralel hale getirilmelidir. Fonksiyonel ve teknik ihtiyaçlarla ilgili işlemler paralel yürütülebilir. Proje modüler hale getirilirse, planlama basitleşir ve proje daha kolay takip edilebilir. Projenin alt projelere bölünmesi tasarım ve kodlama gibi birçok açıdan da ilave kolaylık sağlar. Kapalı çevrim yöntemi proje akışından bağımsız görev tanımı yapmayı önler ve görevler arası bağlantıları doğru şekilde yapılmaya yardımcı olur. Ancak bu yöntemi uygularken çok katı davranmak planı aşırı karmaşıklaştırabilir.

Planlamada ortak doğrular olduğu gibi ortak hatalar da vardır. Hataları arka plandaki etkenler dikkate alarak incelemek gerekir. Kurum ve kişiler planlamaya önem vermemek, acelecilik, işi basit görmek, tecrübe eksikliği, önceki başarıları yanlış değerlendirmek, mali şartlar, müşteri ve üst yönetim zorlaması gibi sebeplerle hata yapabilir. Hatalar, hiç plan yapmamak veya yetersiz planlamadan, her şeyin en başta planlanabileceği zannıyla planlamada aşırıya kaçmaya kadar uç noktalarda gözlenir.

6.8. Sorular

6.1) Görevle ilgili kalite faktörlerini açıklayınız.

6.2) Süreç öncelikli ve modül öncelikli planlama arasındaki farkları avantaj ve dezavantajları göz önünde tutarak sıralayınız.

6.3) Planın bütünüyle ilgili kalite faktörlerini açıklayınız.