İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAŞAM DÖNGÜSÜ YÖNETİMİ KAVRAMININ YAZILIM ÜRÜNLERİNE UYGULANMASI

YÜKSEK LİSANS TEZİ

Elif ERSOY

Anabilim Dalı : İşletme Mühendisliği

Programı : İşletme Mühendisliği

KASIM 2002

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAŞAM DÖNGÜSÜ YÖNETİMİ KAVRAMININ YAZILIM ÜRÜNLERİNE UYGULANMASI

YÜKSEK LİSANS TEZİ
Elif ERSOY
507981005

Tezin Enstitüye Verildiği Tarih : 30 Eylül 2002
Tezin Savunulduğu Tarih : 23 Ekim 2002

Tez Danışmanı : Öğr.Gör.Dr. Halefşan SÜMEN

Diğer Jüri Üyeleri : Prof.Dr. Nimet Uray

Doç.Dr. Cengiz KAHRAMAN

KASIM 2002

ÖNSÖZ

Yaşam döngüsü yönetimi kavramının yazılımlara uygulanmasının incelendiği bu çalışmanın; her aşamasında desteğini esirgemeyen eşime ve aileme, projenin oluşturulmasına ve hızla yürütülmesine katkılarından dolayı başta şirketin satınalma müdürü Sn. Nükhet Yücelay olmak üzere tüm çalışma arkadaşlarıma ve son olarak da tüm konularda göstermiş olduğu yardımlardan dolayı değerli hocam Öğr.Gör.Dr. Halefşan Sümen'e teşekkürü bir borç bilmekteyim.

Eylül 2002 Elif Ersoy

İÇİNDEKİLER

KISALTMALAR **iv**
TABLO LİSTESİ **v**
ŞEKİL LİSTESİ **vi**
ÖZET **vii**
SUMMARY **viii**

1. GİRİŞ **1**
1.1. Giriş, Çalışmanın Amacı ve Yöntemi 1

2. YAŞAM DÖNGÜSÜ YÖNETİMİ KAVRAMI VE YAZILIMLAR **2**
2.1. Yaşam Döngüsünün Aşamaları 4
2.1.1. İhtiyaçlar 4
2.1.2. Tanımlama: Gereksinimlerin Belirlenmesi 5
2.1.3. Tasarım 9
2.1.4. Gerçekleştirme 13
2.1.5. Uygulama 16
2.1.6. Bakım ve Yenileme 20
2.2. Yaşam Döngüsü Modelleri 25
2.2.1. Yap ve düzelt modeli 25
2.2.2. Çağlayan modeli 26
2.2.3. Ön örnek geliştirme modeli 29
2.2.4. Sarmal model 31
2.2.5. Çoğalan model 32
2.2.6. RAD-Hızlı Uygulama Geliştirme 33

3. YAZILIM YAŞAM DÖNGÜSÜ KAVRAMININ EKS ECZACIBAŞI KARO SERAMİK SAN. VE TİC. A.Ş. UYGULAMASI **39**

4. SONUÇ **60**

KAYNAKLAR **64**

EKLER **67**

ÖZGEÇMİŞ **91**

KISALTMALAR

YDY : Yaşam Döngüsü Yönetimi

YYD : Yazılım Yaşam Döngüsü

SGB : Sistem Gereksinimlerinin Belirlenmesi

RAD : Hızlı Uygulama Geliştirme

TABLO LİSTESİ

Sayfa No

Tablo 2.1. Yaygın belgeleme araçları..... 11
Tablo 2.2. Bakım etkinlikleri..... 21
Tablo 2.3. Hızlı uygulama geliştirmenin avantajları-dezavantajları..... 37

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1 **:** Yazılım yaşam döngüsünün altı aşaması..... 3
Şekil 2.2 **:** Veri akış çizelgesi..... 12
Şekil 2.3 **:** Sipariş giriş sistemi veri akış çizelgesi..... 13
Şekil 2.4 **:** Yap-Düzelt modeli..... 26
Şekil 2.5 **:** Çağlayan modeli 27
Şekil 2.6 **:** Ön örnek geliştirme modeli..... 30
Şekil 2.7 **:** Sarmal model..... 32
Şekil 2.8 **:** Hızlı uygulama geliştirme unsurları 34
Şekil 2.9 **:** Hızlı uygulama geliştirme fazları ve kullanılan bazı metodlar..... 36
Şekil 3.1 **:** Prismde sipariş oluşturma ekranı..... 42
Şekil 3.2 **:** Sipariş isteği ekranı- Kayıt görünümü 51
Şekil 3.3 **:** Sipariş isteği ekranı- Liste görünümü 52
Şekil 3.4 **:** Sipariş isteklerini siparişe çevirme ekranı..... 53
Şekil 3.5 **:** Siparişler ekranı- Liste görünümü 54
Şekil 3.6 **:** Sipariş isteği ekranı- Kayıt görünümü 55
Şekil 3.7 **:** Siparişler ekranı- Lojistik..... 56
Şekil 3.8 **:** Siparişler ekranı- Proforma..... 57

YAŞAM DÖNGÜSÜ YÖNETİMİ KAVRAMININ YAZILIM ÜRÜNLERİNE UYGULANMASI

ÖZET

Herhangi bir ürünün, yatırımın veya servisin canlı organizmalar gibi varlıklarını sürdürmelerinden hareketle yaşam döngüsü kavramı ortaya çıkmıştır. Maliyetlerin takibi ve kontrolünün yapılabilmesi için döngünün yönetilmesi gerekmektedir. Yazılımlara gelince, kullanılmakta ve geliştirilmekte olan her sistem için bir yazılım yaşam döngüsü söz konusudur ve bu döngünün de iyi yönetilmesi şarttır.

Yaşam döngüsünde genel kabul görmüş altı aşama vardır. Bunlar; ihtiyaçlar, tanımlama, tasarım, gerçekleştirme, uygulama ve bakım aşamalarıdır. Bu genel çerçeve içinde, bilgi sistemlerinin yaşam döngüsü modellerinden de bahsedilmelidir. Bu yaklaşımların ilki ve en yaygın biçimde kullanılanı, yaşam döngüsündeki aşamaların birbiri ardına sıralandığı, birinin çıktılarının bir sonraki adımın girdilerini oluşturduğu anlayışına dayanan Çağlayan Modeli'dir. Bu modelin gerçek yaşamda ortaya çıkardığı birtakım problemleri ortadan kaldırmak için öne sürülen Ön-Örnek Geliştirme Modeli, çağlayan modelinin bir türevi olan Artışlı Model ve yazılım mühendisliği sürecini her türlü uygulamayı içerecek şekilde açıklayan Sarmal Model, Hızlı Uygulama Geliştirme ve Yap ve Düzelt Modeli yaklaşımlardan belli başlılarıdır. Tüm modellerin kendi içinde zayıf ve kuvvetli yanları bulunmaktadır. Önemli olan bir uygulamada şirket kültürüne en uygun olanı benimseyebilmektir.

Uygulamaya konu olan şirkette kullanılan yazılımın yaşam döngüsünü tamamlamak üzere olduğu görülmüş, bakım ve yenileme ile ömrünün uzatılamayacağına karar verilmiştir. Bu çalışmada, yeni bir yazılımın şirket bünyesinde oluşturulması, ihtiyaçların belirlenmesi aşamasından itibaren tasarım, gerçekleştirme ve uygulama aşamaları adım adım anlatılmaktadır. Yeni yazılıma ilişkin ekranlara da yer verilerek, bu ekranların özellikleri üzerinde durulmuştur. Son olarak yeni yazılımın başarısı mevcut yazılımla karşılaştırılmalı olarak değerlendirilmiştir.

AN APPLICATION OF LIFE CYCLE MANAGEMENT ON SOFTWARE PRODUCTS

SUMMARY

Life cycle concept has emerged from the idea that all products, investments and services survives like living organisms. To track and control costs, it is necessary to manage the cycle. As for software products, those are in use and in development, has a life cycle which should be carefully managed.

There are six generally accepted phases of life cycle. They are requirements, defining, design, development, application and maintenance phases. The life cycle models of information systems should be considered in this context. First and most widely used life cycle model is the waterfall model. The waterfall model has consequently ordered phases and each phase's output forms the input of following phase. The prototyping model was intruced to adress some problems in waterfall model. The incremental model which is derived from waterfall model, the spiral model, rapid application development and build and fix model are also widely accepted life cycle models. All models has their own weaknesses and strength. The key is to choose the model which fits the organizational culture.

In the company, which is the subject of this thesis' last part, had found out that maintenance couldn't extend the current software's economical life. In this thesis the development of a new software product was examined, the life cycle of the software product was throughly observed from requirements analysis to application phase. Also the final product was explained by examining various screens. And lastly the new product's performance was benchmarked with the previous one.

1. GİRİŞ

1.1. ÇALIŞMANIN AMACI VE YÖNTEMİ

Yaşam döngüsü kavramı, herhangi bir ürünün, yatırımın veya servisin tıpkı canlı organizmalar gibi varlıklarını sürdürdüğü gerçeğinden hareketle ortaya çıkmıştır. İçerdikleri analiz, başlangıç, giriş, geliştirme, olgunluk, iniş ve geri çekilme adımlarını yazılımlara da uyarlamak mümkündür.

Yaşam döngüsü kavramının yazılım ürünlerine uygulanması başlıklı bu çalışmanın ikinci bölümünde yaşam döngüsünün aşamaları teker teker ayrıntıları ile anlatılmıştır. Bu bölümde ayrıca, yap ve düzelt, çağlayan, ön örnek geliştirme, sarmal ve çoğalan model gibi yaşam döngüsü modellerine de yer verilmiş ve bu modellerin zayıf ve kuvvetli tarafları irdelenmiştir.

Üçüncü bölüm uygulama bölümü olup uygulamaya konu olan şirket, EKS Eczacıbaşı Karo Seramik San. Ve Tic. A.Ş.'dir. Bu bölüm, yazılım yaşam döngüsünü tamamlamış bir sistemin incelemesini ve buna alternatif yeni yazılımın geliştirilmesini içermektedir. İnceleme şemalar ve akış diyagramları ile gösterilirken, yeni yazılımın geliştirilmesi, tüm gerçekleşen proje adımlarını içerecek şekilde ekranlar aracılığı ile anlatılmıştır. Mevcut yazılımdaki darboğazlar belirtilmiş olup proje ekibinin yeni yazılım ihtiyacını belirlemesinden tasarımına ve uygulanmasına dek olan süreç bu bölümde aktarılmıştır.

Dördüncü ve son bölüm ise sonuç bölümüdür. Bu bölümde, EKS'de yapılan çalışmanın başarısı üzerinde durulmaktadır. Yaşam döngüsünü tamamlamış bir yazılımın yerine yeni bir yazılımın uygulanması sürecindeki temel adımlar gözden geçirilmiş ve yeni yazılımın gelişmeye açık yönleri vurgulanmıştır.

2. YAŞAM DÖNGÜSÜ YÖNETİMİ KAVRAMI VE YAZILIMLAR

Yaşam döngüsü yönetimi, bir ürün, yatırım veya servisin yaşam döngüsü maliyetlerinin takibi ve kontrolü için geliştirilmiş bir kavramdır. Gelişmiş maliyet yönetimi tekniklerini, performans ölçümlerini ve yatırımlar portföyü kavramını bünyesinde barındırır. Amacı yatırımın yaşamı boyunca daha fazla kullanılabilir bilginin sağlanmasıdır.

Yaşam döngüsü yönetimi (YDY) yaklaşımının ögelerini yaşam döngüsü modeli, performans kriterleri, maliyet yönetimi sistemleri ve portföy teorisi oluşturur. Yaşam döngüsü modeli, yaşam döngüsünün 7 adımını konu alır. Herhangi bir endüstrinin, yatırımın veya ürünün hayat döngüsünü analiz, başlangıç, giriş, geliştirme, olgunluk, iniş ve geri çekilme adımları oluşturur. Performans kriterleri kritik başarı faktörlerine bağlı olup sürekli izlenir. Maliyet yönetimi, maliyetleri aktiviteler bazında değil süreçler bazında izler ve son olarak portföy teorisi sayesinde yatırımlar kendi başlarına değil bir bütünün parçaları olarak analiz edilir (Adamany ve Gonsalves, 1994).

YDY, bir yatırımın yaşamının farklı bölümlerinde farklı kazançlar sağladığı varsayımını kabul eder. Bu sava göre yatırımın değerlendirilmesi yaşam döngüsünün her safhasında farklı olmalıdır. Bu uygulama ile yatırımın her safhasında farklı performans kriterleri belirlenir ve kaynak planlaması yapılır.

YDY'nin en çok uygulandığı alanlar aşağıdadır:

1. Ürün Yaşam Döngüsü Yönetimi (Stratejik) :Ürünün tasarımından, piyasadan çekilmesi aşamasına kadar olan adımları stratejik açıdan inceler. Kar maksimizasyonu birincil önceliktir.

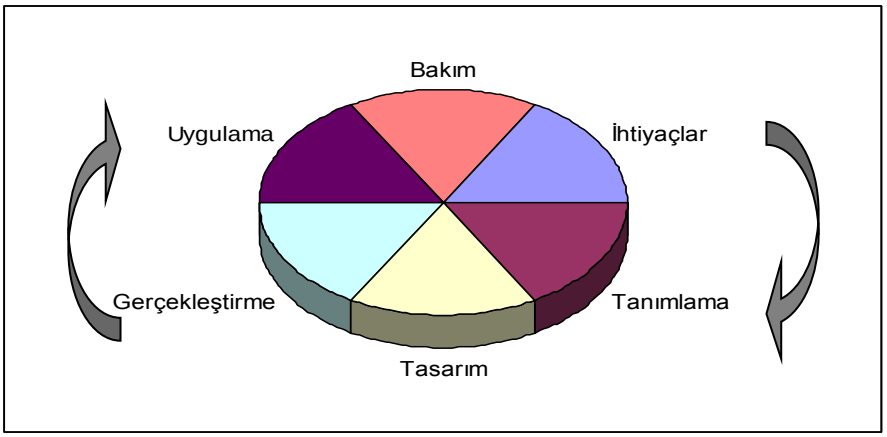
2. Ürün Yaşam Döngüsü Yönetimi (Ekolojik) :Ürünün tasarımından, kullanımının son bulması aşamasına kadar olan adımları ekolojik açıdan inceler. Çevre güvenliği birincil önceliktir.

3. Yatırım yada Varlık Yaşam Döngüsü Yönetimi: Yatırım ve varlıkların ihtiyaç saptamasından, hurdaya ayrılmasına kadar olan adımları inceler. Yatırımın ekonomik ömrü birinci önceliktir.

4. Yazılım Yaşam Döngüsü Yönetimi: Yazılım ihtiyacının saptanmasından, kullanmama kararına kadar olan adımları inceler.

Yazılım yaşam döngüsü, bilgisayara dayalı bilişim sistemi veya alt sistemlerinin bakımında izlenen ve gelişme gösteren bir süreçtir. Bu süreç bilgisayar ile birlikte süren geleneksel bir işletim sürecidir. Bu görevin gerçekleştirimi konusunda bilgisayar bilimciler arasında bir anlaşma vardır. Sadece bazı adımların sayısı ve isimlendirilmesi konusunda ayrılıklar olmaktadır. Her yetkili bu süreci farklı bir şekilde açıklama eğilimindedir. Yakından bir inceleme yapılırsa, tüm açıklamaların aynı genel deseni açıkladığı görülür ve bu desen sistemler yaklaşımını çok yakından izler.

Bilgisayar kullanan firmalarda bir çok YYD ortaya çıkar. Bunların sayısı 100 veya daha fazlada olabilir. Kullanılmakta ve geliştirilmekte olan her sistem için bir YYD söz konusudur. Örneğin, bordro YYD, nakit akışı YYD, satış raporlama YYD, ve diğerleri bunun örnekleridir. YYD'lerinin her biri, firmanın stratejik planında bilişim kaynakları için bir deyimdir. Gerçekte, YYD'leri yönetimin stratejik planda yürüttüğü araçlardır.



Şekil 2.1. Yazılım Yaşam Döngüsünün Altı Aşaması

2.1. YAŞAM DÖNGÜSÜNÜN AŞAMALARI

Şekil 2.1'de de özetlendiği gibi, bazı yorumlara göre yaşam döngüsünde altı aşama vardır:

(1)İhtiyaçlar

(2)Tanımlama

(3)Tasarım

(4)Gerçekleştirme

(5)Uygulama

(6)Bakım

Bu altı aşama YYD işlemleri bütününü tanımlamada kullanılır. Sistem geliştirme bazen aylar, bazen de yıllar sürebilir. Bu aşamalara bir çok insan katılırken büyük maliyetlerde gerektirebilir. Altıncı aşama olan bakım ve yenileme aşaması zaman olarak en uzun süren aşama olabilir. 25-30 yıl boyunca kullanılan sistemler olabildiği gibi birkaç ay ya da birkaç hafta sonra bırakılmış sistemlere de rastlanmaktadır. Kullanım aşamasında, sistemi işletimde tutmak için kesinlikle bakım yapılmalıdır. Bazı durumlarda sisteme bakım uygulamak yerine, yönetim sistemini yeniden geliştirilmesi daha pratik bulunabilir. Bu durumda döngü yeniden başlar.

2.1.1. İhtiyaçlar

YYD, yazılım için ihtiyacın oluşması ile başlar.Bu aşama ihtiyacın ortaya çıktığı birim tarafından üst yönetime bildirilmesi ve onaylanması ile son bulur. Organizasyonun iç prosedürlerine göre değişen bu aşama kimi zaman direkt BT departmanına yapılan yazılım isteği ile son bulduğu gibi, kimi zaman da organizasyondaki onay sürecinin karmaşıklığından haftalar sürebilir. İhtiyaçların onay süreci ihtiyacın niteliğine göre de değişim gösterebilir. Zira basit bir raporlama ihtiyacı için kısa bir onay süreci gerekirken, karmaşık fonksiyonları yerine getirecek bir uzman sistem için onay süreci uzayabilir.

2.1.2. Tanımlama: Sistem Gereksinimlerinin Belirlenmesi

Sistem gereklerinin belirlenmesi, ya da daha yaygın olarak kullanılan adlarıyla, sistem çözümleme ya da sistem analizi, bilgi sistemlerinin geliştirilmesinde, hem süre, hem de işgücü bakımından en fazla yatırımı gerektiren aşamadır. Konuyla ilgilenen bütün araştırmacılar arasında, ortaya çıkacak bilgi sisteminin sağlamlığının, güvenilirliğinin ve genel olarak projenin planlanan sürede ve başarılı olarak sonuçlanmasının, başarılı bir sistem çözümleme aşamasının sonucu olduğunda görüş birliği vardır. (Blackburn, 1996)

Bilgi sistemi geliştirmede hız ve üretkenlik konusunda 1992 ve 1993 yıllarında ABD, Avrupa ve Japonya'da yapılan bir araştırmada ilginç sonuçlar alınmıştır.

Araştırma kapsamında, ABD ve Japonya'da 49 yazılım üreticisi kuruluş yetkilileriyle yapılan görüşmeler, Avrupa'da ise 12 değişik ülkede 98 yazılım üreticisi tarafından yanıtlanan anketler değerlendirilmiştir. Görüşleri alınan yöneticiler, boyları 1200 satırla 6 milyon satır arasında değişen ve 27 farklı programlama dilinin kullanıldığı yazılım projelerinde görev almış kimselerdi.

Araştırma, başlıca üç soruyu yanıtlamayı hedeflemekteydi:

1. ABD, Japonya ve Avrupa'lı bilgi sistemi üreticileri, yazılım projelerinin yönetiminde nasıl farklılıklar göstermektedir?

2. Yöneticiler, yazılım geliştirme hızını nasıl yükseltebilirler?

3. Yöneticiler, yazılım geliştirmede çalışan personelin üretkenliğini nasıl artırabilirler?

İlk soruya ilişkin olarak özetle şu sonuçlar elde edilmiştir:

Her üç ülke/kıta'daki yazılım projelerinde, proje süresinin yaklaşık 1/6'sı proje planlama ve sistem çözümleme etkinliklerine ayrılmaktadır. Japonya'da kodlama (program yazımı) aşamasına ayrılan süre, toplam sürenin yaklaşık %22'si düzeyindeyken Avrupa'da bu oran %29'a çıkmakta, buna karşılık Japonya'da planlama ve sistem çözümlemeye verilen emek (işgücü) toplam işgücünün %26'sı düzeyindeyken Avrupa'da %22 dolayında gerçekleşmektedir. Bu noktada, projenin ilk aşamasındaki %4'lük bir emek fazlasının, sonraki aşamalarda hem gerekli işgücünde hem de sürede azalmaya yol açtığı söylenebilir(Blackburn, 1996).

Bütün yöneticiler, projelerin başarılı olmasındaki en önemli etmenleri: (1)nitelikli sistem çözümleme, (2)proje ekibi içinde etkili ve başarılı iletişim (belgeleme, vb.) ve (3)nitelikli insangücü olarak saymaktalar. Öte yandan yine bütün yöneticiler, kullanılan yazılım geliştirme araçlarının ürün geliştirme süresi üzerinde belirleyici bir etkisi olmadığı görüşündeler. Başka bir deyişle, üreticilere yapılan yatırımın, bilgi sistemleri projelerinde üretim araçlarına yapılan yatırımdan daha etkili bulunduğu görülmektedir.

Bu araştırmada ortaya çıkan en önemli sonuç, projenin ilk aşamalarında ihtiyaçların belirlenmesine yapılan yatırımın, sonraki aşamalarda yapılacak değişiklik ve düzeltmeleri azalttığı, genel olarak proje süresini kısalttığı olmuştur.

Ortaya çıkan bir başka sonuç ta, sistem çözümleme aşaması dışındaki aşamalar için, daha küçük ekiplerin daha verimli çalıştıklarıdır. Belli bir sürede üretilen yazılım ürününün boyutları bakımından, çözümleme aşamasına daha fazla emek sarf eden kuruluşların, genel üretkenliklerinin tutarlı olarak daha yüksek olduğu görülmüştür.

Bilgi sistemi geliştirmede ön örnek yaklaşımının kullanılması da üretkenliği artırırken proje süresini azaltmaktadır. Bunun, öncelikle, proje sürecine "müşteri"nin katkısını artırmak yoluyla gerçekleştiği düşünülmektedir. Bu sonuç, sistem çözümlemeye verilen emeğin ve ayrılan sürenin, "maliyet"ten ziyade "yatırım" olarak düşünülmesi gereğini vurgulamaktadır. İleride anlatılacağı üzere ön örnek yaklaşımı, sistem gereklerinin belirlenmesinde doğrudan doğruya kullanıcının katkısından yararlanılmasını sağlamaktadır.

2.1.2.1. Sistem Gereksinimlerinin Belirlenmesi (SGB) Raporu Öğeleri

SGB raporunun içeriği:

Uluslararası düzeyde en yaygın kabul görmüş SGB rapor standardı, Elektrik ve Elektronik Mühendisliği Enstitüsü (IEEE) tarafından önerilmiştir (IEEE Software Engineering Standards, 1988).

Bu standardın dışında, çeşitli ulusal ve uluslararası kuruluşların önerdiği rapor yapıları kullanılmaktadır.

Genel olarak SGB raporunun, anlaşılır, kesin ve eksiksiz biçimde yazılım ihtiyaçlarını ortaya koyması gerekir. "Müşteri" ile en yoğun iletişimin gerçekleştiği aşamanın sonunda yazılmış bir belge olmasıyla, bilgi sistemi geliştirme sürecindeki diğer belgelerden ayrılır. SGB raporu, müşteriyle yazılımcının teknik düzeyde yaptıkları anlaşmayı belgeler. Bu nedenle, müşteri ya da kullanıcı tarafından bütünüyle anlaşılabilmelidir. Bu, teknik olmayan bir dille yazılması anlamına gelmez. Bazı konularda kesin ve eksiksiz bilgi aktarımının yolu bazı biçimsel ve teknik terimlerin kullanılmasından geçer. Bu durumda müşteriye ya da kullanıcıya düşen, kendisi için hazırlanan sistemin ne yapacağını rahatça anlayıp onaylayabileceği ya da değişiklik isteyebileceği biçimde ayrıntıların üzerinden gitmektir.

Nitelikli bir SGB raporu nasıl olmalıdır?

Yüksek nitelikli bir SGB raporunun özellikleri şöyle sıralanabilir:

Doğruluk, tek anlamlılık, bütünlük, doğrulanabilirlik, tutarlılık, anlaşılabilirlik, geliştirilebilirlik, izlenebilirlik, ulaşılabilirlik. Bu özelliklerin her biri aşağıda kısaca gözden geçirilecektir.

- **Doğruluk:** Doğruluk, sistem gerekleri belirtiminin, kullanıcı ihtiyaçlarını doğru olarak, eksiksiz ve fazlasız olarak ortaya koyma özelliğidir. Bu özelliğin sağlanmasında, kullanıcıya büyük sorumluluk düşer, çünkü ne isteyip ne istemediğini en iyi bilecek, yine odur.

- **Tek anlamlılık:** Tek anlamlılık, gerekler belirtiminin yalnızca tek biçimde ve doğru olarak anlaşılabilme özelliğidir. Örneğin "Saniyede 1 Kbit hızında üretilen bilgi okunmalıdır." gibi bir cümle tek anlamlı değildir, çünki üretim hızının mı yoksa okunma hızının mı 1Kbit/sn olmasının istendiği anlaşılamamaktadır.

- **Bütünlük:** Bütünlük, ya da eksiksizlik, SGB raporunun, kullanıcı ihtiyaçlarının tümünü içerme özelliğidir. Bu belge, ekleriyle birlikte, tek başına, daha sonraki tasarım aşamasına yeterli dayanak oluşturabilmelidir.

- **Doğrulanabilirlik:** Doğrulanabilirlik, kullanıcı ihtiyaçlarının belirsiz ve soyut biçimde değil, kesin, ölçülebilir biçimde ortaya konulma özelliğidir. Örneğin "veriler büyük bir hızla okunmalıdır" cümlesinde ortaya konulan istemin karşılanıp karşılanmadığı hiçbir zaman doğrulanamaz. Ne kadar hız "büyük bir hız"

dır? Oysa ki “veriler, en az saniyede 1000 harf hızıyla okunmalıdır” gibi bir belirtim kesindir ve doğrulanabilir.

- **Tutarlılık:** Tutarlılık, SGB raporunun bir yerinde yazan ile bir başka yerindekinin çelişmeme özelliğidir. Ortaya konulan ihtiyaçların her biri, bütün diğer ihtiyaçlarla, belgenin bütününde belirtilen amaçlar, kapsam, ayrıntılı ihtiyaçlar,vb ile tutarlı olmalıdır. Bunu sağlamanın yolları arasında, gereksiz yinelemelerden kaçınmak, her konuyu yalnızca tek yerde ele almak vardır.

- **Anlaşılabilirlik:** Anlaşılabilirlik, diğer niteliklerin yanısıra, doğru ve açık bir dilin kullanılması demektir. Kimi uygulamalarda matematiksel bir dil, kimilerinde doğal diller, ör. Türkçe, İngilizce, Fransızca, vb , kimi zaman çizimler, vb kullanılabilir. Ancak, SGB raporu, yalnızca teknik uzmanlara değil, doğrudan doğruya bilgi sisteminin kullanıcılarına da hitap eder. Bu nedenle, onların da anlayacağı ve onay verip vermemede rahatça, güvenle karar verebilecekleri bir dille hazırlanmış olmalıdır. Bilgi sistemleri çalışmalarındaki başarısızlıkların çok büyük bir bölümünün, kişilerin dil becerilerindeki yetersizlik olduğu, yazılım mühendisliği alanının belli başlı yazarlarınca defalarca dile getirilmiştir.

- **Geliştirilebilirlik:** Geliştirilebilirlik, bilgi sistemlerinin en bilinen özellikleri arasında bulunan değişkenliğin, başarısızlığa yol açmaması bakımından önemlidir. SGB raporunda belirtilen ihtiyaçların zaman içinde değişikliklere uğraması olağandır. Bu nedenle, SGB raporunun da kolayca değiştirilebilmesi, gelişmelere açık olması gerekir. Bunu sağlamanın yolları arasında, örneğin, her konuyu yalnızca bir tek yerde ele almak, yinelemelerden kaçınmak, ele alınan her konuya çeşitli anahtar sözcükler üzerinden ulaşılmasını sağlayarak dizinlere yer vermek, vb sayılabilir.

- **İzlenebilirlik:** İzlenebilirlik, SGB raporunun incelenmesi sırasında, ya da daha sonraki aşamalarda, tasarıma temel olarak, ya da üretilen sistemin kullanıcı ihtiyaçlarını ne ölçüde karşıladığının irdelenmesinde, bu raporun her bir bölümüne, çok çeşitli mantıksal zincirler üzerinden ulaşılması gerekebilir. Örneğin her bir tasarım öğesinin SGB raporundaki hangi maddeye karşı geldiğinin kolayca bulunabilmesi gerekir.

- **Ulaşılabilirlik:** Ulaşılabilirlik, SGB raporundaki her türlü bilginin, doküman içinde kolayca bulunabilmesi özelliğidir. Bunun için en geçerli yöntem,

dizin(ler) kullanılmasıdır. Raporda geçen her terim, her kişi adı, her örgütsel birim, her dış kuruluş, vb. ayrı ayrı dizinlerde sıralanarak rapor içinde hangi sayfalarda geçtikleri gösterilirse, raporun ulaşılabilirlik niteliği yükselecektir.

2.1.2.2. Biçimsel yöntemler

Bilgi sistemleri geliştirme projelerinin başarısının belki de en önemli koşulu, kullanıcı ihtiyaçlarının doğru olarak anlaşılması ve tanımlanmasıdır. Buna yönelik olarak, SGB raporu üzerinde alınabilecek önlemlere daha önce değindik. Doğal dillerle (Türkçe, İngilizce, Fransızca, vb.) yapılan iletişimde, tek anlamlılık, belirsizlik, eksiksizlik, tutarlılık, ancak iletişimde bulunanların dikkat ve özeniyle sağlanır. Öte yandan, bu özellikleri zorunlu kılan, hata yapılmasına olanak vermeyen biçimsel diller de vardır. Örneğin bilgisayar programlama dilleri böyledir. Siz programı sözdizim kurallarına uygun biçimde yazarsanız, bilgisayarın bunu "yanlış anlaması" söz konusu değildir. Biçimsel diller, tutarsızlığı, çok anlamlılığı, belirsizliği önler. Bu açıdan bakıldığında, bütün bilgi sistemleri gerekleri biçimsel dillerle tanımlanabilse, bu projelerde başarı düzeyinin çok yükseleceği söylenebilir. Ancak, doğal dillerin rahatlığı, yaygınlığı ve herkes tarafından kullanılabilme özelliği, elbette biçimsel dillerde yoktur. Bu nedenle de her türlü gereksinim tanımının, sistem betimlemesinin biçimsel dillerle yapılması, en azından bu gün için olanaksızdır. Genel kabul görmüş bir yaklaşım, sistem tanımlarının yanlış anlaşılması maliyetinin çok fazla olacağı durumlarda biçimsel yöntemlerden yararlanılmasıdır.

Biçimsel betimleme yöntemlerinin belli başlıları arasında Sonlu Durum Makinası , karar çizelgeleri ve ağaçları, Tanımlama ve Betimleme Dili (Rockstrom ve Saracco, 1982), PAISLey (Zave, 1982) ve Petri Ağları vardır.

2.1.3. Tasarım

Bu bölümde, bilgi sistemlerinin, belirlenmiş ihtiyaçları yerine getirmek üzere nasıl tasarlandığı üzerinde durulacaktır. Tasarım, işlevsel (ya da işleyişe, davranışa ilişkin) ihtiyaçların, yapısal (ya da kuruluş ve örgütlenmeye ilişkin) bir sistem tanımına dönüştürülmesidir. Başka bir deyişle, ihtiyaçlar belirtimi, ya da SGB raporu, kurulacak sistemin neyi, nasıl yapması gerektiğini ortaya koyar. Sistem tasarımı ise bu sistemin yapısını, parçalarını, parçaların birbirleriyle ilişkisini, kısacası sistemin nasıl kurulacağını gösterir.

Bilgi sistemi geliştirmede genellikle ulaşılamayan ideal durum, sistemin, gereksinimler belirtiminden, türetilmesidir. Başka bir deyişle, tasarımın akılcı biçimde otomatik ve matematiksel olarak ortaya konulması, hata olasılığını en aza indirecektir. Ancak bunun olanaklı olduğu söylenemez. Bu olanaksızlığın başlıca nedenleri şunlardır:

i. Gereksinimler, hiçbir zaman tam değildir.

ii. Gereksinimler, hiçbir zaman ek destek gerektirmeden kullanılabilecek durumda değildir – yazılım araçları, biçimler, çevre sistemler standart olmayıp değişiklik gösterirler.

iii. Tasarımcıların zihinsel kapasiteleri, çoğu zaman sistemin bütün gereklerini bir arada kavrayamaz.

iv. Gereksinimler zaman içinde değişir.

v. Önyargılar, uzmanlık düzeyleri, gizli ve açık tercihler, hevesler, vb. tasarım sürecini kesinlikten uzaklaştırıp insan hatalarına yol açar.

vi. Sıfırdan başlayıp her şeyi yeniden üretmek yerine elde var olan (alt-) sistemleri kullanma çabası, tasarımı yönlendirir.

Öte yandan, akılcı biçimde yürütülmesi öngörülen bir tasarım sürecinin tanımlanması aşağıdaki bakımlardan yararlıdır:

i. İdeal süreç, tasarımcılara, işe nasıl girişecekleri konusunda yol gösterir.

ii. Sistemin bütün özelliklerini gözönünde tutma çabası, birçok özelliğin birlikte dikkate alınmasını sağlar.

iii. İdeal süreç, tasarım sürecinin aşamalarını ortaya koyar, böylece gelişmenin izlenmesi, gecikmelerin fark edilmesi ve üretilen tasarımın değerlendirilmesi kolaylaşır (Parnas ve Clements, 1987).

Tasarım aşamasının sonucunda ise bir tasarım raporu oluşturulur. Tasarım raporu içeriği ise, yine kesin, açık ve tutarlı biçimde, kurulacak sistemin yapısını tanımlar. Bilgi sistemi tasarım raporu içeriği şu alt başlıkları içermelidir:

i. Birimler rehberi – Sistem birimleri arasındaki sıradüzen ilişkisini ortaya koyacak bir ağaç biçiminde, birimlerin birbirini çağırma (denetim) ilişkisi gösterilir. Bu rehberde, her birimin görev ve sorumlulukları kısaca belirtilir.

ii. Arabirimler – Her birimin *kara kutu* olarak dış dünyayla ilişkisi belirtilir; her birimin kullanıcıları (o birimi çağıran/işleten birimler) için gereken kullanım bilgileri verilir.

iii. Yazılım tanıtımı – Tek tek her birimin çağırılış biçimi, çağrı komut yapısı gösterilir; her çağrının ortaya çıkartacağı sonuçlar, veri yapıları üzerindeki etkileri; her çağrının başka birimler üzerindeki etkileri; zamanlama ve duyarlık düzeyleri; hata ve diğer özel durumlarda ne olacağı tanımlanır. Birim yapılarının tanımlanmasında, yine birimler rehberi yapısı kullanılabileceği gibi, her birim için veri yapıları ve bunlar üzerinde her çağrının etkileri tanımlanabilir.

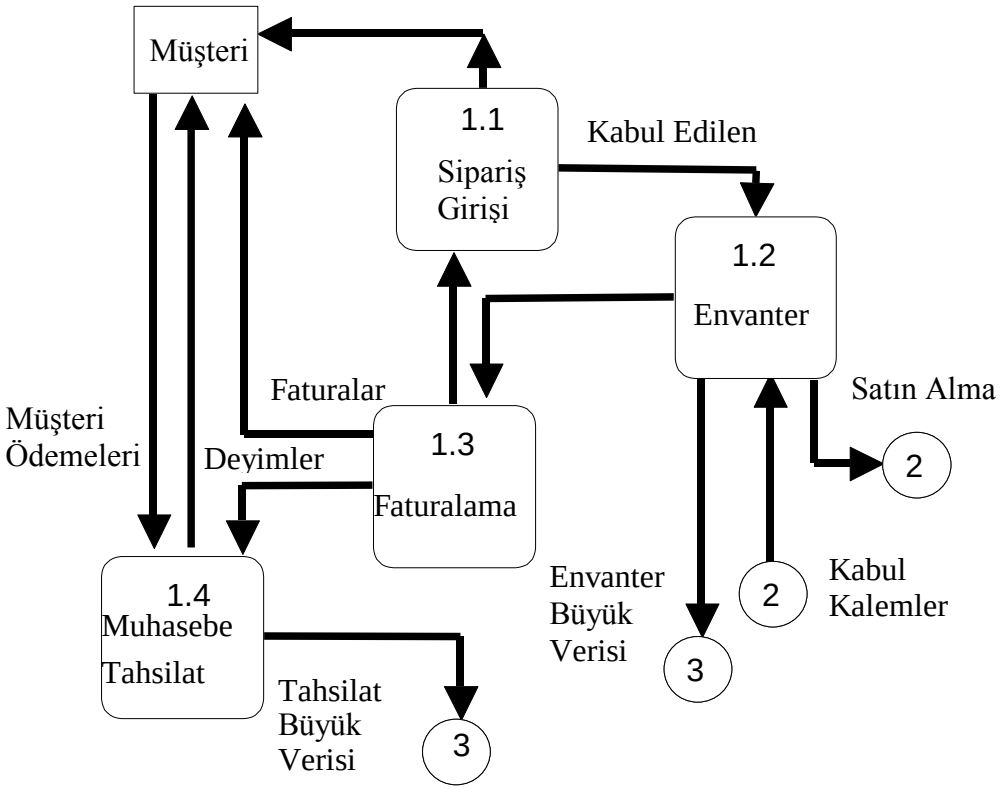
2.1.3.1. Yapısal ve işlevsel tasarım yöntemleri

Bu aşamada sistem analistleri, kullanıcılar ile birlikte çalışır ve yeni sistem tasarımını aşağıda açıklanan yaygın tasarım araçları ile belgeler.

Tablo 2.1. Yaygın Belgeleme Araçları

	İşlemleri Belgeleme	**Veri Belgeleme**
Grafik Araçları	Sistem Akış Şeması	Varlık İlişki Çizgesi
	Veri Akış Şeması	Yazıcı Uzayı Çizgesi
	Warnier-Orr Çizgesi	Kayıt Anahat Çizgesi
		Veri Yapısı Çizgesi
Anlatımsal Araçlar	Yapısal İngilizce	Veri Sözlüğü
	Karar Mantık Çizgesi	

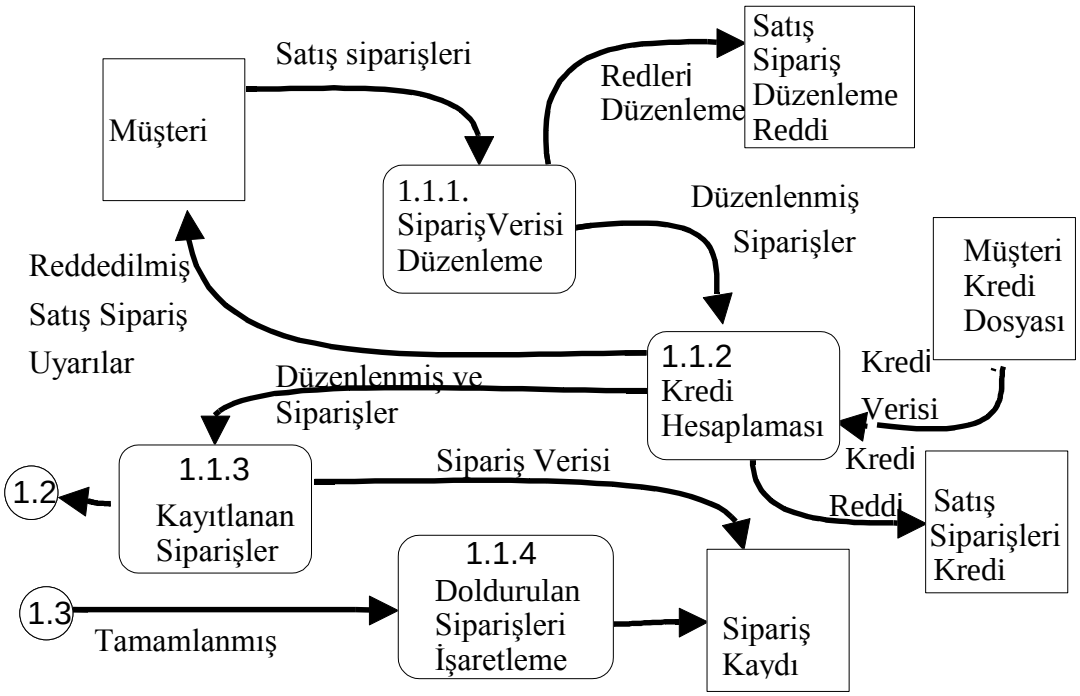
Aygıtların bazıları, çözümleyicinin belgelemeyi yukardan aşağı bir biçimde yapmasını sağlar. Burada büyük bir resimle başlanır ve giderek küçük ayrıntılara inilir.Yukarıdan aşağı yaklaşım, yapısal tasarımın bir özelliğidir. Burada bir sistemden bir alt sistem düzeyine inilir (Thorp, 1998). Aşağıdaki şekiller bu yukarıdan aşağı işlemi göstermektedir.



Şekil 2.2. Veri Akış Çizgesi (Thorp, 1998)

Şekillerdeki oklar, veri akışını gösterir ve veri sözlüğünde bir giriş ile belgelenirler. Veri Sözlüğü firmanın veritabanının kapsamının resmi bir açıklamasıdır. Veri Sözlüğü, tüm proje takımlarına, firma veri kaynağını açıklamada kullanılan genel bir dil üretir. Aşağıdaki şekil, Satış Siparişleri veri akışı için veri akış sözlüğü girişidir.

İşlemler ve verinin bu belgelenişi, hem bilgisayar sistemleri hem de bilgisayarsız sistemler için taban oluşturur.Burada bir bilgisayarın kullanılacağı varsayılmıştır (Forsberg ve diğerleri, 1996).



Şekil 2.3. Sipariş Giriş Sistemi Veri Akış Çizgesi (Thorp, 1998)

2.1.4. Gerçekleştirme

Gerçekleştirme aşaması iki bölümden oluşur: Programlama ve sınama. Aşağıda bu iki bölüm ayrıntılı olarak incelenecektir.

2.1.4.1. Programlama

Programlama, bilgi sistemi geliştirmenin görece en rahat yönetilebilen, denetlenebilen, başı sonu belli aşamasıdır. Özellikle gereksinimleri belirleme çalışması yeterince nitelikli biçimde yapılmışsa, ve sistem tasarımı yeterince ayrıntılı ve doğruysa, programlama çalışması doğrudan doğruya tasarımın hedeflenen programlama diline aktarılmasından ibarettir.

Amatör çalışmalarda sistem çözümleme ve tasarım aşamaları yeterince ciddi olarak ele alınmadığından programlama, yazılım geliştirmenin en çetrefil, belki de tek aşaması olarak görülecek, ve o zaman da başarısızlık kaçınılmaz, en azından elde edilecek başarı rastlantısal olacaktır.

Burada, önceki bölümlerde ele alınan proje adımlarının tam olarak gerçekleştirildiği varsayılır. Böyle olunca, programlama hakkında söylenecek az şey kalacaktır. Bunlar (a) dil seçimi, (b) gözden geçirmeler başlıkları altında özetlenebilir.

a. Dil seçimi: Kullanılacak dilin, bir yandan hedeflenen sistemin gereklerine uygun olması, diğer yandan sistemin üzerinde çalışacağı altyapının en etkin ve verimli biçimde değerlendirilmesine olanak sağlaması, üçüncü bir yandan da, programları yazacak ekip açısından yüksek üretkenlik ve güvenilirlik düzeylerine ulaşılabilmesi için, özenle seçilmesi gerekir. Yazılım üreticileri, yeni bir dile geçmenin getireceği eğitim ve birikim oluşturma yatırımlarının maliyeti ile yeni olanakların ve artacak etkinlik ve verimlilik düzeylerin getirisini karşılaştırarak her projede yeniden bu kararı vermek durumundadırlar.

b. Gözden geçirmeler: Bilgi sistemi geliştirme projesinin her aşamasında gözden geçirmelerin, yüksek nitelik bakımından büyük önemi vardır. Programlama aşamasında da, her birimin, sınamaya geçilmeden önce, programı yazan kişinin dışındaki bir kişi (kişiler) tarafından gözden geçirilmesinin, hata bulma ve ayıklamada çok büyük yararı olduğunu yinelemek gerekir.

2.1.4.2. Sınama

Yazılım sınama sürecinin dört temel aşamadan oluştuğu kabul edilmektedir: (1) Birim sınama, (2) Bütünleşim sonrası sınama, (3) Gereksinimler sınama, (4) Sistem (ya da etki) sınama. Aşağıda, bu aşamaların her birini kısaca ele alınacaktır. Ancak bu aşamalardan önce, sınamanın planlanması gereğini de vurgulanmalıdır.

Sınama planı, ileride incelenecek her bir sınama aşamasında ne yapılacağını, alınması gereken sonuçların ayrıntılı olarak her birini, alınacak sonuçların nasıl değerlendirileceğini göstererek sınama sürecini yönlendirir.

Sınama, doğrulama, geçerlileme gibi terimler çoğu kez birbirinin yerine geçebilmekte, bunlar arasına kimi uzmanlarca ince anlam farkları yerleştirilmekte, ancak farklı kişiler, aynı terimi farklı anlamlar için yeğleyebilmektedir. (Ör. İngilizce *validation* ve *verification* terimleri çeşitli kaynaklarda farklı anlamlarla kullanılmaktadır.) Burada yalnızca "sınama" terimini kullanılacaktır.

1. **Birim sınama:** Bu aşama, her yazılım biriminin programının ortaya çıkması sırasında "beyaz kutu" yöntemiyle, programın her işlem dizisinin en az bir kez çalıştırılması ve sonuçların beklentilerle karşılaştırılması anlamına gelir. "Beyaz kutu" yöntemi, yazılım tasarımının elde bulunduğu, programın mantık yapısının ve işlem dizilerinin bilindiği varsayımına dayanır. Hataların, programın sıkça çalışan görece olağan bölümlerinden çok, seyrek olarak başvurulan, olağandışı koşullara

karşı gelen bölümlerinde bulunma olasılığının daha yüksek olduğu bilinir. Beyaz kutu yöntemi, programın sınanmamış bölümü kalmaması için kullanılır. Program biriminin, eksiksiz olarak olası her türlü veri ile işletilmesi, baştan sona olası her işlem dizisinin denenmesi çoğu zaman pratik değildir. Ancak olası her işlem dizisinin, yani işlemlerin olası her yinelenme durumunun değilse de her işlemin en az bir kez sınanması çoğu kez olanaklıdır.

2. Bütünleşim sonrası sınama: Her birimin sınanması başarılı olarak sonuçlandıktan sonra, birimlerin bir araya getirilmesi sırasında, ortaya çıkan yazılımın tasarımla karşılaştırılması gerekir. Başka bir deyişle, bu aşamada, üretilen yazılımın doğru üretilip üretilmediği sınanır. Tasarlanan sistemle yazılan programların ne ölçüde çakıştığı incelenir.

3. İhtiyaçlar sınama: Bir önceki aşamada "üretilen yazılımın doğru olup olmadığı" incelenmişken, ihtiyaçlar sınamasına "doğru yazılımın üretilip üretilmediği" incelenir. Başka bir deyişle, bu aşamada, ortaya çıkan ürünün, bu kez ihtiyaçlar belirtimine ne ölçüde uyduğu irdelenir. Eğer sistem çözümlemede ortaya konulan her bir ihtiyaç, ölçülebilir biçimde tanımlanmışsa bu aşamada yapılacak şey yalnızca sistem gereksinimleri belgesindeki her bir maddenin üzerinden giderek yazılımın hedefleneni ne ölçüde gerçekleştirdiğini saptamaktır. Bu iş, "alfa sınama" adıyla yazılım üreticisinin ortamında ve denetimli koşullarda yapılabileceği gibi "beta sınama" adıyla, kullanıcılar tarafından, kendi ortamlarında, üreticinin denetiminin dışındaki koşullarda gerçekleştirilebilir. Genellikle, ihtiyaçlar sınaması aşamasına gelindiğinde, ortaya çıkacak eksiklik ve hataların giderilmesi için vaktin geç olduğu kabul edilir. Bu aşamada görülen sorunlar, ya bir sonraki yazılım sürümünde giderilmek üzere saptanmış olur, ya da müşteri ile üretici arsındaki görüşmelerde dikkate alınır.

4. Sistem (ya da etki) sınama: Sistem sınama, yazılımın, gerçek kullanım ortamında, bir bilgi sistemi bütünlüğü içindeki etkisinin değerlendirilmesi demektir. Program yazımı, sistem geliştirme sürecinin ne kadar tanımlı bir aşamasıysa, sistem etkilerinin değerlendirilmesi de o kadar tanımsız bir süreçtir. "Etki" olarak neyin değerlendirileceği, bu etkinin ortaya çıkmasında bilgi sisteminin payının ne olduğu, bu payın nasıl ölçüleceği, hep yanıtsız sorulardır. Bir zorluk ta, kurulup işlemeye başlayan bir bilgi sisteminin büyük ölçekli etkilerinin ancak uzun bir süreden sonra ortaya çıkacağıdır. "Etki"nin nasıl tanımlanacağı sorusuna, parasal

getiriye yönelik olarak çalışan sistemlere ilişkin olarak nasıl yanıt verileceği bellidir. "Karlılık" çoğu kez yeterince kapsamlı bir hedeftir. Ancak, toplumsal, elle tutulamayan hedeflere yönelik sistemlerde bu çok daha zordur. Örneğin ulusal ölçekteki bir sağlık bilgi sisteminde, hedef "toplumun sağlık düzeyinin yükselmesi" olarak konabilir, ancak bunun ölçütlerinin tanımlanması, bu ölçütlerin net olarak elde edilmesi, ve bu hedefe yönelik başka etkenlerden bağımsız olarak yalnızca bilgi sisteminin etkisinin ölçümü olanaksız denilebilecek denli zordur. Öte yandan, bilişim sistemleri çalışmalarının, yalnızca "en son teknolojilerin peşinden ümitsizce koşmak" olmadığını bilenler için sistem sınamasının doyurucu sonuçları da olmalıdır.

Bu noktada, belki geniş ölçekli ve kurumsal düzeydeki bilgi sistemleri için değilse de en azından yazılım birimleri düzeyinde anlamlı olabilen "biçimsel doğrulama" yöntemlerinden de söz etmekte yarar var. Bu çerçevede amaç şudur: Sistem gereksinimlerini matematiksel kesinlikle tanımlayabilirsek, bunlardan yola çıkan sistem tasarımını da aynı matematiksel kesinlikle ortaya koyabilirsek, o zaman, üretilen yazılımın hem tasarıma uygunluğunu, hem de gereksinimleri karşılama düzeyini yine matematiksel kesinlikle saptayabiliriz. Bu yalnızca yazılan programların biçimsel olarak sınanıp doğrulanmasını değil, gereksinimlerden tasarımın, tasarımdan da yazılım gerçekleştiriminin otomatik olarak elde edilmesinin de olanaklı olması anlamına gelir. Günümüzde bu alanda çok sayıda çalışma yapılmaktadır. Özellikle telekomünikasyon sistemleri, donanım tasarımı gibi altyapıları, bileşenleri net olarak tanımlanabilen alanlarda biçimsel yöntemler kullanılabilmektedir. İletişim protokollerinin ve bunları gerçekleştirecek sistemlerin tanımlanmasında, tasarımında ve sınanıp doğrulanmasında biçimsel yöntemler yaygın olarak kullanılmaktadır. Bu konuda örneğin *SDL, ESTELLE, LOTUS, Z* gibi dil ve ortamlar kullanılmaktadır. Ancak, henüz, yönetim bilgi sistemlerinde, karar destek sistemlerinde, genel olarak biçimsel/matematiksel yöntemler yerine doğal diller kullanılmakta, bu da çok anlamlılık, belirsizlik, eksiklik, tutarsızlık gibi kaçınılmaz özellikleri birlikte getirmektedir.

2.1.5. Uygulama

Gereksinimleri belirlenmiş, tasarlanmış ve gerçekleştirilerek sınanmış bir yazılım sisteminin, bilgi sistemi içinde yerini almasına ve yeni bilgi sisteminin bütünüyle kullanıma girmesine kısaca "uygulama" denir. Uygulama süreci, çoğu kez

ilgili kuruluşun bütününü, tüm çalışanlarını ilgilendirir. Canlı bir organizmanın herhangi bir ameliyat geçirmesi gibi, yerel ve önemsiz bir boyutta olabileceği gibi o organizmanın bütün yaşamsal düzeneklerini zorlayan bir süreç de olabilir.

Uygulamanın kapsadığı adımlar şöyle özetlenebilir:

1. Uygulamanın planlanması
2. Uygulama projesinin kuruluşa duyurulması
3. Uygulamadan sorumlu çalışanların örgütlenmesi
4. Donanım ve yazılım altyapısının bütünüyle hazır duruma getirilmesi
5. Veri tabanının hazırlanması
6. Etkilenecek birimlerin ve çalışanların eğitimi
7. Fiziksel ortamın (ofis, vb) hazırlanması
8. Dönüşüm (Royce, 1998).

Bu adımların hangi sırayla geçileceği, birlikte mi ardarda mı gerçekleştirileceği ya da her birinin diğerlerine göre önemi ve gerektirdiği zaman, işgücü ve diğer kaynaklar, projeden projeye değişir. Birçok projede, diğer adımlar kaçınılmaz olduğundan, yalnızca yeterince özenle ele alınmamasında büyük sakıncalar bulunan üç en yaşamsal adımın: (1) planlama, (6) eğitim, ve (8) dönüşüm adımlarının vurgulanması yeterli olur. Aşağıda bu adımların her birini kısaca gözden geçirilecektir:

1. Uygulamanın planlanması : Bilgi sistemi projesinin en başında, olurluk çalışması aşamasında, bütün proje aşamaları planlanmış durumdadır. Özel olarak uygulama aşamasının planlanması ise, en başta yapılan bu planın gereken biçimde ayrıntılandırılması anlamına gelir. Bu planda uygulama takvimi gözden geçirilir. Kuruluş birimlerinin ve çalışanlarının yetki ve sorumlulukları belirlenir. Uygulama projesi çerçevesindeki örgütlenme oluşturulur. Uygulama planında, altyapıya ilişkin bütün hazırlıkların gereken zamanda bitirilmiş olmasının nasıl sağlanacağı da net olarak ortaya konur. Yeni sistemin yürürlüğe girmesinden önce, veri tabanının nasıl oluşturacağı, buna yönelik olarak hangi kaynakların kullanılacağı belirlenir. Birçok bilgi sistemi projesinde sistem çözümleme, tasarım ve gerçekleştirme, uzun dönemde sistemin nasıl işleyeceğini tanımlar, ancak en

başta, bir kerelik de olsa, veri tabanının nasıl oluşturulacağı, eski bilgilerin yeni sistemle nasıl bütünleştirileceği saptanmaz, bu da baştan hesaplanmayan kaynak gereksinimlerine ve gecikmelere yol açar. Doğru planlama, hem projenin başlangıcında, hem de uygulama aşamasının başında, bir kerelik ve sürekli işlerin her birinin nasıl, kim tarafından ve hangi kaynaklardan yararlanılarak gerçekleştirileceğini saptamayı içerir. Bilgi sisteminin her düzeydeki kullanıcılarının eğitimi, uygulama aşamasının önemli adımlarındandır. Belki de bütün proje sürecinde en yüksek sayıda kişinin katılımı eğitim aşamasında gerçekleşir. Alt düzeydeki kullanıcılarla üst düzeylerdeki karar vericilerin bilgi sistemiyle etkileşimi farklı olacaktır. Bu doğrultuda, alacakları eğitimin de farklı olması doğaldır. Alt düzeydeki kullanıcılar, sistemin kendi işlerinde nasıl kullanılacağını, bilgi girişini, döküm almayı, hata durumlarında ne yapılacağını öğrenmek durumundadır. Kuruluş sıradüzeninde yukarılara çıkıldıkça, basit kullanıcı eğitimi yerine çalışma ve düşünme biçimini değiştirme yönü ağır basan eğitim gereksinimleri öne çıkar. Üst yöneticilere, sınıf içinde klasik tarzda eğitim vermek yerine kimi zaman seminerler, kimi zaman tartışma grupları, kimi zaman ise yemek, kokteyl vb çok daha farklı ortamlarda yapılacak görüş alışverişleri yeğlenebilir. Eğitimde sıkça karşılaşılan bir güçlük de kuruluş için her düzeyde önemli olan kişilerin eğitim almaya daha az vakit ayırabilmeleri, eğitime vakit ayırabilenlerin ise kuruluş içinde görece daha az önemli işlerle uğraşmaları çelişkisidir. Dönüşümün nasıl gerçekleştirileceği ise başlı başına dikkat ve ustalık gerektiren, çoğu zaman hataları affetmeyen bir planlama ve yönetim sorunudur. Uygulama planlaması yalnızca yöneticilere yararlı olacak planların hazırlanmasından ibaret değildir. Bu adım, uygulamanın bütün adımları için gereken yönergeleri, kılavuzları, yetki ve sorumluluk dağılımını, sorunların nasıl çözüleceğini, vb. yazılı olarak ortaya koymayı ve bütün ilgililere dağıtmayı da içerir.

2. **Uygulama projesinin kuruluşa duyurulması :** Uygulama projesinin kuruluşa duyurulması, aslında uygulama aşamasına gelinmeden önce yapılması gereken ve ustalık isteyen bir iştir. Örgütlerin genel olarak her türlü yeniliğe ve değişime karşı çıktıkları bilinen bir gerçektir. Bunu aşmanın en doğal yolu ise, her birimin ve her çalışanın, değişimden nasıl somut kazanç sağlayacağının açıklanmasıdır. Bu kazanç bireysel ve parasal kazanç olmayabilir. Ancak çalışanların herhangi bir biçimde olumlu görmedikleri bir değişime katılmaları, ona katkıda bulunmaları da beklenemez. Burada yöneticilere düşen, yeni bilgi sisteminin

her düzeyde ne gibi yararlar sağlayacağı konusunda kuruluşun tümünün aydınlanmasını sağlamaktır.

3. **Uygulamadan sorumlu çalışanların örgütlenmesi :** Uygulamadan sorumlu çalışanların örgütlenmesi, uygulama projesinin başlı başına bir proje olarak ele alınması anlamına gelir. Sorumluları, işbölümü, kaynakları, süreleri belirlenmiş bir uygulama projesinin başarı şansı bütün bunların gelişigüzel biçimde sağlandığı (ki bu hiç te seyrek değildir) durumlardan çok daha yüksek olacaktır (MIS, 2001).

4. **Donanım ve yazılım altyapısının bütünüyle hazır duruma getirilmesi :** Altyapının hazır edilmesi, çok bileşene bağımlı projelerin hemen hepsinde, oldukça ciddi bir eşgüdüm çalışması gerektirebilir. Yazılımda gecikmeler, donanım olanaklarının öngörülmemiş nedenlerle kurulamaması, vb hep bu aşamayı geciktirebilen nedenlerdir.

5. **Veri tabanının hazırlanması :** Veri tabanının hazırlanması, birçok projede, bir kez yapılan ancak kuruluşun geçmişiyle geleceğini birbirine bağlaması nedeniyle önemli olan bir iştir. Eğer proje planlamasında yalnız geleceğe yönelik sistem gereksinimleri saptanmış ve geçmişte birikmiş verilerin yeni sistemle nasıl bütünleştirileceği açıkça ortaya konulmamışsa bu aşama önemli bir sıkıntı yaratacaktır. Bu aşama için özel yazılımlar geliştirilmesi, geçici işgücü sağlanması, belki toplu veri girişinin yapılacağı ya da belgelerin taranıp veri tabanına yükleneceği özel olanaklara gereksinim olabilir.

6. **Etkilenecek birimlerin ve çalışanların eğitimi :** Eğitim, yukarıda uygulamanın planlanması başlığı altında da değinildiği gibi projenin en çok kişiyi bir araya getiren aşaması olabilir. Bu kişilerin nasıl seçileceği, eğitime ilgi göstermelerinin nasıl sağlanacağı, kısaca eğitimin başarılı olmasının nasıl sağlanacağı hep yaşamsal sorulardır. Ayrıca, özellikle ülkemizde kamu kesiminde insan gücünün yeterince değerli görülmemesi, birçok uygulamada, eğitim almış kişilerin, kuruluşu yakından tanımayan yöneticiler tarafından görevlerinin değiştirilmesine ve bu da önemli kayıplara yol açabilmektedir.

7. **Fiziksel ortamın (ofis, vb) hazırlanması :** Fiziksel ortamın hazırlanması, görece kolay, ancak yine de belli bir yönetim becerisi gerektiren bir uygulama adımıdır. Ofislerin, iletişim olanaklarının, belki bilgisayar sistem

odalarının vb hazırlanması, güvenlik sorunlarının çözülmesi hep bu adımın parçalarıdır.

8. Dönüşüm : Dönüşüm adımı, yalnızca eski sistemin (ki bu el emeğine ve bütünüyle kağıda dayalı bir sistem olabilir) çalıştığı dönemden, yalnızca yeni sistemin yürürlükte bulunacağı döneme kadar geçen bütün süreyi kapsar. Kimi basit ve etki alanı dar uygulamalarda birden bire eskiden yeniye geçilebileceği gibi, birçok uygulamada belli bir süre iki sistem birlikte çalıştırılır. Bunun getireceği işgücü ve kaynak gereksinimi yüksek olmakla birlikte kuruluşta yeni sisteme yeterli güvenin kazanılması için zorunlu olabilir. Birçok projede de birlikte çalışmadan eski sistemin devre dışı bırakılmasına geçiş, sistemin bütünü için bir defada yapılmayıp aşamalı olarak gerçekleştirilebilir. Bu aşamalar, sistemin mantıksal parçalarının birer birer dönüştürülmesi biçiminde (ör. önce satınalma alt sistemi, sonra stok denetimi alt sistemi, sonra personel alt sistemi, daha sonra muhasebe alt sistemi ve en sonda maliyet muhasebesi alt sisteminin yeniye dönüştürülmesi) olabileceği gibi, geniş bir kuruluşun çeşitli yerleşkelerinde ya da coğrafi/idari birimlerinde bire birer yeniye geçiş yapılabilir. Dönüşüm sürecinde esas, kuruluşa en az sarsıntı getirecek, çalışanların etki ve verimlilik düzeyini en az düşürecek, ancak öte yandan gecikmelerin getireceği kayıpları da en aza indirecek dönüşüm sürecinin belirlenip uygulanmasıdır (Kitaoka, 2000).

2.1.6. Bakım ve Yenileme

Bilgi sisteminin, önceki bölümlerde kısaca gözden geçirildiği gibi kullanıma girmesi, yaşam çevriminin en başına dönülmesi anlamına gelir. Uygulama ve elde edilen etkiler değerlendirilecek ve kimi zaman yenilemeler, düzeltmeler gerekebilecektir. Günümüzde, ortalama yazılım ömrünün beş yıldan az olduğu kabul edilmekte, dünya çapında pazarlanan yazılım ürünleri, hepimizin bildiği gibi bir iki yılda bir yenileriyle değiştirilmektedir.

Bu bölümde, önce bilgi sistemlerinde bakımın anlamı ve türleri üzerinde durulacaktır. Daha sonra, yazılımda değişiklik kararına yakından bakarak, en son olarak ta bakım işinin örgütlenmesi, değerlendirilmesi ve tüm bu süreçte kullanıcılara düşen sorumluluk konularını incelenecektir.

Yukarıda, bilgi sistemleri ile iş süreçlerinin karşılıklı olarak birbirleri üzerinde belirleyici etkileri olduğunu vurgulanmıştı. Bilgi sisteminin bakım sürecinde de bu karşılıklı etkilerin belirleyici olması kaçınılmazdır.

2.1.6.1. Yazılım Bakımı Nedir?

Ne denli iyi sınanmış, doğruluğu ve güvenilirliği kanıtlanmış olursa olsun, yazılımın tümüyle kusursuz olarak kullanıma sunulması neredeyse olanaksızdır. Özellikle, tek (ya da az sayıda) kullanıcı kuruluş için özel olarak geliştirilmiş, çok geniş bir pazarda kendini kanıtlamamış bir üründe zaman içinde ortaya çıkacak kusurların bulunması doğaldır. Bunların giderilmesine düzeltici bakım denilir.

Öte yandan, kullanım süresince kullanıcı gereksinimlerinin değişmesi de doğaldır. Kuruluşlar yaşayan sistemlerdir, ve değişim kaçınılmazdır. Çevre koşulları (müşteriler, yasal mevzuat, vb) değişir, kuruluş yeni alanlara girer, vb nedenlerle yazılım değişiklikleri gerekir. Bunların gerçekleştirilmesine uyarlayıcı bakım adı verilir.

Özellikle başarılı olan, pazarda geniş kabul gören yazılım türlerinde uygulanan bakım ise üçüncü türü oluşturur. Pazar payını genişletmek, yazılıma yeni yetenekler eklemek, başarım (performans) düzeyini yükseltmek, yeni altyapılar üzerinde çalışmasını sağlamak vb amaçlarla yapılan bakıma da yetkinleştirici bakım adı verilir.

Yazılım dünyasında bakım etkinliklerinin aşağıdaki gibi dağıldığı bilinmektedir (Blackburn, 1996):

Tablo 2.2. Bakım etkinlikleri

Bakım Türü	**Genel İçindeki Yüzdesi**
Düzeltici	21
Uyarlayıcı	25
Yetkinleştirici	50
Diğer	4

Yazılımda hata bulma ve düzeltme maliyetinin yaşam çevrimi boyunca katlanarak arttığı da bilinmektedir. Bu artış, hataları süreç içinde olanaklı en erken zamanda bulmanın önemini açıkça ortaya koymaktadır. Öte yandan, yazılımın "bakıma uygunluk" niteliği, maliyetleri düşürmede etkili olacaktır. Yazılım nitelikleri bakımından, iyi belgelenmiş ve birimsellik niteliği yüksek ürünlerin bakımının görece kolay olacağı da açıktır.

Bakım işinin bir yönü de kendi kendini güçleştirici olmasıdır. Yazılım sistemleri bakım gördükçe bakıma uygunluk nitelikleri azalır. Aynı sistemin bakım maliyeti zamanla yükselir. Özellikle düzeltici ve uyarlayıcı bakım, çoğu kez bunalım koşullarında gerçekleştirildiğinden enine boyuna irdelenmemiş değişiklikler getirmekte, bunların sonuçları da uzun dönemde ortaya çıkabilmektedir. Bunu engellemenin yolu, bakım sürecinin biçimsel ve sağlam kurallara bağlanmasıdır.

2.1.6.2 Bakım Süreci

Yazılımda değişiklik kararı : Değişiklik kararıyla ilgili olarak şu soruların sorulması gerekir:

1. Değişiklik yapılması zorunlu mu?
2. Değişimin toplam maliyeti nedir? Bunun hesaplanmasında bütün sistem geliştirme süreci dikkate alınmalı, gereksinimlerin belirlenmesi, tasarım, geliştirme, sınama, dönüşüm, ve onun kapsadığı eğitim gibi etkinlikler gözardı edilmemelidir.
3. Değişim için gereken kaynaklar nelerdir? Zaman, işgücü, donanım, yazılım altyapıları, vb.
4. Toplam değişim süresi ne kadar olacaktır?
5. Hizmet ne süreyle duracaktır?
6. Gerekecek kullanıcı eğitiminin çapı, kapsamı, süresi, kuruluş üzerindeki etkisi ne olacaktır?
7. Yazılımın genel niteliği ve bakılabilirliği nasıl etkilenecek? Bu değişim, gelecekte gerekecek değişimleri nasıl etkileyecek?
8. Sistemin kalan ömrü nedir?

9. Eski sistem bütünüyle mi ortadan kaldırılacak, yoksa kısman yürürlükte mi kalacak?

Bu sorulara verilecek yanıtlar, bilinen geleneksel "örgütsel tutuculuk" yönünde, değişime karşı güçleri desteklemekten çok, değişimin ne zaman ve hangi kapsamda yapılması gerektiğini belirlemede yararlı olacaktır. Bu değerlendirmenin ışığında çoğu kez, dar kapsamlı ve "önemsiz" görülen değişikliklerin gerçek çözüme götürmediği, köklü ve kapsamlı değişimlerin gerektiği ortaya çıkabilecek, kuruluşun bunun için gereken planlamayı ve yatırımı gerçekleştirmesi sağlanabilecektir.

Bakım Adımları : Bakım sürecinde kullanıcı, kullanıcı-çözümleyici-yazılımcı üçgeninin önemli bir köşesini oluşturmaktadır. Özellikle düzeltici ve uyarlayıcı bakımda süreci başlatan doğrudan doğruya kullanıcılardır. Bu nedenle, yukarıda yazılımda değişiklik kararına ilişkin olarak ortaya konulan soruların, kullanıcının kendisi tarafından yanıtlanmasıyla işe başlamakta yarar vardır.

Yazılım bakımı, çoğu zaman yazılımı geliştiren kişilerce gerçekleştirilir. Tam da bu nedenle, hep "ikinci iş" ya da "ikinci sorumluluk" olarak yapılır. Uzmanlar, genellikle yeni sistem geliştirmeyi, eskisi üzerinde çalışmaya yeğlerler. Öte yandan, eski yazılım üzerinde yapılan bakım çalışmalarının, yazılımcıların üretkenliğini düşüren belli başlı etmenler arasında olduğu da kabul edilmektedir (Genuchten, 1991).

Oysa ki bakım, yaşayan bir sistemi ayakta tutmak anlamına gelir ve birçok durumda, yeni sistem geliştirmekten çok daha etkin ve verimli bir çözüm olabilir. Bu nedenle, bakım sorumlularının belirlenmesi, bunu olanaklar elverdiğince "yan" iş olarak değil "asıl" iş olarak yapmalarının sağlanması, sorumluların yanlarında yedek personelin de görevlendirilmesi, bakım sürecinin niteliğini yükseltecektir.

Bakım süreci, aşağıdaki aşamaları içerir:

a. Bakım istemi – genellikle kullanıcı tarafından ortaya konur. Bir sorundan yakınma ya da bir gereksinimi ortaya koyma biçiminde olabilir. Acil – olağan – önemsiz biçiminde sınıflandırılabilir.

b. İstemin değerlendirilmesi – bakım isteminin, önemine göre belirlenecek bir süre içinde, bakım sorumluları ve kullanıcı (temsilcileri) tarafından değerlendirilerek, yukarıdaki soruların yanıtlanması ve bir karara varılmasını içerir.

Bu karar, “derhal değişiklik” – “bir sonraki sürüm için değişikliğin planlanması” – “hiç bir şey yapılmaması” vb biçimlerinde özetlenebilir.

c. Değişikliğin yapılması – gerekli karar verilip kaynaklar sağlandığında değişiklik işlemi gerçekleştirilir. Bu, yazılım yaşam çevriminin tümünü içerecektir. Gereksinimlerin belirlenmesi, tasarım, gerçekleştirme, sınama, dönüşüm, eğitim, ve her aşamada gereken bütün belgeleme adımları, değişiklik çalışmasını oluşturur.

d. Değişikliğin değerlendirilmesi – yine tüm yazılım yaşam çevriminde olduğu gibi, uygulama sonrasında, yapılanların etkilerinin değerlendirilmesiyle bakım süreci tamamlanmış olur.

Bakım Değerlendirmesi : Bakım değerlendirmesinde kayda geçecek ilk bilgi kullanıcı değerlendirmesidir. Bakım sürecini başlatan istemin sahipleri yeni işlevsellikten hoşnutsa, sorunsuz çalışabiliyorsa başarının ilk koşulu sağlanmış demektir.

Bakım sürecinin niteliği, yine tüm yazılım süreci gibi, yalnızca ortaya çıkan ürünle ölçülmez. Değişiklik sürecinde izlenen yöntem, gözden geçirme ve diğer nitelik öğeleri, belgeleme, sorumluların bulunabilirliği, personelin varlığı, ulaşılabilirliği, yedeklenmesi ve beceri düzeyleri hep süreç niteliğini belirleyen öğelerdir.

Bakıma harcanan zaman ve emek istatistikleri sürecin somut ve sayısal olark değerlendirilmesini sağlar.

Bakım öncesindeki ve sonrasındaki kullanıcı değerlendirmeleri ise, bu sürecin niteliksel yönünü ortaya koyacaktır. Bir birimin istekleri doğrultusunda yapılacak değişiklik, bazen, kuruluşun diğer birimlerinin olumsuz etkilenmesine yol açabilir. Bu ise, ilk karara yönelik değerlendirmenin yetersizliğini gösterir. Daha da kötüsü, yazılımın bir işlevindeki değişiklik, başka işlevlerde, giderek daha önemli ve belki de istenmeyen değişikliklere yol açabilir.

Bu nedenlerle, bakım sürecinin, belki de bütün yazılım yaşam çevriminden daha önemli yönlerinin bulunduğu da öne sürülebilir. En azından, bakımın nitelikli ve yeterli personel tarafından yapılması, değişiklik kararının, ancak yeterli bir değerlendirmeden sonra, niteliği yüksek bir süreç içinde oluşturulması, başarının yollarını oluşturacaktır (Boehm ve Ross, 1989).

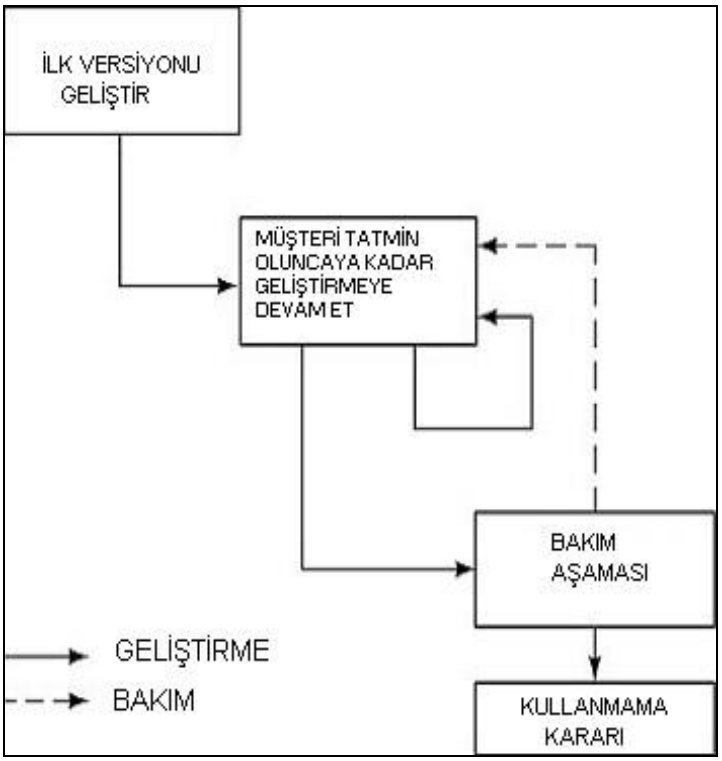
2.2. YAŞAM DÖNGÜSÜ MODELLERİ

Bu bölümde, bilgi sistemlerinin yaşam döngüsü modelleri incelenecektir. Her sistem gibi bilgi sistemlerinin de yaşamından, evriminden, doğuş, gelişme, dönüşüm ve yeniden oluşumundan söz edilebilir. Bu sistemlerin yaşam döngüsü kabaca gereksinimlerin tanımlanması, sistemin tasarlanması ve kurulması, uygulamada yeni gereksinimlerin ortaya çıkması ve yeniden tasarım aşamasına dönüş olarak özetlenebilir. Aşağıda, bu genel çerçeve içinde, özellikle bilgi sistemi oluşturulmasının projelendirilmesinde izlenen belli başlı yaklaşımlar gözden geçirilecektir. Bunların ilki ve en yaygın biçimde kullanılanı, yaşam döngüsündeki aşamaların birbiri ardına sıralandığı, birinin çıktılarının, bir sonrakine girdi oluşturduğu anlayışına dayanan çağlayan modelidir (waterfall model). Çağlayan modelinin gerçek yaşamda ortaya çıkardığı birtakım sıkıntıları çözmeye yönelik olarak yaygın biçimde kullanılan ön örnek geliştirme (prototyping), ve bu yaklaşımların hepsini birlikte tanımlayan ve yazılım mühendisliği sürecini her türlü uygulamayı içerecek biçimde açıklayan sarmal modeli (spiral model) de ileride ele alınacaktır. Tüm bu modellere ek olarak modelsizlik olarak tanımlanabilecek yap ve düzelt(build&fix) ve çağlayan modelinin bir türevi olan artışlı (incremental) model de incelenecektir.

2.2.1. Yap ve Düzelt Modeli (Build & Fix)

Bu model daha önce de belirtildiği gibi modelsizlik olarak tanımlanabilir.

Yazılım, ihtiyaçlar belirtimi doğrultusunda hazırlanır ve müşteri tatmin oluncaya kadar geliştirilir. Aşağıda sistemin grafik gösterimi görülebilir.

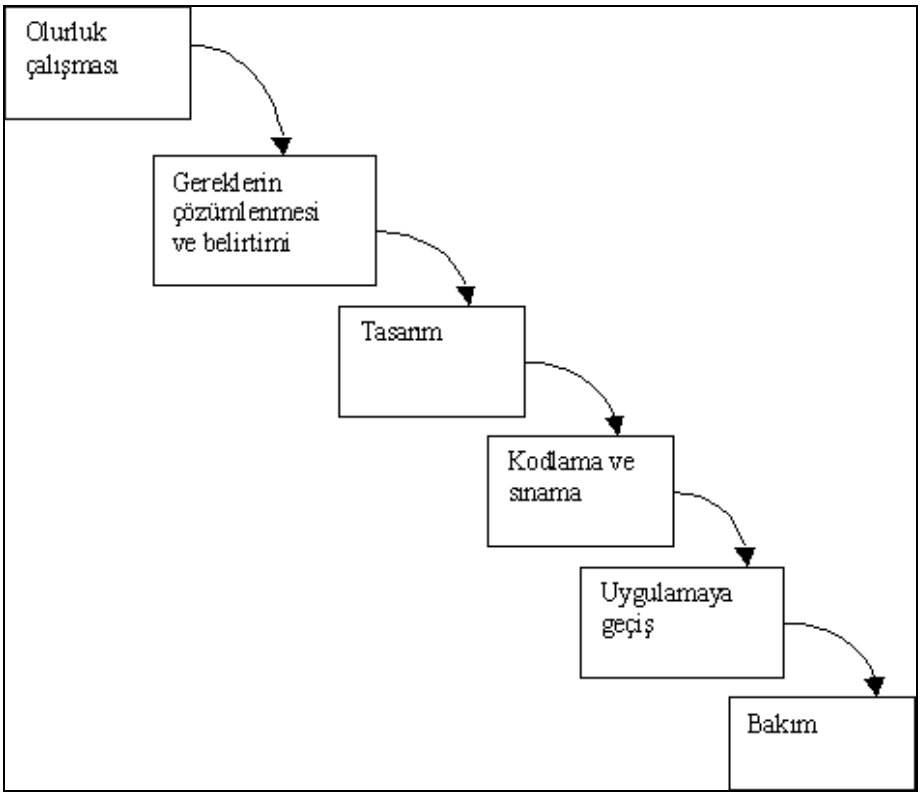


Şekil 2.4. Yap-Düzelt Modeli (Ling, 2002)

Şekilden de görüleceği üzere bu modelde hiçbir analiz ve dizayn aşaması yoktur. Müşteri tatmin oluncaya kadar gerçekleşecek çok uzun bir geliştirme döngüsü söz konusudur. Döngünün çok uzun olmasını ve beraberinde getirdiği yüksek maliyet nedeniyle "yap ve düzelt" modeli genellikle tercih edilmez. Çok küçük çaplı yazılım isteklerinde ise uygulama kolaylığı nedeniyle tercih edilebilir.

2.2.2. Çağlayan Model (Waterfall Model)

1970'li yıllarda yaygınlaşan çağlayan modeli, basit ve tek boyutlu oluşuyla, günümüzde de bilgi sistemleri geliştirmede özellikle proje sözleşmelerine temel oluşturmakta yaygın olarak kullanılmaktadır. Projenin her aşaması, çağlayanın bir düzeyi olarak görülür, süreç, ilk düzeyden son düzeye doğru geriye dönüş olmadan ilerler. Aşamalar ve düzeyler, farklı uygulamalarda farklı biçimlerde bölünebilir, ancak bu modelin belirgin özelliği, geriye dönüşlerin öngörülmemesidir. Şekilde, tipik bir çağlayan modeli örneği görülmektedir.



Şekil 2.5. Çağlayan Modeli (Ling, 2002)

Çağlayan modelinin, bilgi sistemi geliştirme projelerinde nasıl sözleşme çerçevesi olarak kullanılabileceği açıktır. İşin sahibi ile projeyi geçekleştiren kuruluş arasında, hangi aşamanın sonunda hangi ara ürünlerin teslim edileceği, bunların nasıl denetleneceği, vb hükme bağlanır, hangi aşamalarda ne kadar ödeme yapılacağı kararlaştırılır ve proje başlar. Burada önce, bu aşamaların her birinde kısaca söz edeceğiz, sonra da bu modelin olumlu ve olumsuz yanlarına kısaca göz atacağız.

Olurluk çalışması, ciddi projelerin hepsinde zorunlu görülen, yapılacak işin temel gerekçelerinin ortaya konulduğu, "neredeyiz?", "nereye ulaşmak istiyoruz?", "bunun çeşitli yolları nelerdir?", "farklı seçeneklerin maliyetleri ve getirileri nelerdir?", "girişilecek projenin temel varsayım, yöntem, aşama ve çıktıları neler olmalıdır?", "girişilecek projenin maliyeti ve süresi nasıl öngörülmektedir?" gibi soruların tartışıldığı, projeye girişilip girişilmeyeceği konusunda verilecek karara esas belgenin hazırlanması demektir.

Gereklerin çözümlenmesi ve belirtimi, bilgi sistemi projelerinin ilk ve en önemli aşamasıdır. Bu aşamada, kurulacak sistemin kullanıcılarından, ayrıntılı olarak istemleri, sistemin hangi amaçlarla ve nasıl kullanılacağı öğrenilir. Yalnızca kullanıcılar tarafından dile getirilen istekler değil, kullanıcının temel strateji ve

hedefleri, bu hedeflere onu en doğru, etkili ve verimli biçimde ulaştıracak araçlar çözümlenir. Bu aşama, bilgi sistemleri projelerinin en belirleyici aşamasıdır. Kullanıcılar çoğu kez gerçek ve somut gereksinimlerini tanımlayamaz, en etkili çözümler için istemde bulunmazlar. Piyasanın sunduğu çözümler, sağdan soldan edindikleri duyum ve izlenimler, kişisel beklenti ve hedefleri, çoğu zaman, kurumsal gereksinimleri gölgeler. Ayrıca kullanıcı istemleri zaman içinde değişir. Projenin başında öne sürülen istemler, ortaya çıkan ürün o istemleri yüzde yüz karşılasa bile projenin sonraki aşamalarında farklı biçimde dile getirilir. "…Ama bizim istediğimiz bu değildi ki!", ya da "…bu sistem hiç kimsenin işine yaramaz!", ya da "…siz bizim istemlerimizi hiç anlamamışsınız!…" gibi yakınmaların duyulması, ne yazık ki bilgi sistemleri projelerinde hiç te seyrek değildir. Bu tür başarısızlıkları en aza indirmek için öncelikle gereksinimleri çözümleyen ekibin işinin ustası olması, yüksek düzeyde iletişim becerisi göstermeleri, ortaya konulan bilgilerin sistematik biçimde belgelenmesi, yanlış anlamaları önlemek için elden gelen duyarlığın gösterilmesi gerekir. Özellikle karmaşık mekanik, elektronik, vb denetim sistemlerinde, gereksinimlerin eksiksiz ve tutarlı biçimde ortaya konulabilmesi için çok sayıda biçimsel betimleme dili geliştirilmiştir. Yazılım mühendisliği konusundaki temel kaynaklar, bu aşamaya yapılacak insangücü ve süre yatırımının, bilgi sistemleri projelerinin başarısında doğrudan doğruya yansıdığında, bu aşamada yapılan hataların sonraki aşamalarda giderilmesinin zor ve pahalı olacağında birleşirler (Finkelstein ve diğerleri, 1991).

Tasarım aşaması, gerekleri belirtilen sistemin kurulması için gereken araç ve altyapıların biraraya getirilmesidir. Bilgisayar programlarının tasarlanması, kullanılacak insan – makine sistemlerinin girdi, çıktı ve süreçlerinin ortaya konulması, kurumsal örgütlenmede ve / ya da işleyişte yapılacak değişikliklerin tanımlanması, hep bu aşamada yapılır. Bilgi sistemlerinde nitelik güvencesinin önemli öğeleri olan belgeleme ve gözden geçirme tasarım aşamasında da büyük önem taşır.

Çağlayan modelinde tasarımın ardından, uygulama geliştirmeye, yani kodlama ve sınama aşamasına geçilir. Eğer gereksinim belirtimi ve ona dayalı olarak geliştirilen tasarım yeterince olgunsa ve nitelikli bir belgelemeyle ortaya konulmuşsa, bu aşama, kullanıcılarla en az etkileşim gerektiren, en "teknik"

aşamadır. (Ön örnek geliştirme yöntemi ise bunun tam tersine, üretilen her ara ürünün doğrudan doğruya kullanıcının değerlendirmesine sunulmasını öngörür.)

Uygulamaya geçiş, bir anlamda, "kapalı kapılar ardında" geliştirilen sistemin, gerçek kullanıma sokulmasıdır. Uygulama yönergelerinin hazırlanması, sistemin her bir parçasının bu dönüşümdeki rolünün belirlenmesi, belki bir süre eski ve yeni sistemin birlikte – yan yana – yürütülmesi, vb. hep bu aşamanın içinde yer alır.

Bakım ise, yaşayan her sistemde olduğu gibi, bilgi sistemlerinde de ortaya çıkabilecek sorunların giderilmesi, yeniliklere yanıt verilmesidir. Öngörülmemiş uygulama biçimleri, artan ya da azalan iş hacimleri, ortaya çıkabilecek mevzuat değişiklikleri, donanım teknolojisinde oluşan değişiklikler, çevre ve altyapıdaki yeniliklere uyum gereksinimi, hep sistem bakımının konuları arasındadır.

Çağlayan modeline yöneltilen en temel eleştiri, bu modelin yapay bir doğrusallık sergilemesine ilişkindir. Başka bir deyişle, sanki her şey ilk yapıldığında doğru yapılırmış gibi, gereksinimler belirlendikten sonra tasarıma geçilmekte ve bir daha gereksinim belirleme aşamasına dönülmemektedir (Landay, 2001). Oysa ki gerçekte defalarca geri dönüş zorunluluğu vardır. Tasarım "tamamlandı" denilir, uygulamaya geçilir, ama görülen aksaklıklar, gereksinim belirtimini değiştirmese bile belki tasarımı baştan ele almak söz konusu olabilir. Gerçekte her aşamanın içinde ve aşamalar arasında geri dönüşler kaçınılmazdır.

Bu modelin ikinci zayıf noktası ise, gereksinim çözümleme ve tasarım aşamalarında, proje süresinin yarısı boyunca ortaya, kullanıcıyı (ya da "müşteri"yi) doyuracak hiçbir ürün çıkmamasıdır. Bu "fazla" uzun hazırlık dönemi, bir yandan sistem başarısı için zorunlu görülmekte, ancak müşteriler açısından, "sonuçsuz" yatırımlar anlamına gelmektedir. Hazırlanan raporların, gereksinim ya da tasarım belirtim raporlarının, müşteri açısından kullanım değeri yoktur. Çünkü müşteri, somut yazılım, somut bilgi giriş çıkışı peşindedir (Landay, 2001).

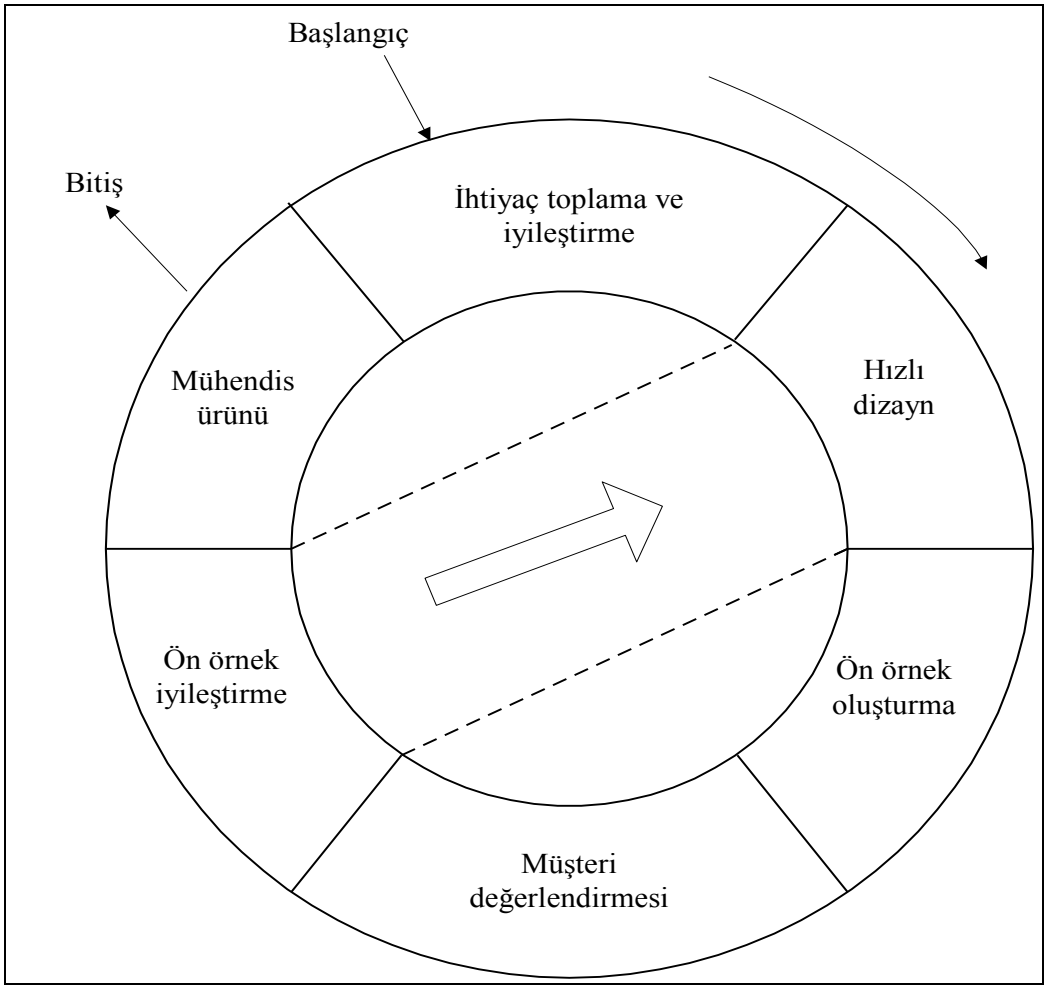
2.2.3. Ön Örnek Geliştirme Modeli

Sıklıkla, müşteri yazılımın genel hatları belirler, fakat girdi, işlemler ve çıktı konusunda ayrıntılı olarak bilgi sağlamaz. Bunun yanı sıra, geliştirici algoritmanın

kesinliğinden, işletim sisteminin uygunluğundan emin olamaz. Bu tür durumlarda ön örnek modeli en iyi yaklaşımdır.

Ön örnek yaratma, geliştiricinin yapacak olduğu yazılımın bir modelini(örneğini) yaratması işidir.

Şekil 2.6'da bu modelin ayrıntıları gösterilmektedir. Diğer yaklaşımlar gibi, bu modelde gereksinimlerin belirlenmesi ile başlar. Bundan sonra hızlı tasarım aşaması başlar, bundan sonra ön örnek yaratma gelir. Bu örnek müşteri tarafından değerlendirilir ve iyileştirmeler başlar. Bu öteleme, müşterinin gereksinimlerinin tam olarak karşılandığı anda son bulur.



Şekil 2.6. Ön Örnek Geliştirme Modeli

İdeal olarak ön örnek yaratma, yazılımın gereksinimlerini belirleme de kullanılır. Çalışan bir ön örnek yaratılırsa, geliştirici çalışan programı kullanılır hale hızlı bir şekilde getirmeye çalışır.

Çağlayan modeli gibi, bu modelde bazı durumlarda problemli olmaktadır. Bunlar:

1. Müşteri çalışan modeli gördüğünde, bu ön örneğin ne işe yarayacağını bilmez. Bu yaklaşımın gereklerini bilmediği için problemler ile karşılaşılır.

2. Geliştirici çalışan bir ön örneğe sahip olmak için, bazı uzlaşmalarda bulunabilir. Uygun olmayan bir işletim sistemi veya algoritma, iyi bilindiği için kullanılabilir. Zaman geçtikçe, geliştirici bu seçimlerle uzlaşıp, neden diğer seçimlerin yapıldığını unutabilir (Landay, 2001).

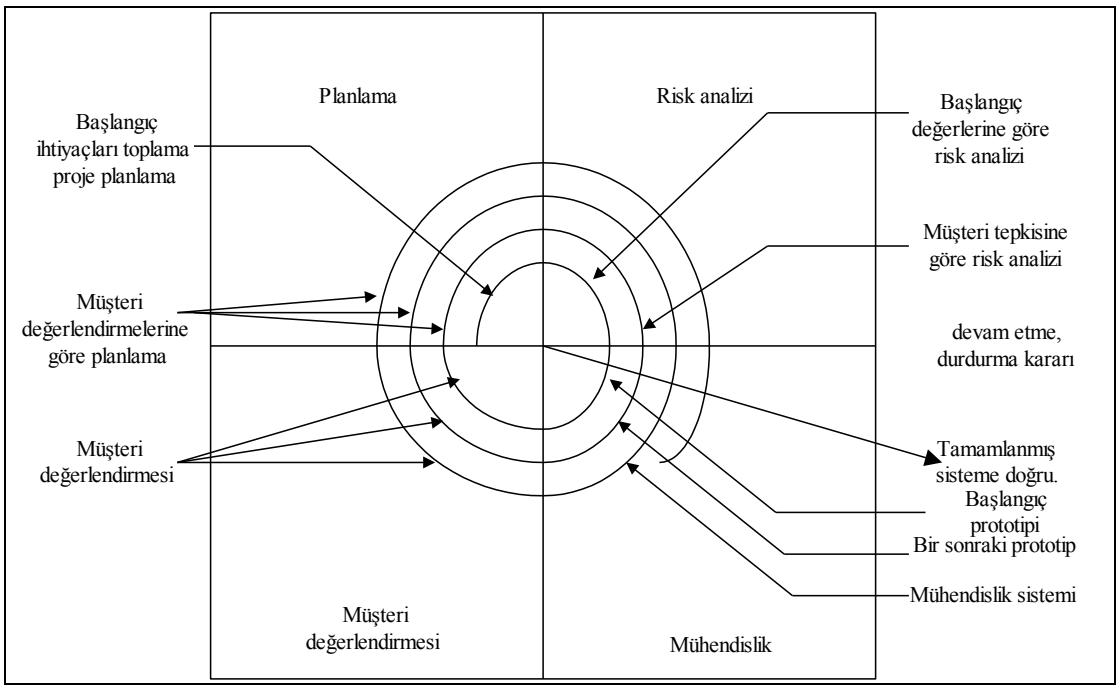
Problemlerle karşılaşıldığı halde, prototip modeli yazılım mühendisliği için etkin bir yaklaşımdır. Yöntemin iyi uygulanmasının anahtarı, oyunun kurallarının baştan belirlenmesidir. Müşteri ile baştan, bu modeli gereksinimlerin belirlenmesine yardımcı olduğu belirlenmelidir.

2.2.4. Sarmal Model (Spiral Model)

Yazılım geliştirme yaklaşımlarından olan bu model, klasik yaşam döngüsü ve prototip modelinin en iyi özelliklerinin birleştirilmiş halidir. Ek olarak, bu birleşime risk analizi eklemiştir. Şekil 2.7, bu modeli göstermektedir.

1. Planlama-hedeflerin, alternatiflerin ve sınırlamaların belirlenmesi.
2. Risk analizi-alternatiflerin ve risk tanımlalarının analizi.
3. Mühendislik-bir sonraki düzey ürünün geliştirilmesi.
4. Müşteri değerlendirmesi-mühendisliğin sonuçlarının değerlendirilmesi.

Spiral modelin numarası şekil 2.7'de düzenlenmiş merkezden çevreye doğru yapıda saklıdır. Spiralin çevresindeki ilk devre üzerinde, hedefler, alternatifler ve sınırlamalar tanımlanır ve riskler tanımlanıp analiz edilir. Eğer risk analizinde, bir problem varsa, prototip yaratma mühendislik kısmında yaratılır ve bu prototip geliştirici ile müşteriye yardımcı olur. Benzeşim veya diğer modeller, daha sonraki problem tanımlama ve gereksinimlerin iyileştirilmesinde kullanılır.



Şekil 2.7. Sarmal Model

Müşteri, mühendislik işini değerlendirir ve önerilerde bulunur. Bu veriler göre, tekrar planlama ve risk analizi aşamaları gerçekleştirilir. Risk analizinin sonucuna göre, projeye devam edilir yada durdurulur.

Spiral model yaklaşımı, çok büyük hacimli işlerin geliştirilmesinde kullanılan en gerçekçi yaklaşımdır. Spiral model, projenin her aşamasındaki teknik risklerin üzerinde durur. Eğer iyi uygulanırsa, problem haline gelmeden riskleri indirger (Boehm,2000).

Bu model de diğer modeller gibi her derde deva değildir. Her müşteriyi bu modelin kontrollü olduğuna ikna edebilmek zordur. Modelin uygulanması, risk kestirim deneyimi gerektirir. Eğer büyük bir risk fark edilmez ise, problemler şüphesiz olarak oluşur. Sonuç olarak bu model yeni bir modeldir, diğer iki model kadar sıklıkla kullanılmamaktadır.

2.2.5. Artışlı Model (Incremental Model)

Pratikteki ihtiyaç ve uygulamalardan doğan bu model bir varsayım hariç çağlayan modelinin benzeridir. Çağlayan modelinde tüm uygulama süreci tüm ihtiyaçların karşılandığı bir yazılımın uygulanması şeklinde iken artışlı modelde ise

yazılım yapı adı verilen fonksiyonel parçalara bölünerek uygulanır. Yapıların belirlenmesi ise sistem gereksinimlerinin belirlenmesi aşamasında müşterinin ihtiyaç duyduğu fonksiyonları kritiklik açısından değerlendirmesiyle olur (Ling, 2002). En kritik yapılar ilk olarak geliştirilip, uygulamaya konulur. Ardından tüm gereksinimler karşılanıncaya kadar yeni yapıların geliştirilmesi ile geliştirme-uygulama süreçleri arasında bir döngü oluşur. Bu esnada uygulamaya alınmış yapılar ise geliştirme-uygulama döngüsüne paralel olarak bakım sürecine geçerler. Artışlı model müşteri tatmini açısından oldukça başarılı bir modeldir. Müşteri bir çok kritik olmayan fonksiyon için beklemek zorunda kalmaz. Bu modelin eksik yönü ise uygulanabileceği durumların modüler olarak modellenebilir durumlarla sınırlı olmasıdır. Pratikte neredeyse her fonksiyon diğer fonksiyonlarla etkileşim içerisinde olduğundan modüler yapılara rastlanma sıklığı enderdir (Basili ve diğerleri, 1986).

2.2.6. Hızlı Uygulama Geliştirme (RAD)

Hızlı uygulama geliştirme (RAD) nedir?

RAD temelde klasik yazılım geliştirme metotlarının bir uygulamayı çok uzun zamanda ortaya çıkartmasına (hatta kimi zaman uygulama ortaya çıktığında, değişen iş ortamıyla beraber ihtiyaçlarında değişmesi sonucu, uygulamanın ihtiyaçları karşılayamamasına) bir tepki olarak ilk defa James Martin tarafından 1991'de ortaya atılmıştır. RAD temellerini Barry Boehm'in spiral modelinden alır. Spiral modeline göre iki temel farkı vardır. Spiral modelinde bir döngü içinde belirlenen tüm ihtiyaçlar karşılanır. Spiral organizasyonda yeni ihtiyaçlar oluştukça devam eder. RAD'de ise her döngü ihtiyaçların ancak bir kısmını karşılar. İkinci temel fark spiral modelinde zaman değişken bir kısıt iken RAD'de ise sabittir (Cavaye, 1995).

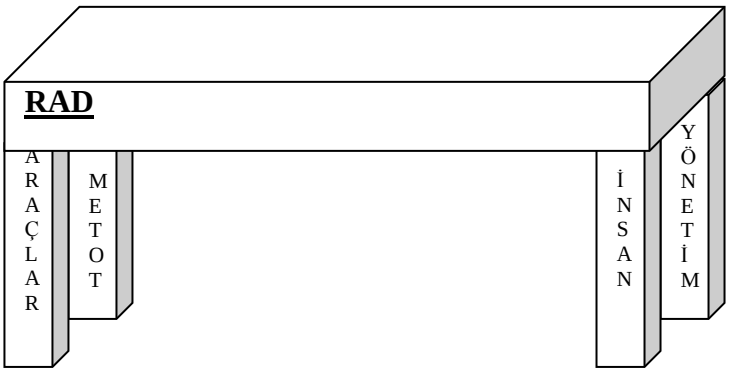
Zaman kısıtının sabit olması aslında RAD'in en belirleyici özelliğidir. Organizasyon ihtiyaçları öncelik sırasına göre belirlendikten sonra proje için bir takvim belirlenir. Eğer takvime yetişememe gibi bir risk ortaya çıkarsa takvim güncellenmez. Aksine önceden belirlenen ihtiyaçlardan düşük önceliği olanlar uygulamanın kapsamından çıkarılarak takvim yakalanır. Bu sayede klasik metotların %20 ila %50'si bir sürede ihtiyaçların %70 ila %90'ı karşılanır (Martin,1991). Ortaya çıkan yazılım organizasyonda uygulanmaya başlandıktan sonra kalan ihtiyaçların ve yeni ihtiyaçların karşılanacağı bir yazılım geliştirmek için proses yeniden başlar.

Hızlı uygulama geliştirmenin (RAD) unsurları nelerdir?

RAD'in 4 temel unsuru vardır:

- Araçlar,
- Metot,
- İnsan,
- Yönetim.

Bu unsurlardan birinin eksikliği yada yetersizliği uygulama geliştirme sürecini yavaşlatır.



Şekil 2.8. Hızlı uygulama geliştirme unsurları (Hart,2002)

Araçlar: 4. Nesil programlama dilleri, CASE (Bilgisayar destekli yazılım mühendisliği) araçları gibi yazılım araçları RAD'in araçlar unsurunu oluşturur. Bu araçlar yardımı ile hızlı bir şekilde prototipler oluşturulur. RAD'in inşa fazını çabuklaştırır. 4. nesil programlama diline verilecek en iyi örnekler MS Visual Basic, Borland Delphi, Oracle Forms'dur. CASE araçlarına örnek olarak Oracle Designer gösterilebilir (The Gov. of HongKong, 2002).

Metod: RAD'sürecinin her bir fazında en etkili metotlar kullanılmalıdır. Kişiler arasındaki komünikasyon en hızlı ve etkin biçimde gerçekleştirilmelidir. Süreçler arasında bilgi akışını sağlayacak dokümantasyon da en hızlı ve etkili bir şekilde olmalıdır. Tüm bu unsurları optimize edecek metotlar RAD sürecinde kullanılır. Burada dikkat edilmesi gerek unsur bu metotlar süreçlerin hızlı işlemesi için informal komünikasyon ve dokümantasyon kullanabilirler (Hart, 2002).

İnsan: RAD'in belki de en önemli unsurudur. Geliştirme takımının bilgi sistemleri kanadı diğer uygulama geliştirme metotlarının aksine birden fazla

uzmanlığa sahip olmaları gerekir. Çünkü iletişim için geçen süreyi azaltmak, RAD'de yazılım kalitesinden daha önemlidir. Burada bahsedilen yazılım kalitesi, kodun daha hızlı ve az yer kaplaması gibi teknik unsurlardır. Bu sayede bir geliştirme takımı örneğin 4 sistem analisti, 4 programcı ve 2 veri tabanı tasarımcısı içerceğine, 4 çoklu uzmanlığa(Multi talented) sahip kişi ile gereksinimleri karşılarken iletişim için geçen zamanı büyük ölçüde azaltmış olur. İletişim için harcanan zaman kişi sayısı ile geometrik olarak arttığı için bazı durumlarda 1 kişilik tasarruf iletiş zamanını %90 azaltabilir (Koh ve Heng, 1996)

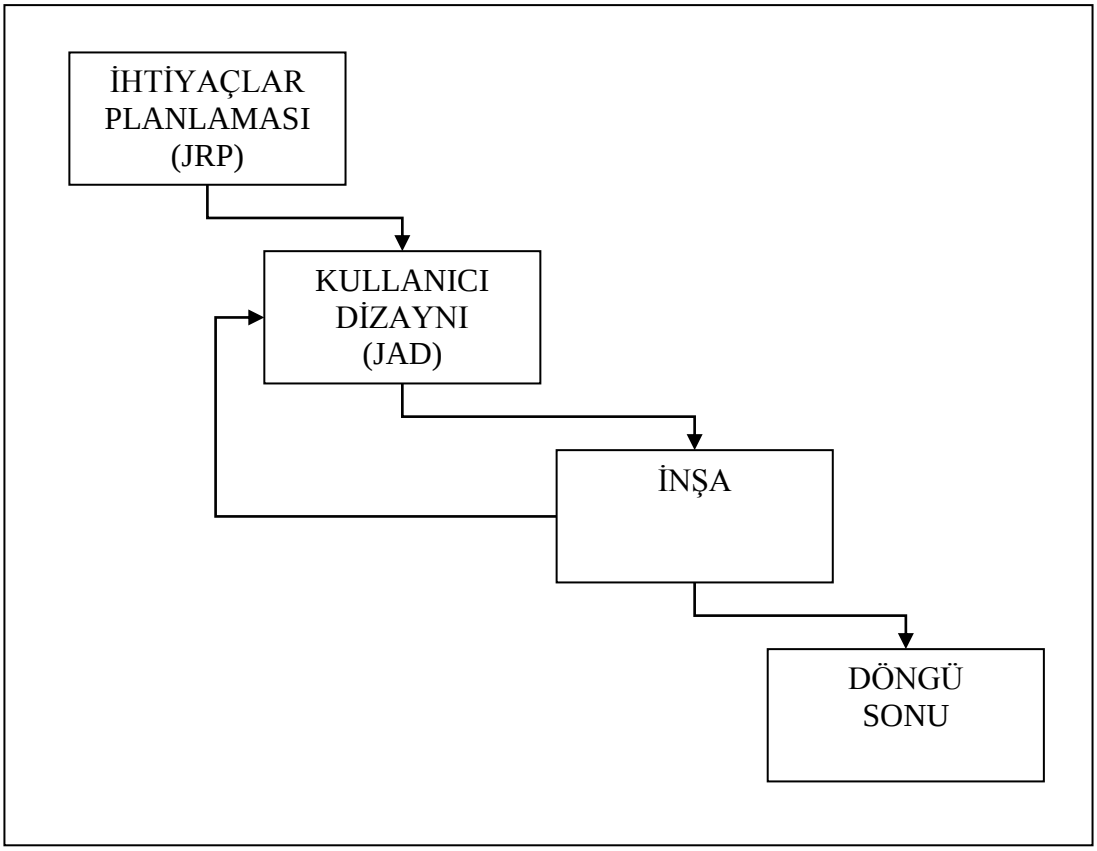
İnsan unsurunda dikkat edilmesi gerek bir diğer konu ise kişilerin daha önce beraber çalışmış olmalarının tercih sebebi olmasıdır. Daha önce beraber çalışmış kişilerin daha etkin iletişim kurabilmeleri bu tercihin sebebidir.

Yönetim: Yönetim metotların nasıl uygulandığıdır. Etkin bir yönetim olmadan metotların etkili olmayacağı aşikardır(The Gov. of HongKong, 2002).

Hızlı uygulama geliştirme (RAD) süreci nasıl işler?

RAD 4 fazdan oluşur :

- İhtiyaçlar Planlaması,
- Kullanıcı Dizaynı,
- İnşa,
- Döngü Sonu.



Şekil 2.9. Hızlı uygulama geliştirme fazları ve kullanılan bazı metotlar (Taggart,2001)

İhtiyaçlar planlaması : Bu fazda geliştirilecek uygulamanın kapsamında olan ihtiyaçlar belirlenir. Bu ihtiyaçlar önceliklerine ve basitliklerine göre sıralanır. Bu aşamada en fazla kullanılan metot JRP'dir (Joint Requirements Planning-Ortak ihtiyaçlar planlaması). Bu fazın bilgi sistemleri dışındaki katılımcıları organizasyonda fonksiyonların birbirleri ile ilişkisini bilen üst yönetimden olmaları tercih edilir (Cavaye, 1995).

1. Kullanıcı dizaynı : Bu fazda yazılımın fonksiyonel olmayan bir prototipi geliştirilir. Bu süreçte bilgi sistemleri personeli ile kullanıcılar beraber çalışırlar. Prototip sadece grafik arabirim dizaynı değil aynı zamanda veri akışlarını ve yapısını, aksiyon kodlarını (pseudocode isimli konuşma dili kullanılan kod özeti) içerir. Bu aşamada bilgi sistemleri personeli dizayna katılmaz, sadece karşılaşılabilecek teknik kısıtlar konusunda destek olur ve prototipleme araçlarını kullanırlar (Kusters ve diğerleri, 1999). Bu aşamada en fazla kullnılan metot IBM'in 70'li yılların sonlarında geliştirdiği JAD'dir (Joint application design-Ortak kullanıcı dizaynı).

2. İnşa : İnşa fazı bilgi sistemleri elemanları tarafından fonksiyonel bir prototipin geliştirildiği kısımdır. Bu takım tipik olarak 4 ile 8 kişiden oluşur. 4.nesil programlama dilleri kullanılır. Mümkünse COTS (Commercial Off The Shelf) tabir edilen hazır komponentler kullanılarak faz kısaltılmaya çalışılır. Geliştirilen her prototip kullanıcılara onaylatılır. Onaylanmayan prototipler için tekrar ikinci faza dönülür. Bu döngü, ihtiyaçları karşılayan bir prototip geliştirilinceye kadar veya belirlenen takvime ulaşıldığında biter (Taggart, 2001).

3. Döngü sonu : Bu fazda en son prototip yoğun testlere tabii tutulur. Ardından kullanıcı eğitimleri ve pilot uygulamaya geçilir. Pilot uygulama başarılı olursa organizasyon çapında uygulama hayata geçirilir.

Takvim kısıtı nedeniyle uygulanamayan ihtiyaçlar ve bu süreç esnasında oluşan ihtiyaçlar tüm sürecin tekrar ve bir öncekinden bağımsız olarak işletilmesi ile devam eder.

Tablo 2.3. Hızlı uygulama geliştirmenin avantaj ve dezavantajları (Maner, 1997)

AVANTAJLAR	***DEZAVANTAJLAR***
• Kısa geliştirme süresi	• Teknik performans
• Zamanla beraber azalan emek ve maliyet	• Değişen iş proseslerine uyumsuzluk
• İhtiyaçları daha çok karşılama	• Kısıtlı ölçeklenebilirlik
• Kullanıcı tatmini	• Kısıtlı yeniden kullanılabilirlik
	• Eksik dokümantasyon

Sonuç olarak RAD ihtiyaçların acil olduğu durumlarda kullanımı daha uygun olan bir uygulama geliştirme metodolojisidir (Martin, 1991). Kullanıcı katılımı diğer metodolojilere oranla daha fazla olduğundan ortaya çıkan uygulamanın teknik özellikleri diğer metodolojilere daha zayıf kalmasına rağmen fonksiyonel açıdan daha kısa zamanda daha fazla performans gösterdiği açıktır. Fakat RAD'ın genel

performansı unsurlarının kalitesine çok bağlıdır. Araç unsuru piyasadaki hazır çözümlerle bir kısıt olmaktan çıkmasına rağmen RAD'in şart koştuğu çoklu uzmanlığa sahip kişiler kolay bulunabilir değildir. Kısa geliştirme süresi sayesinde azalan emek maliyeti aslında birim maliyeti ya da marjinal maliyet yüksek olan bir kalemdir. Bu sebeplerden dolayı yazılım firmaları haricinde fazla uygulaması yoktur.

3. YAZILIM YAŞAM DÖNGÜSÜ KAVRAMININ EKS ECZACIBAŞI KARO SERAMİK SAN. VE TİC. A.Ş. UYGULAMASI

Eczacıbaşı, başlangıçta, yirminci yüzyıla girilirken Ege'de gerçekleştirilen ulusal bir ilaç üretim çabasının simgesiydi. Günümüzden 60 yıl kadar önce, Eczacıbaşı simgesinden İstanbul'da hem modern Türk ilaç endüstrisinin hem de Türk ekonomisinin öncü girişimlerinden birisi doğmuştur. Türkiye'ye yarım yüzyıl boyunca sağlık üreten Eczacıbaşı, 2000'li yıllara doğru, bu kez de dünyaya açılmaya başlayan bir topluluk görevini almıştır.

Eczacıbaşı Topluluğu'nu kuran Dr. Nejat Eczacıbaşı, ilk ilaç fabrikasını 1952'de Levent'te açmış ve birbirini izleyen yatırımlar sonucunda Eczacıbaşı İlaç ülkenin en büyük kuruluşlarından birisi olmuştur.

Eczacıbaşı'nın ikinci önemli çalışma alanı, yapı gereçleri olmuştur. Dr. Eczacıbaşı seramik üretimine de 1940'lı yılların başlarında girmiştir. Eczacıbaşı Seramik, ülkedeki ilk modern seramik sağlık gereçleri tesislerini 1958 yılında Kartal'da açmış, bunu Bozüyük'te yeni bir seramik kompleksi izlemiştir. Bir sonraki aşama ise, yapı sektörünün batarya, musluk, vana, sifon vb. sağlık armatürü gereksinimlerini karşılamak için Bozüyük'te kurulan Artema tesisinin 1983'te üretime başlaması olmuştur. Bundan sonra karo seramik, mutfak dolapları ve banyo küvetleri üretim tesisleri gerçekleştirilmiş, mutfak ve banyolar için anahtar teslim hizmet sunularak tam bir bütünlük sağlanmıştır. Eczacıbaşı Yapı Grubunun ürün yelpazesini genişletmek amacıyla kurulan Eczacıbaşı Karo Seramik (EKS), tek pişirim teknolojisi ile renk ve desen açısından özgün karolar üretmektedir.

Eczacıbaşı Karo Seramik fabrikası 1991 senesinde 3 milyon m² ile üretime başlamış olup bugün 14 milyon m² ile hizmet vermektedir. İtalyan Marazzi Ceramiche S.P.A şirketiyle joint-venture olarak kurulmuştur. Gerek İtalya'nın gerekse de Marazzi firmasının sektörde lider konumda olmaları nedeniyle başlangıçta "know-how" onlar tarafından karşılanmıştır. Kullanılan tüm üretim malzemeleri, fırınlar, presler onların bilgileri ile alınmıştır. Ancak, zaman içinde

ortaklıktaki payları azalmış, bugün sadece bilgi akışını sağlamak amacıyla bir miktar payları bulunmaktadır.

Üretim, yer ve duvar karoları,bordürler, havuz seramikleri olmak üzere çok çeşitli alanlardadır. Granit üretimine ve havuz seramiklerine son yıllarda ağırlık verilmiş olup ürün yelpazesi genişletilmiştir. Üç fabrikanın birisi Bozüyük'te olmak üzere iki tanesi İstanbul'dadır. Ayrıca, İrlanda'da 2000'li yıllarda Quelceram ile kurulan ortaklık, 2002 başlarında hisselerin tamamının EKS'ye devri ile sonuçlanmıştır. İrlanda'daki bu fabrika aracılığı ile İngiltere pazarı hedeflenmiş olup, yoğun şekilde Türkiye'ye de ihracat yüklemesi yapılmaktadır. Bozüyük fabrikasında ilk başlarda sadece yer ve duvar karoları üretilmiştir. Tuzla'daki fabrikaların birinde dekorlar, diğerinde ise 10x10, 5x5 ve son olarak da 2,5x2,5 ebatlarında karolar üretilmektedir. Tuzla fabrikaların kapasiteleri gelen talebi karşılayamaz noktaya geldiğinden buradaki ürünlerin bir kısmı Bozüyük'teki fabrikaya taşınmıştır.

Günümüzde, şirketlerin başarısı, bilginin edinilmesi, yaratılması, saklanması ve etkin paylaşımıyla orantılıdır. Daha hızlı, daha iyi bir iletişim işlerin yapılma süresini kısalttığı gibi kalitesini de önemli ölçüde arttırır. Hızlı ve doğru karar vermede kapsamlı bilgiye çabuk erişimin önemi çok büyüktür. Dolayısı ile, EKS'nin rekabet gücünü doğrudan etkileyecek bir çalışmanın öncelikle maliyetlerin düşürülmesine en büyük katkıyı sağlayacak olan satınalama departmanında yapılmasına karar verilmiştir. Satınalma departmanının mevcut işleyişi incelendiğinde bunun aslında günümüz koşullarında bir zorunluluk olduğu da görülebilir.

Eczacıbaşı Karo Seramik San ve Tic A.Ş.'de; tüm hammadde,yardımcı malzeme, yedek parça işletme malzemesi ve makine alımlarının etkin ve efektif olarak gerçekleştirilmesi için satınalma departmanı kurulmuş ve departman şirketin ilk kuruluşundan beri organizasyondaki yerini korumuştur. EKS Satınalma departmanı, şirketin sahip olduğu üç fabrikaya daha etkin ve hızlı hizmet sunabilmek amacıyla kendi içinde fonksiyonellik ön plana çıkarılarak üç bölüme ayrılmıştır. Bu bölümleri :

1. Dış Satınalma
2. Bozüyük İç Satınalma

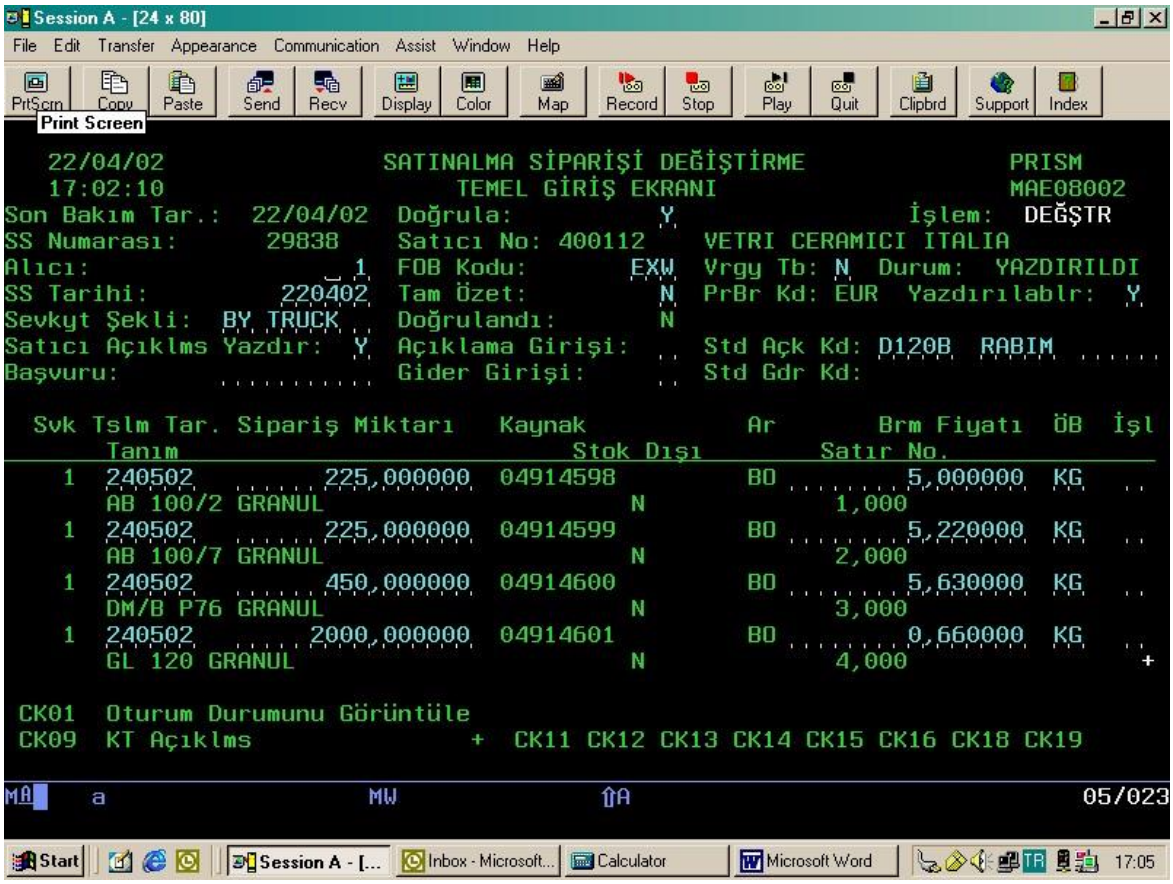
olarak sıralamak mümkündür. Tüm bu bölümler, birbirleriyle koordineli çalışarak şirket amacına hizmet etmektedir. İşleyiş hemen hepsinde benzer özellikler göstermekle birlikte, birtakım ithalat prosedürleri sebebi ile dış satınalma bölümünde farklılıklar göze çarpmaktadır. Bu sebeple, incelemeye dış satınalma bölümünün iş akışı ile başlamak uygun olacaktır.

Mevcut işleyişte, şirkette, planlama departmanları tarafından malzeme planlaması yapılmakta, ihtiyaçlar belirlenmekte ve ilgili sipariş talepleri birtakım listeler halinde Satınalma departmanına e-mailler aracılığı ile iletilmektedir. Tüm şirkette, AS-400 tabanlı Marcam Prism kullanılmaktadır. 1993 yılında alınan bir karar doğrultusunda EKS'de hayat bulan bu yazılımın, 10 yıla yaklaşan ömrü boyunca pek çok departmandan ihtiyaçları doğrultusunda yeni uygulama talepleri gelmiştir. Dış Satımın, muhasebenin bu talepler doğrultusunda hazırlanmış özel uygulama paket yazılımları da vardır. Bu uygulamalar arayüzler ile prisme bağlanmışlardır. Ancak gerek prismin ithalat konusundaki yetersizliği, gerek veri girişindeki zorluklar satınalma çalışanlarını kendi veri tabanlarını oluşturmaya itmiştir. Kişisel olarak saklanmakta olan verilerin bilgiye dönüşü çoğu zaman güçlükle yapılmakta, kimi zaman da güncellenmediğinden yanılgılara neden olmaktadır.

Örnek vermek gerekirse, mevcut işleyişe göre prismde ancak sipariş açmak ve bu siparişlerin gümrüğe gelip gelmediğini ya da ambarda olup olmadığını görmek mümkündür. Ancak siparişin ne zaman hangi fatura ile sevk edildiğinin, gümrükten çekilebilmesi için işleme verilip verilmediğinin bilgisine prismden ulaşmak, bu şekilde takip etmek mümkün değildir. Dolayısı ile, bu takipler, çalışanların kendi oluşturdukları birtakım dosyalar ve bu dosyalarda kimi zaman kağıt üzerinde tuttukları kayıtlar yardımıyla yapılmaktadır.

Mevcut işleyişe göre, sipariş talebi departmana geldikten sonra prismde sipariş yaratılmakta ve bu siparişin içeriği olan sipariş numarası, firma adı, malzeme tanımı ve miktarı ve hangi fabrikanın sipariş talebi olduğuna dair bilgiler 'dosya takip' isimli klasöre kaydedilmektedir (Ek A.1). Kayıt bilgisayar ortamında değildir. Daha sonra prismden alınan sipariş yazısı ve MS Word'de ayrıca yazılmış nakliyeci talimatı Satınalma Şefi ve Müdürünün onayına sunulur. Aslında nakliyeci talimatının

oluşturulması da çalışanı yaptığı işi tekrarlamaya iten bir iştir. Zira, prismde sipariş yaratılırken malların hangi nakliyeci ile taşınması gerektiği, firma adı, malzeme adı ve miktarı, sipariş numarası vb. bilgiler sisteme girilmiştir. Bu bilgilerin tekrar MS Word'de bir dosyaya yazılması ve bu çıktının arşivlenmesi iş tekrarından başka bir çalışma değildir. (Şekil 3.1 ve Ek A.1.)



Şekil 3.1. Prismde sipariş oluşturma ekranı

Departman şefi ve müdürünün onayı alındıktan sonra sipariş yazısı tedarikçi firmalara, nakliyeci talimatı da ilgili nakliyeci firmalara fakslanır. Onay kağıt üzerinde olduğu ve daha birçok kağıdın oluşturduğu dosyaların tam görünmesi için şirketteki faks sunucu kullanılmamaktadır.

Çıktıların çokluğu ve gerçekte içerdikleri bilgilerin aslında birer veri tekrarından ibaret olması, bilgiye erişimin güçlüğü sebebiyle departmanda oluşan zaman ve verimlilik kayıpları önlenmek istenmiştir. Bu çalışmanın temelini oluşturan bu görüş çerçevesinde, Satınalma departmanı için kullanılabilecek yeni bir program ihtiyacı doğmuştur.

Yeni bir program yazılımı demek elbette, üst yönetimin ve bilgi sistemleri departmanının onayı ile gerçekleşebilecek bir durumdur. İhtiyaçlar ve beklentiler doğrultusunda oluşan talepler üst yönetime iletildikten sonra konunun bilgi sistemleri departmanınca incelenmesi istenmiştir. Bu aşamada, satınalama yöneticisi başkanlığında bilgi sistemlerinde ve satınalma departmanında çalışanlardan bir proje grubu kurulmuştur. Bu grubun başlangıçtaki amacı "neredeyiz?", "nereye ulaşmak istiyoruz?" ve "hedeflerimize ulaşmamız için alternatiflerimiz nelerdir?" gibi sorulara cevap aramaktır. Zira, her ne kadar satınalma çalışanları prismin beklentilerine cevap vermediğine inansalar da, bilgi sistemlerinden sağlanabilecek bir destek (yetkinleştirici bakım) ile prisme yeni yetenekler eklemek, performans düzeyini yükseltmek mümkün olabileceği düşünülmüştür. Bilindiği gibi bakım, yaşayan bir sistemi ayakta tutmak anlamına gelir ve bir çok durumda yeni sistem geliştirmekten çok daha etkin ve verimli bir çözüm olabilmektedir. Bu sebeplerden dolayı proje çalışmalarının bilgi sistemleri departmanı ile ortaklaşa götürülmesine karar verilmiştir. Böylece, projedeki kararlar, kullanıcı ya da müşteri olarak adlandırılan satınalma elemanlarının talepleri doğrultusunda ve bilgi sistemlerindeki elemanların profesyonellikleri öncülüğünde alınacaktır.

2000'li yılların başlangıcındaki ekonomik belirsizlikler, yöneticileri, yeni bir program satınalmak yerine, şirket ihtiyaçlarına hitap edecek bir paket yazılması düşüncesine itmiştir. Gerek maliyet, gerekse ihtiyaçları karşılayabileceğine olan inancımız yüzünden yeni sistemin MS Access'te kurulmasına karar verilmiştir. Zira, kullanım kolaylığı ve yatırım maliyetinin olmaması bu başlangıç kararının verilmesini desteklemektedir. Ancak, yine de mevcut sistem ve onun aksayan noktalarını irdelemeye devam edecek olursak, yeni bir sisteme neden ihtiyaç duyulduğu ortaya çıkacaktır. Bu aşama aslında beraberinde gereksinimlerin tanımlanmasını da getirecektir. Proje en başından beri bilgi sistemleri-satınalma işbirliği içinde yürütüldüğü için yeni uygulamanın nasıl çalışması gerektiği aslında kendiliğinden ortaya çıkacaktır. Bu sebeple, işleyişi anlatmaya devam etmek uygundur.

Sipariş yazısını ve nakliyeci talimatını ilgili firmalara faksladıktan sonra tedarikçi firmalardan proforma faturalarını göndermeleri beklenmektedir. Ancak proforma faturanın gelişi sanıldığı gibi bilgisayar ortamında bir kaydı etkilememekte ancak 'Dosya Takip' isimli klasörde proforma fatura tutarı işlenerek bir anlam

kazanmaktadır. Bu anlam, aslında sipariş yazısının bir dosya haline gelmeye başlamasıdır. Ama tahmin edileceği gibi prism ya da herhangi bir bilgisayar sisteminden proforması gelmeyen siparişler raporu almak mümkün değildir. Bu raporun oluşturulabilmesi için Dosya Takip'in sayfa sayfa incelenerek fatura tutarı işlenmemiş olan siparişlerin kayıtlarının bir kenara not edilmesi gerekmektedir. Tabi bu da çokça zaman gerektiren insanları ikinci kez iş yapmaya zorlayan bir çalışmadır. Raporun gerekliliğine gelince, bu rapor sayesinde fakslamış olduğunuz siparişlerin firmaların eline geçip geçmediğini anında görmek mümkün olacaktır. Dolayısıyla, yeni planlanan sistemde sipariş kayıtları ve proforma bilgilerini içeren ekranlar olmalıdır.

Sevkıyatın beklendiği zaman yani termin tarihinde tedarikçi firmaların malları sevk edip etmediklerini kontrol amaçlı raporlar alınmakta, raporlar tedarikçi firmalara fakslanarak geri bildirim alınmaktadır. Ancak bu raporlar da tahmin edildiği gibi kolay kolay oluşturulamamaktadır. Zira, firmalar malları sevk ettikten sonra ilgili faturalarını EKS'ye fakslamakta veya sevkıyat bilgilerini içeren yazılarını göndermektedir. Bu aşamada işin içine yine bilgisayar ortamında olmayan bir dosya girmektedir. Bu dosyanın adı 'Gümrük' olup faturalar geldikçe sipariş numarası, firma adı, fatura tutarı, fatura numarası ve tarihi malzeme tanımı ve miktarı, sevkıyat bilgileri gibi kayıtların tutulması amacıyla kullanılmaktadır (Ek A.2). Mevcut durumda, sevkıyatları takip etmek amacıyla kullanılan rapor, prismden alınan, ilgili firmaya ait tüm açık siparişlerin listelenmesi ile oluşturulmuş bir rapordur. Ancak bu raporun içinde daha önceden sevk edilmiş, gümrük lokasyonuna gelmiş siparişler de yer alabilmektedir. Bu tür bilgiler için tedarikçi firmaya tekrar aynı sevkıyatları sormak takipsizliğin göstergesi olduğundan gümrük dosyasına işlenmiş sevkıyatlar tek tek raporlardan ayıklanarak yola çıkmamış malzemeler belirlenerek doğru rapor oluşturulur. Bu bir çalışanın en değerli varlığı olan zamanını harcamasından başka bir şey değildir. Zira yeni yazılan programda sevk edilmeyen malzemeler raporunu almak son derece kolaydır. Çünkü bilgilerin tamamı bilgisayar ortamında ve tek bir veri tabanında kayıtlıdır. Bilgisayar ortamında olmayan verilerden ya da tek bir ortamda saklanmayan verilerden doğru bilgiye ulaşmak, bu bilgileri raporlamak çok güçtür.

Sevkıyatlara ilişkin faturalar geldikçe gümrük dosyasına işlenir demiştik. Aynı şekilde malzemeler gümrüklere ulaştıklarında da varış gümrüğü ve tarihi bu

dosyaya işlenir. Bu da planlamalar tarafından tıpkı malzemenin sevkıyatının izlenemediği gibi çok net görülemeyen bir durumdur. Çok net görülememektedir dememin sebebi, gümrüğe gelmiş malzemeler için prismde gümrük lokasyonu tanımlanmıştır. Anacak bu lokasyon sadece hammaddeler ve karolar için kullanılmaktadır. Yedek parça, işletme malzemesi ve makinalar için satınalma elemanı dışında bir kimsenin sipariş isteğinin hangi durumda olduğunu görebilmesi mümkün değildir. Prismde lokasyonları yolda, gümrükte, işlemde vs. gibi çoğaltmak ve tüm malzeme grupları için uygulamak mümkün ancak uğraştırıcı ve bir o kadar da zorlayıcı bir iştir. Zira, EKS'de bir ayda yaklaşık yüz dosya açılmakta ve her bir dosya birden çok malzemeyi içermektedir. Prismde bir lokasyonun değiştirilmesi malzeme bazında yapılabilmektedir. Her bir malzeme için önce aktarılacağı lokasyon tanımlanmakta, sonra bir önceki lokasyondan diğer lokasyona geçişi tarihlerle işlenmektedir. Bu işlem, az önce de bahsedildiği gibi, sadece hammadde grubu için malzemeler gümrüğe geldikten sonra yapılmaktadır. Dolayısı ile, sadece bir malzemeyi içeren bir lokasyon değişimi için bile prismin çalışma hızına bağlı olarak ortalama 7-8 dakika sürebilmektedir.

Gümrük lokasyonuna gelmiş bir malzemeyi, planlama departmanındakiler prismden görebilmekte ve ihtiyacı doğrultusunda malzemenin gümrükten çekilmesini talep etmektedir. Gümrükten çekme denilen işlem gümrük komisyoncusuna iş emri verilmesi ile başlayan banka ve birtakım yasal işlemlerden sonra malzemenin ambara ulaşmasına kadar geçen süreci kapsamaktadır. Mevcut işleyişte, gümrük komisyoncusu talimatı MS Excel'de yazılmış bir talimattır. Gümrükçü İş Emri denilen bu talimat, gümrükten çekilmesi planlanan işlemin bulunduğu gümrüğün adı, o işleme ait tarih, numara, tutar gibi fatura bilgileri, dosya no, firma adı, malzeme cinsi ve faturanın ne şekilde hangi bankadan ödeneceğine ilişkin bilgileri içerir. Bilgilerin çoğu gümrük dosyasından alınarak bilgisayar ortamına taşınır. Anlaşılacağı gibi ikinci bir iş yapılmış olmaktadır. Aslında en başından beri söylediğim gibi tüm kayıtlar bilgisayar ortamında düzenli bir şekilde tutulursa bu iş tekrarlarından ve oluşabilecek hatalardan kurtulmak mümkün olur.

Bütün bu işlemlerden sonra yani gümrük komisyoncusu işlemi tamamladıktan sonra yaptığı masrafları ve gümrük beyannamesini içeren dekontu satınalma departmanına gönderir. Beyanname numarası ve tarihi gümrük dosyasına işlenir ve böylece işlemin tamamlandığı anlaşılabilir. Bu aynı zamanda işlemin dekontunun

görüldüğü ve kontrol edildiği anlamını da taşımaktadır. Bu aşamada dekontlar muhasebeye teslim edilmeden bir işlem daha yapılmaktadır. Her gümrük komisyoncusu iş emri, dosya no, tarih, gümrük adı, fatura tutarı, firma adı gibi birtakım bilgiler ile dekont tarih ve numarası, fiili ithalat tarihi vb bilgiler bilgisayarda bir MS Excel sayfasına kaydedilerek işlemlere ait dekontların gelip gelmediği aylık olarak takip edilir. Bunun amacı muhasebe departmanına dekontların kesildiği ay içerisinde ulaşmasını sağlamaktır. Muhasebe kayıtları için önemli olan bu işlem ayrıca gümrük komisyoncularının performanslarının izlenmesi için de gereklidir.

Aynı kayıtlar faturaların kontrolü yapılırken bir başka çalışan tarafından da bir şekilde tutulmaktadır. Teorik hesaplamaların olduğu bir MS Excel sayfasına bir başka çalışan dekont masraflarının yanı sıra beyanname tarih ve numarası, ilgili işleme ait dosya numarası, fatura tutarı vb. bilgileri girmektedir. Bu iki birbiri ile bağlantılı iş için, iki ayrı veri tabanı oluşturularak zaman ve verimlilik kaybına neden olunmaktadır. Aslında, gümrükçü iş emirleri çıktı olarak saklanmak ve sonra birden fazla ekrana defalarca girilmek yerine her iş emri için bir kayıt yaratılarak, bu kayıtların ilgili şahıslar tarafından kendi amaçları doğrultusunda kullanmaları elbette sağlanabilir.

Ayrı ayrı oluşturulmuş ancak tek bir veri tabanı ile yürütülmesi gereken bir başka konu da akreditifler ile ilgilidir. Satınalma departmanının mevcut işleyişinde bir çalışan akreditifleri açmak, bir diğeri bunların ödemelerini takip etmek ile yükümlüdür. Prism bu tür kayıtları saklamak için çalışanlara olanak sunmadığı için yine MS Excel'de birtakım veritabanları oluşturulmuştur. Akreditifi açan kişi, açtığı her akreditif için bir kayıt yaratmakta ve bu kayıta dosya numarası, para birimi, akreditif tutarı, akreditif vadesi, akreditifin açıldığı banka adı, referansı ve komisyon tutarı gibi bilgileri işlemektedir. Ödemeleri takip eden kişi ise, ayrı bir MS Excel sayfasında dosya numarası, para birimi, akreditifin bankası, referansı, ödeme tutarı ve tarihi gibi bilgileri işlemektedir. Birbirinden bağımsız yapılan bu takipler çok zaman akreditiflerin ödeme vadelerinin atlanmasına neden olmaktadır. Zira, açılan her akreditifin kaydı olduğu halde ödeme takibi yapılırken bankaların verdiği bilgiler göz önüne alındığı için bazı akreditiflerin kaydı atlanabilmektedir. Ödeme vadesi geldiğinde, satınalmada kaydı gözükmeyen akreditif için finansmana daha önce haber verilemediğinden finansman da zorluklar yaşamaktadır. EKS'de açılan

akreditifler genelde parsiyel yüklemeye izin verdiği için sevkıyat bazında ödeme vadeleri oluşmakta ve finansmanı zor duruma düşürmemek için bu vadelerin çok sıkı bir şekilde takip edilmesi gerekmektedir. Bu açıdan bakıldığında, iki ayrı kişinin iki ayrı yere yaptığı, genelde aynı verileri içeren girişlerin, iş tekrarı ve zaman kaybından çok yanılgılara sürüklediği söylenebilir. Halbuki, tek bir veri tabanında bu tür veriler saklanırsa, hem bankadan ödeme vadesi bildirilmemiş akreditiflerin takibi daha kolay ve etkin yapılabilir, hem de tüm akreditifler için vadeler önceden bilindiğinden finansman hiçbir zaman zor durumda bırakılmamış olur.

Bir başka konu da, özellikle krizde daha ön plana çıkan tedarikçi firmalara ilişkin borçların takibidir. Ödemelerin %95'e yakın kısmının vadeleri olduğu göz önüne alındığında tedarikçilere ilişkin borçların da etkin bir şekilde izlenmesi gerekli olduğu görülebilir. Daha önce bahsedildiği gibi her bir sevkıyatla birlikte oluşan faturalar bilgisayar ortamında değil de gümrük dosyasına işlenerek takip edilmekte idi. Dolayısı ile, bu bilgilerin tekrar bilgisayar ortamına taşınması söz konusudur. Ödemelerin daha kolay takip edilmesi amacıyla, satınalma departmanından bir kişi mal mukabil çalışılan firmalar için, dosya numarası, para birimi, tutar, firma adı, fatura numarası, tarihi ve vadesi gibi bilgileri bir MS Excel dosyasına işlemektedir. Her firma için ayrı bir sayfa açılmış ve faturalar ödendikçe ilgili ödemelere ait ödeme tarihi girilerek borç durumu güncellenmektedir. Ancak bu işlem elbette bütünü görmeyi engellemektedir. Raporlaması kolay değildir. Aylık bazda vadesi dolan ödemelere ilişkin raporu ya da firma bazında borçları gösterir raporu anında almak mümkün değildir. Çoğu zaman firmalar ile yapılan mutabakatlarda da bu girişlerin yetersiz kaldığı hissedilmektedir.

Tedarikçi firmalara ilişkin borçların etkin bir şekilde yönetilmesi demek şirketin nakit akışına katkı sağlamak anlamını taşımaktadır. Zira, her türlü ödeme finansmanı doğrudan etkilemektedir. Firmalardan alınan ödeme vadelerinin uzunluğu da finansmanın bu süre içinde sahip olduğu nakiti istediği şekilde kullanabilmesi demektir. Tedarikçi firmaya ödemesinin vadesinde veya firmayı zor durumda bırakmayacak, mal sevkıyatını durdurmayacak şekilde yapılması için bu vadelerin çok iyi takip ediliyor olması gerekmektedir. Firma borçları bir takım yasal zorunluluklardan ötürü muhasebe tarafında da takip edilmektedir. Ancak tedarikçilerle olan ilişkileri düzenleyen ve sürdüren satınalma departmanı olduğundan ödeme kararlarını vermede de satınalma departmanı yükümlüdür. Hangi

firmaya ne kadar ödeme yapılacağına, gümrükten hangi malların çekileceğine ilişkin kararlar verildikten sonra haftalık ilerleyen aylık ödeme planını içeren raporlar ve bir sonraki dört aylık tahminler finansman departmanına bildirilmektedir. Böylece finansman haftalık olarak nakit çıkışını planlayabilmektedir. Satınalma departmanı borçlarını ne kadar net görebilir ise finansmana da o ölçüde doğru bilgi verebilir.

Satınalma departmanın doğru verileri kullanması, finansmanı nasıl etkiliyorsa aynı şekilde planlama departmanlarını da etkilemektedir. Bir malzemenin yola çıkıp çıkmadığı, gümrüğe ulaşıp ulaşmadığı ya da ne zaman ve ne ile sevk edileceği bilgileri planlamalar için son derece önemli ve doğru olması gereken bilgilerdir. Bir planlamacı herhangi bir malzemeyi gümrükte görür ise o malzeme ile yapılacak üretimleri bir süre sonrası için planlayabilir demektir. Ancak, gümrük lokasyonuna gelmemiş malzemeler onun için bilinmez bir kara kutudan ibarettir. Zira, siparişten gümrüğe gelene kadar yaşanan süreci, şu anda sadece satınalmadaki elemanlar takip edebilmektedir. Oysa, planlama departmanları ihtiyaç duydukları malzemenin ne zaman sevk edileceğinin, ne zaman gümrüğe varacağının ve ne zaman ellerinde olacağının bilgisine her an ulaşabilmelidirler. Aksi halde, üretim programları aksamakta, stoka mal yapılmasına neden olunmaktadır.

Satınalma departmanına bağlı çalışan malzeme ambarları da bilginin zamanında ve doğru gelmemesinden kaynaklanan problemler yaşayabilmektedir. Tüm girişler insana dayalı olduğu ve defalarca yapıldığı için atlamalar ve hatalar olması son derece normal karşılanmalıdır. Örneğin, sipariş edilen bir malzeme için hangi fabrikanın siparişi olduğunu belirten kod prism ekranında ve dosya takipte ambar tanımı yazılarak belirtilmektedir. Ancak bu kodlama sevkıyatların yazıldığı gümrük dosyasına yansımamaktadır. Prisme bakılmadan yapılacak bir malzeme bildirimi – ki gümrük dosyasından gümrükten çekilen her malzeme için ambarları bilgilendirmek amacıyla yapılır- hatalı olabilmektedir. Oysaki, malzeme bildirimleri; dosya no, para birimi, tutar, fatura no ve tarihi, malzemenin tanımı, miktarı ve ilgili ambar, sipariş isteğinin sahibi bilgilerinden oluşmaktadır. Dolayısı ile, sistemden kolaylıkla temin edilebilmeli, hatalara meydan bırakmamalıdır.

Görüldüğü üzere siparişe ilişkin bir bilgi farklı işlemler için defalarca gündeme gelmekte ve her seferinde tekrar yazılarak işlem sürdürülmektedir. Her girişin zaman kaybına ve muhtemel hatalara neden olduğunu söylemek yanlış olmaz.

Son olarak, gümrükten çekim işlemi tamamlanmış olan malzemeler için ambarlar malzeme bildirimi yapan çalışanın aynı zamanda bu bilgileri bilgisayar ortamında bulunmayan dosya takip dosyasına da işlemektedir. Sipariş edilmiş malzeme miktarlarının yanına gümrükten çekilen miktar, çekiliş tarihi ve işlemin toplam tutarıyla birlikte ödeme şekli de bu dosyaya kaydedilmektedir. Her ayın sonunda gümrükten gelen dekontlar ile birlikte bu dosya incelenir ve mal mukabili olarak yapılmamış işlemler için kambiyo kapatması yapılmaktadır. Ancak bu işlem için de, beyanname numarası ve tarihi, dosya numarası vs bilgilerinin tekrar girilmesiyle oluşturulan MS Word çıktıları kullanılmaktadır.

Mevcut sürecin belirlenmesi için departman çalışanları ve departmanın iç müşterileri ile görüşülmüştür. Satınalma departmanı için mevcut akış şeması oluşturulmuştur. Oluşturulan bu akış şeması üzerinden satınalma yöneticisi ve çalışanlar ile tekrar birebir görüşmeler yapılarak tıkanıklıkların, darboğaz noktalarının tespit edilmesi sağlanmıştır. Daha sonra toplu olarak beyin fırtınası metodu ile çalışanların yaşadıkları sıkıntıları nasıl aşmayı planladıkları, prismden kaynaklanan problemleri ve beklentileri üzerinde yoğunlaşılmıştır. Mevcut süreç çalışmasının MS Visio programı ile çizilmiş halini ekte görmek mümkündür (Ek A.3)

Çalışanların beklentilerini karşılamak, iş tekrarlarını elemine etmek, bilgiyi paylaşılabilir hale getirmek mevcut sürecin iyileştirilmesi için yeni bir program yazılması düşünülmüştür. Ancak, maliyetler göz önüne alındığında, bu programın kolay yazılabilmesi ve şirket amacına hizmet edebilecek başarıyı sağlayabilmesi beklenmektedir. MS Access'te düşünülen bu programın ekranlarını aşağıda görmek mümkündür. Yeni program, mevcut işleyişi çok fazla değiştirmeden ancak, iş tekrarlarını ortadan kaldırmayı amaçlayarak ortak bir veri tabanı kurulması mantığı ile yazılmıştır.

Bilindiği gibi MS Access'te veri tabanları ve programlar oluşturmak, bilgileri düzenlemek ve raporlamak son derece basittir. Tüm bilgisayarlarda MS Access'in kurulu olması ve çalışanların da programa yabancı olmaması sebebiyle kullanımının da sorun teşkil etmeyeceği açıktır. Zaman içerisinde oluşabilecek raporlama ihtiyaçlarına da cevap verebilecek yapıdadır. Ancak program güvenliği açısından herkesin yetki seviyesi belirlenmiş ve çalışanlara kullanıcı adları ve şifreler tanımlanmıştır.

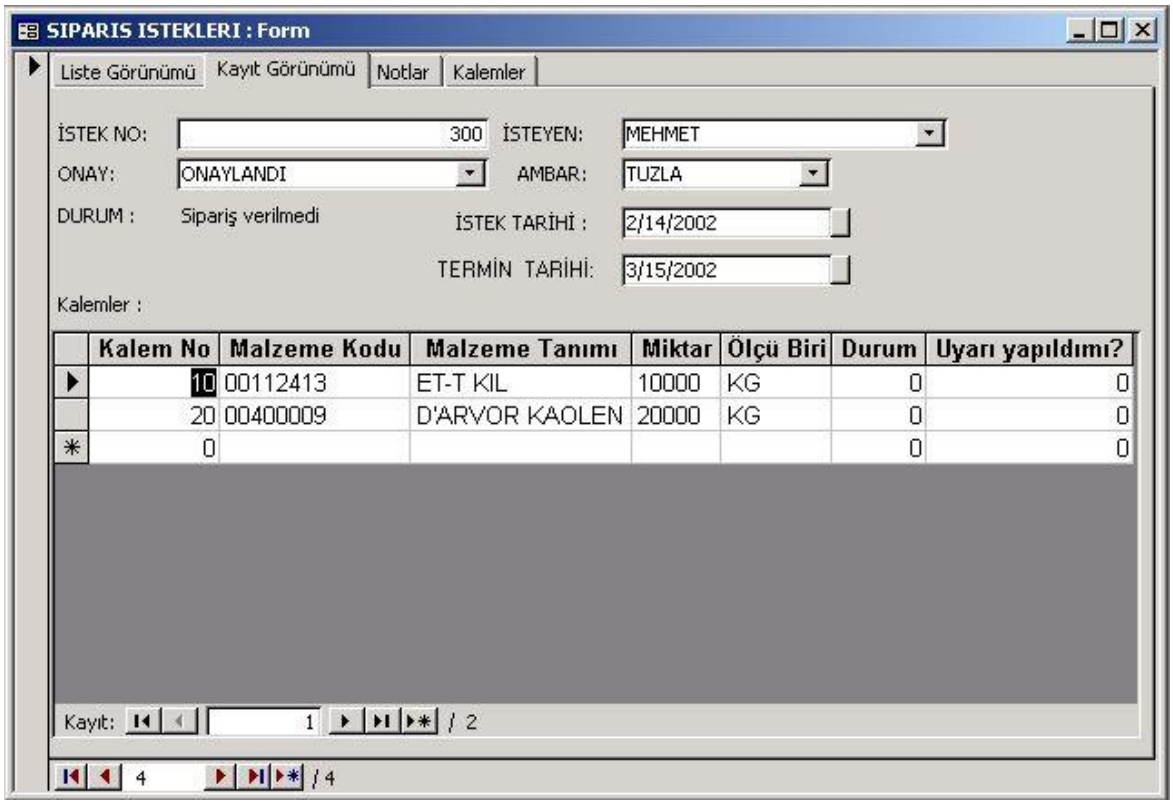
ODBC aracılığıyla prismin kullandığı DB/2 veri tabanına belirli aralıklarla sipariş bilgileri gönderilmesi planlanmaktadır. Böylece şirket genelinde kullanılan prisme veri akışı sağlanmış olacaktır. Çift taraflı besleme için ara yüzler şirketteki bilgi işlem sorumlularınca yazılmaktadır. Ancak programın halen tam olarak kullanımına geçilmediği için prismden bilgi alış verişine henüz imkan vermemektedir. Test aşamasından sonra uygulamaların tamamının yeni program üzerinden yürütülmesi, sadece gerekli birtakım bilgilerin prisme gönderilerek veri tabanının güncellenmesi istenmektedir.

Projenin en başından beri çalışmalar birlikte sürdürüldüğü, anında geri beslemeler yapıldığı ve çalışanlar arasında çok iyi bir iletişim ortamı sağlandığı için uygulama aşamasında büyük problemler çıkacağı düşünülmemektedir. Düzenli aralıklarla yapılan toplantılarda gerekli gözden geçirmeler yapılmış ve ortak dil kurulmuştur. Tasarlanacak yeni yazılımın sadece mevcut işleyişe ve beklentilere değil, gelecekteki beklentilere ve zamana ayak uydurması gerekliliği düşünülmüştür. Bütün bu proje için zaman kısıdı konmamış, ancak sistem gereksinimlerinin belirlenmesi ve tanımlanması aşaması için dört aylık bir ön süre konulmuştur.

Dört ayın sonundaki toplantıda, başlangıçta verilen yeni yazılıma ilişkin kararların doğru olduğu ve proje grubunun çalışmalarını daha yoğun bir şekilde sürdürmesi gerektiği sonucu çıkmıştır. Tasarım aşamasında da, proje grubundan beklenen, koordineli bir şekilde çalışmaya devam ederek ve gerektiğinde müşterileri de proje kapsamına alarak projenin mümkün olan en kısa zamanda tamamlanmasıdır.

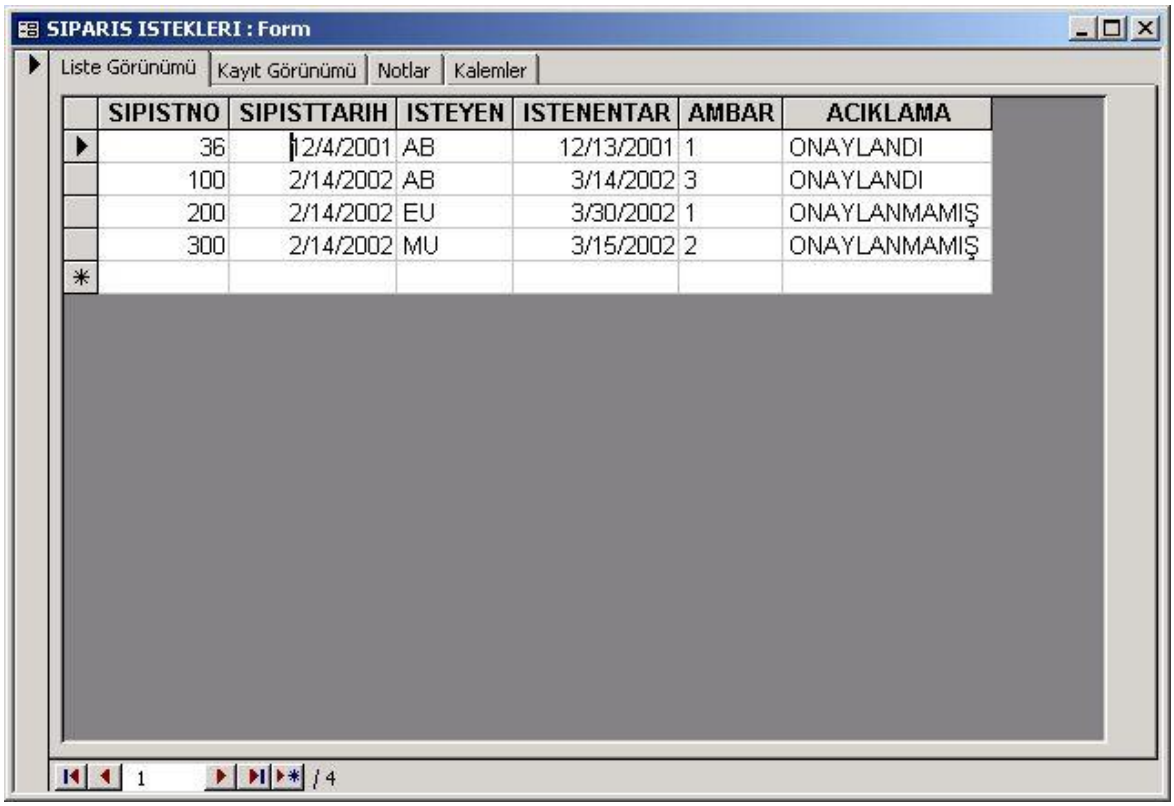
Yeni yazılım, sadece satınalma departmanının değil, şirketin hedeflerine ulaşması için tasarlanmaktadır. Bu sebeple, satınalma departmanının iç müşterilerinden olan planlama departmanlarının da yeni programa erişimleri için kullanıcı adları ve şifreleri tanımlanmıştır. Dolayısı ile artık satınalma departmanındaki çalışanlara sormaksızın sipariş isteklerinin durumunu anında görebileceklerdir.

Bundan sonraki bölümlerde tasarlanan ekranlar ve bu ekranların özellikleri irdelenecektir. Satınalma süreci departmana satınalma isteğinin ulaşması ile başladığına göre incelenmesi gereken ilk ekran sipariş isteği ekranıdır:



Şekil 3.2. Sipariş İsteği Ekranı-Kayıt Görünümü

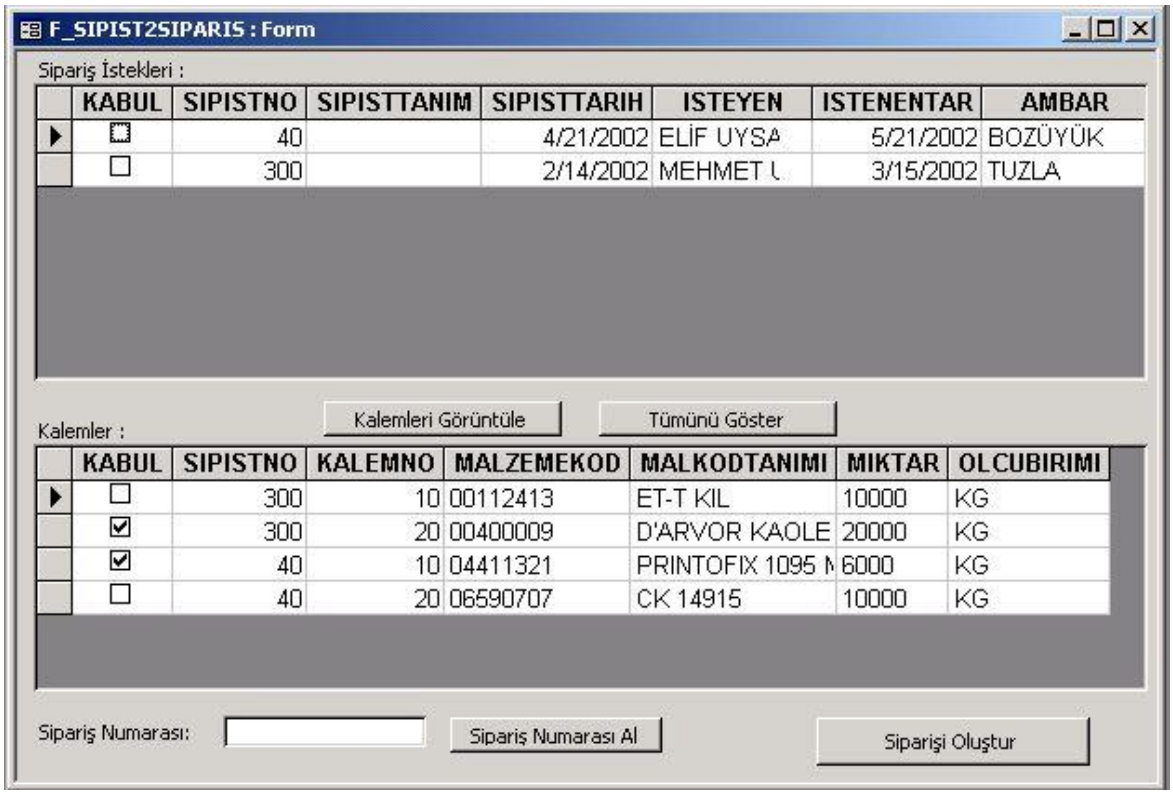
Bu ekranı malzeme talebinde bulunan kişinin ya da talepte bulunana yerine satınalma elemanının doldurması beklenmektedir. Onay kısmı bir malzeme isteğinin bir üst makamca uygun bulunduğunun göstergesidir. Ancak onaylanmış sipariş istekleri siparişe çevrilebilir. Bu ekranda görülen termin tarihi sipariş isteğini yaratan kişinin malzemeyi istediği tarihtir. Sipariş istek no. otomatik olarak program tarafından belirlenir. Ekranda görülen durum kısmından ise isteğin siparişe çevrilip çevrilmediği, red olup olmadığı, kısmi kabul görüp görmediği anlaşılmaktadır.



Şekil 3.3. Sipariş İsteği Ekranı-Liste Görünümü

Görüldüğü gibi liste görünümünde tüm sipariş isteklerini, tarihlerini, isteyen kişileri, istenen tarihleri ve hangi sipariş isteklerinin onaylandığını takip etmek mümkündür. Notlar ekranında, ilgili sipariş isteğini yaratan kişinin belirtmek istediği notlar yazılabilir. Kalemler ekranı ise, kayıt görünümdeki malzeme kalemlerinin girildiği ekrandır. Kayıt görünümündeki kalemler salt-okunur halde bulunmaktadır.

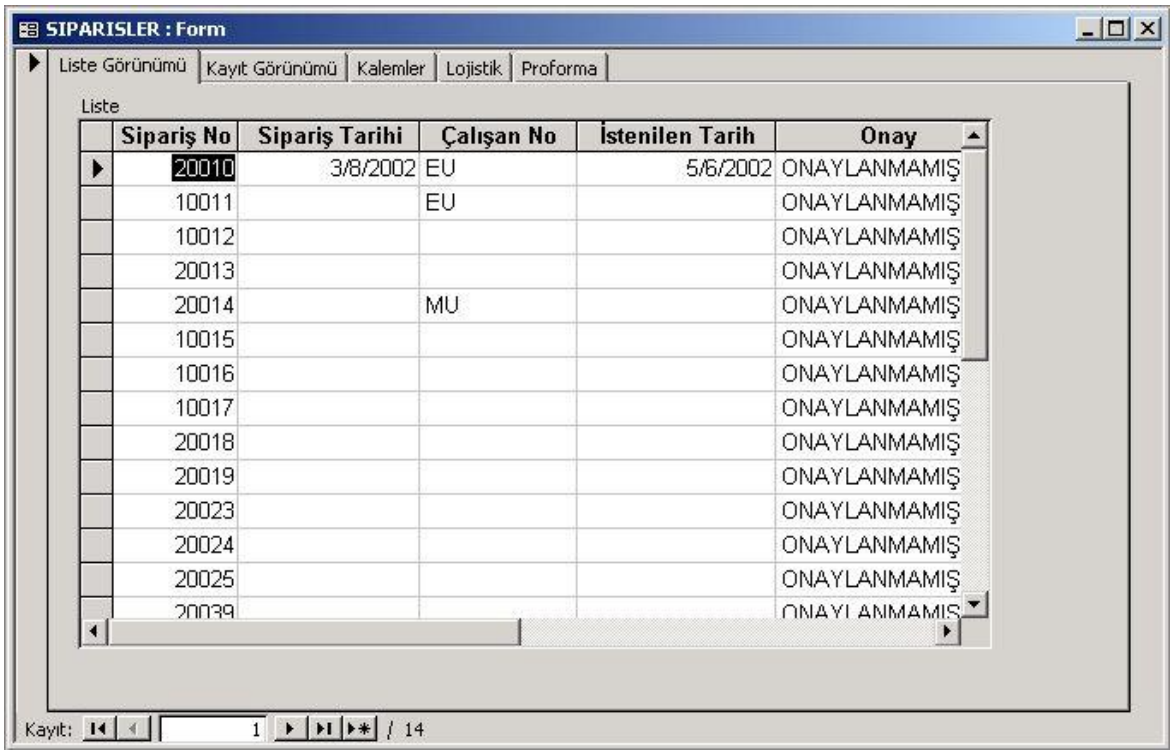
Bir sonraki ekran sipariş isteklerinden sipariş oluşturmaya yöneliktir. Üst tabloda onaylanmış sipariş istekleri, alt tabloda ise bu isteklere ait kalemler listelenmiştir. Siparişi oluşturan kişi herhangi bir sipariş isteğinin ister kabul kısmını seçerek o sipariş isteğine ait tüm kalemleri işaretler, isterse de kalemler arasından seçim yapabilir. "Kalemleri Görüntüle" düğmesi sadece seçili sipariş isteğine ait malzemelerin görüntülenmesini sağlar. "Tümünü Göster" düğmesi ise bu filtreyi temizler.



Şekil 3.4. Sipariş İsteklerini Siparişe Çevirme Ekranı

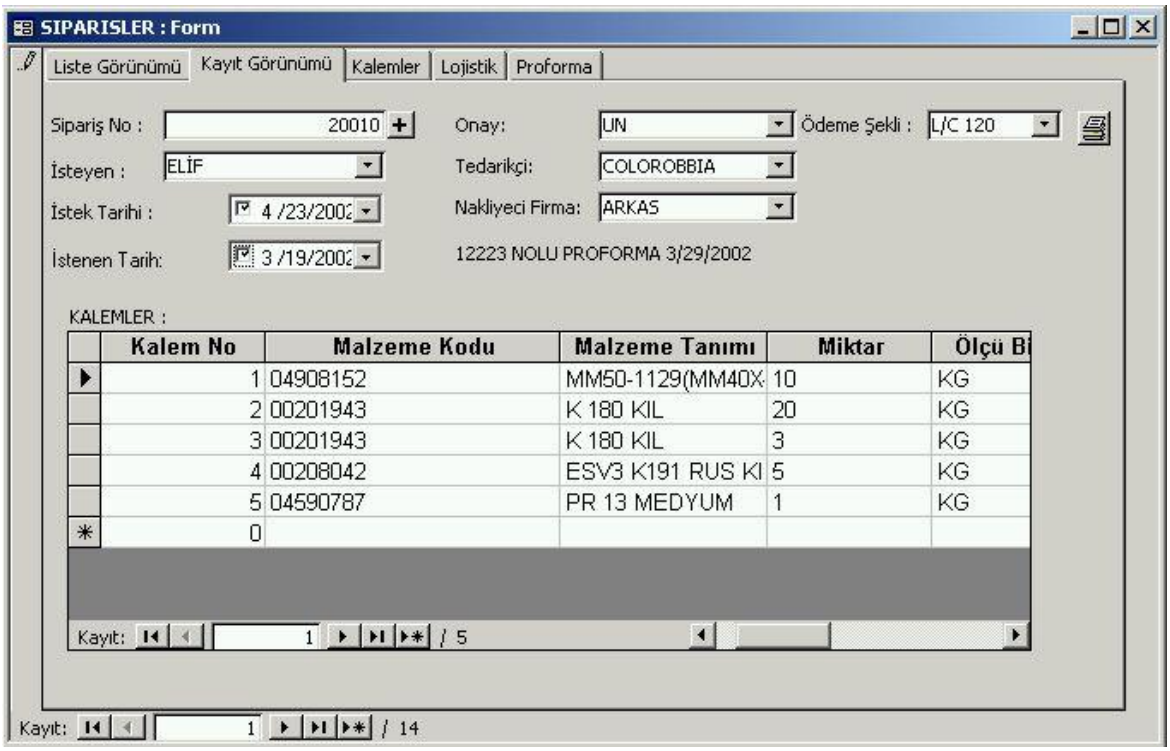
"Sipariş numarası al" düğmesine basılarak sipariş türü seçim ekranına geçilir. Sipariş numaraları akıllı kod sistemine dayanmaktadır. Bu kodlama eski işleyişi aksatmamak maksadıyla tasarlanmıştır. Zira mevcut işleyişte 1'le başlayan sipariş numaraları makine alımlarını, 2 ile başlayan sipariş numaraları hammadde alımlarını, 3 yedek parça, 4 işletme malzemesi ve 5 ile başlayanlar karo alımlarını temsil etmektedir. Sipariş numarası ilk rakamdan bağımsız olarak sıra ile verilmektedir. Örneğin küçükten büyüğe şöyle bir sıralama imkanlıdır : 10100,50110,40200,20500, vb. Alınan sipariş numarası "Sipariş numarası" alanına yazılır.

Bu ekrandaki son aşama "sipariş oluştur" düğmesine basılmasıdır. Bu düğmeye basılınca alınan sipariş numarası siparişler tablosuna eklenir ve seçili sipariş istekleri kalemleri, sipariş kalemleri tablosuna yazdırılır. Ardından eğer bir sipariş isteğine ait tüm kalemler seçilmiş ise sipariş istekleri tablosunda o isteğin durum alanı "tamamen siparişe çevrildi" olarak güncellenir. Bu durumda o sipariş isteği ve ona ait kalemler sipariş oluşturma ekranında görünmez. Eğer belirli bir sipariş isteğinin tüm kalemleri siparişe çevrilmezse, sipariş isteğinin durumu "kısmi sipariş" olarak güncellenir ve seçili kalemler listeden çıkarılır.



Şekil 3.5. Siparişler Ekranı-Liste Görünümü

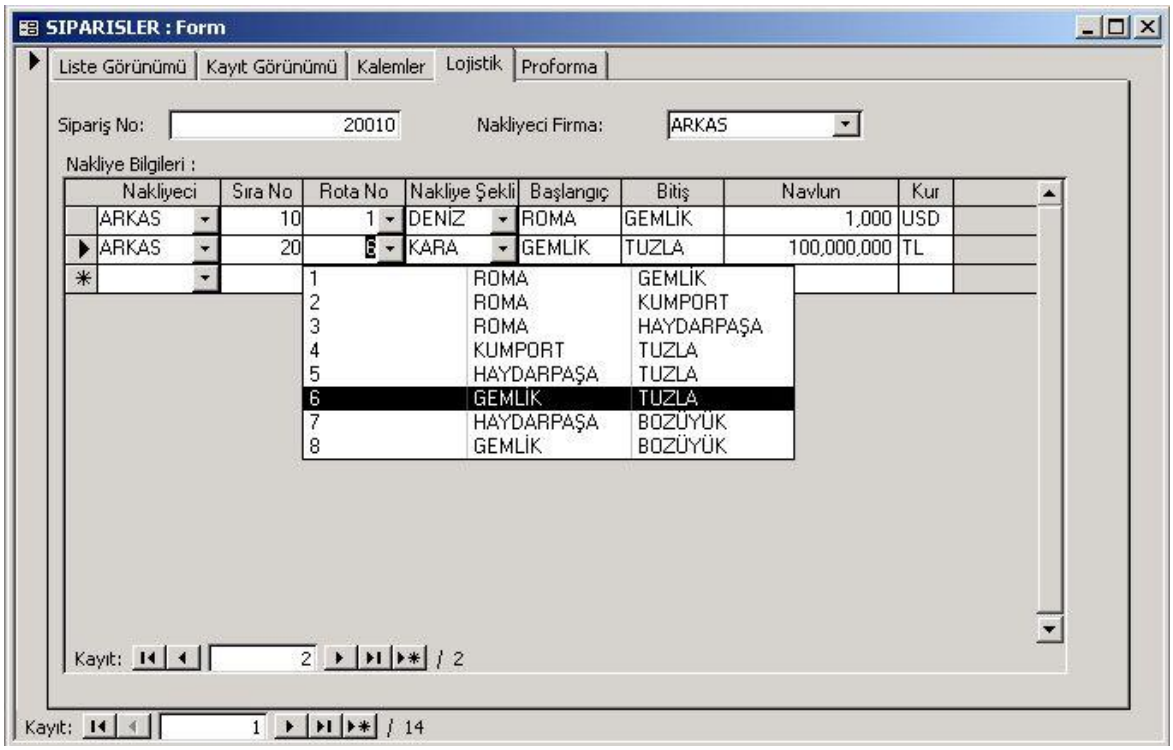
Sipariş ekranına geldiğimizde ise, oluşturduğumuz siparişin numarasını liste görünümünden seçerek kayıt görünümüne geçilir.



Şekil 3.6. Siparişler Ekranı- Kayıt Görünümü

Bu ekrandan sipariş için tedarikçi firma, nakliyeci firma ve istenen tarih belirlenir. "İstek tarihi" siparişin yaratılma tarihi iken "İstenen tarih" malzeme sevkıyatlarının talep edildiği tarihtir. Bu aşamada da satınalma müdürünün onayı söz konusu olduğu için ekranda "onay" alanı yaratılmıştır. Ödeme şekli sipariş bazında değişebildiğinden her sipariş için yeni bir ödeme şekli belirlemeye olanak sağlanmıştır.

Ayrıca siparişe ait proforma geldi ise proforma numarası ve tarihide bu ekranda gösterilir. Ekranın sol altındaki standart MS Access navigasyon tuşlarıyla bir önceki ve sonraki siparişlere gidilebilir. Sipariş numarasının yanında bulunan "+" tuşuyla da sipariş isteği olmaksızın sipariş açılabilir.

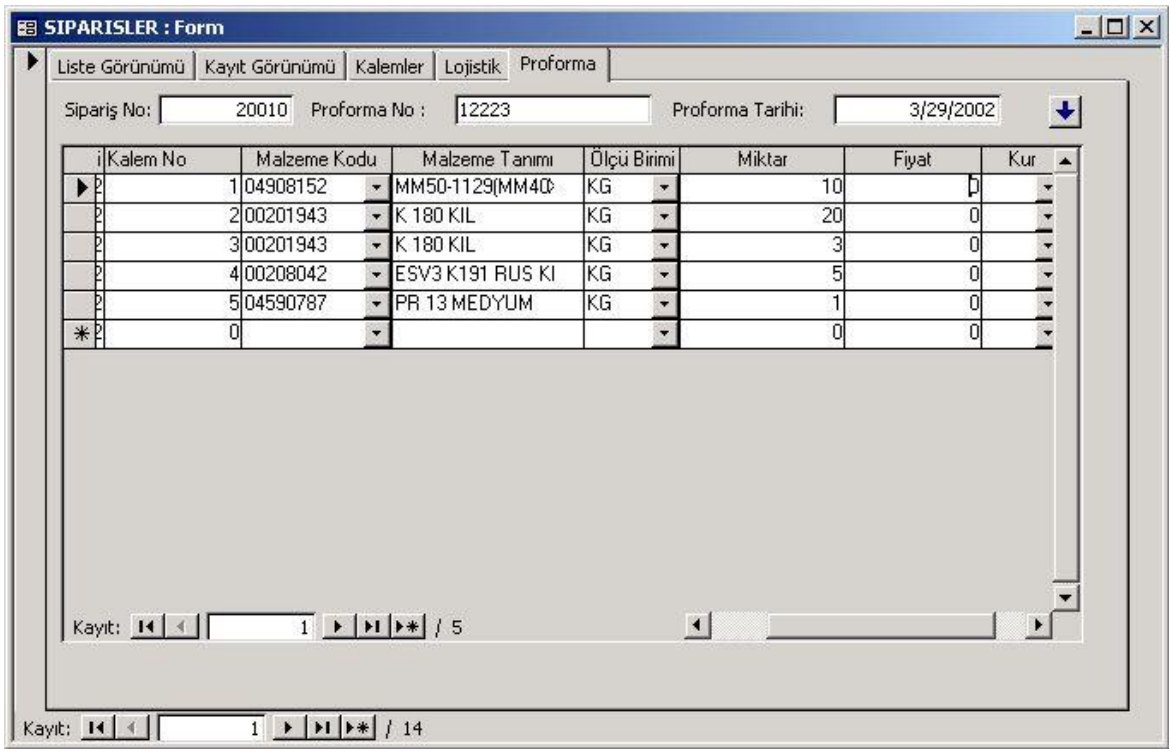


Şekil 3.7. Siparişler Ekranı- Lojistik

Lojistik ekranında ise seçili nakliyeci firmanın yüklemeyi nereden yapacağı ve malzemeleri hangi gümrüğe getireceği saptanır. Bu ekranın en güçlü özelliği kombine sevkıyata izin vermesidir. Burada tek koşul aynı nakliyeciyle olması şartıdır. Bu ekranda sevkıyatlar rota mantığı ile yapılmaktadır. Rota başlangıç ve bitiş noktalarından oluşur ve her nakliyecinin belirli rotalardaki navlunu sisteme kayıtlıdır. Nakliyeci, rota ve nakliye şeklinin seçimiyle mevcut navlun otomatik olarak ekrana gelir. İstenirse mevcut navlun bu ekrandan girilerek güncellenebilir. Bu sayede belirli bir malzemenin birim maliyeti daha rahat raporlanabilir. Bu özellik en çok teklif değerlendirmelerde kullanılmaktadır.

Malzeme ve Nakliye detayları girildikten sonra kayıt görününümünde üzerinde yazıcı resmi bulunan düğmeye basılarak sipariş ve nakliye emirleri basılır. Baskı faks server'a direkt gönderilebileceği gibi TIF formatında bir kopya e-mail aracılığıyla firmalara geçilebilir. Örnek sipariş emri çıktısı eklerde görülebilir (Ek A.4.).

Bu aşamadan sonra tedarikçi firmalardan proforma faturalarını göndermeleri beklenir. Gelen proformanın detayları aşağıdaki sipariş-proforma ekranına işlenir. Malzeme birim fiyat, miktar ve sevk tarihine ilişkin bilgiler güncellenir.



Şekil 3.8. Siparişler Ekranı - Proforma

Proforma numarası girildikten sonra kullanıcı isterse proformadaki kalemleri tek tek girebildiği gibi, isterse ekranın sağ üst köşesindeki düğmeye basarak o siparişe ait malzeme bilgilerini kopyalayabilir. Her yeni kayıt girişinden sonra fiyat ve/veya miktar ve/veya sevk tarihinde, sipariş detaylarından bir fark bulunursa o kayıt kırmızı bir arka plan rengiyle işaretlenir.

Proforması henüz gelmemiş siparişler raporu istenilen sıklıkla çalıştırılarak firmalara proformalarını göndermeleri için uyarı yazıları yazılabilir. Firmaların proformalarında belirttikleri sevk tarihlerinde sevkıyat yapıp yapmadıklarını kontrol etmek için, sevkıyatı beklenen malzemeler raporu çalıştırılır. Bu rapor doğrultusunda firmalarla iletişim kurularak sevkıyatların gerçekleşmesi sağlanır. Sevkıyatı gerçekleşen malzemeler için firmalardan gelen fatura bilgileri fatura ekranına girilir. Bir siparişe ait faturalar ise sipariş-fatura ekranından izlenebilir.

Teslim şekli CIF olmayan siparişler için sigorta yaptırmak zorunlu olduğundan sigorta talimatı oluşturulur. Buradaki tüm bilgiler otomatik olarak sistemden gelir. Bir tek istisna gemi ile taşıma durumu söz konusu olduğunda olur.

Bu durumda sistem sigorta talimatına basmak amacıyla kullanıcıdan gemi adını talep eder.

Sevkıyatı tamamlanan malzemeler siparişte belirtilen gümrüğe gelir. Satın alma elemanı gümrüğe geliş tarihini sisteme girer. Gümrüğe geliş tarihi işlenmiş her malzeme için planlama departmanları satın alma departmanından malzemenin gümrükten çekilmesini talep edebilir. Bu takip gümrüğe gelmiş malzemeler raporu ile yapılır.

Bu aşamadan sonra satın alma çalışanı ilgili gümrük komisyoncusuna iş emri vererek malzemenin gümrükten çekilmesi işlemine başlar. Gümrük komisyoncusu iş emrinde fatura, sevk, gümrük ve ödemeye ilişkin bilgiler yer alır.

Fiili ithalatı biten işlemler için gümrük komisyoncusundan gelen dekontlar da sisteme işlenir. Böylece iş emri verilmiş her işlem için dekontun gelip gelmediği de kolaylıkla takip edilebilir. Kambiyo kapatma talimatı da kendiliğinden oluşabilir. Zira, dosya no, gümrük adı, beyanname tarihi ve numarası gibi tüm gerekli bilgiler sistemde mevcuttur. Ayrıca bir çaba gerektirmeksizin çıktılar alınabilir. Bu çıktıların alınması maalesef bankaların tüm işlemlerinde halen ıslak imzada ısrar etmesinden kaynaklanan yasal bir zorunluluktur.

Ödeme vadelerinin takibi için de program aracılığıyla son derece etkin takip yapılabilmektedir. Zira, istenilen her an ödemelerin durumu ‘Firma Borçları Raporu’ veya ‘Aylara Göre Vadesi Gelen Borçlar Raporu’ çalıştırılarak gözlenebilir. Bu raporlar, özellikle kriz döneminde firmaya büyük katkı sağlamıştır. Ödemeler, sevkıyatları durdurmayacak ölçüde ve finansmanı zor durumda bırakmayacak şekilde ayarlanmıştır.

Bu raporlardan biraz daha bahsedecek olursak, birinci rapor yani firma borçları raporu için öncelikle gelen ilk ekrana firma adını yazıp ‘ok’ butonuna basılarak rapor çalıştırılır. Sonra ekte bir örneğini görebileceğiniz rapor kendiliğinden oluşur (Ek A.5.). Bu rapor, genellikle firmalar ile yapılan mutabakatlarda kullanılmaktadır. Ancak, bu rapordan, tedarikçi firmaların ödemelerine ilişkin sordukları bilgilerin karşılaştırılması ve onlara hangi tarihte ödeme yapıldığının veya hangi tarihte ödenmesinin planlandığının aktarılması amaçlı da yararlanılabilmektedir.

‘Aylara Göre Borç Toplamları Raporu’ ise ekrana girilen aralıkta, vadesi dolan ancak ödemesi tamamlanmamış borçları aylara göre listeleyerek aylık bazda toplamlarını almaktadır. Bu rapor ödeme planı yapmak amaçlı hazırlanmıştır. Finansmana sunulan raporlarda bu rapordan faydalanılmaktadır. Tüm borçları gösteren bu tabloya üstten bakabilmek nakit yönetimi için şarttır. Bu raporun bir örneği de eklerde bulunabilir (Ek A.6.)

Bu program sayesinde kolaylaşan bir başka raporlama da pareto analizleridir. Programdan önce prismden çekilen 8 aylık sarf raporlarının üzerine yine prismden çekilen birim fiyatlar MS Excel’de eşleştirilerek analizler yapılmaktadır. Ancak, artık tüm veriler yeni veriler sistem üzerinde olduğundan bu tip raporlamalar için ayrı bir çaba sarf etmek gerekmemektedir. Eskiden her fabrika için ayrı analiz raporları hazırlanırken şimdi üç fabrikanın stok seviyelerini bir kerede görmek mümkündür. Bu analizler, stok seviyelerinin optimizasyonu, stokta kalmış malzemelerin teknik müdürlük çalışmaları ile eritilmesi ve maliyetlerin düşürülmesi amaçlı hazırlanmaktadır. Analizler sonucunda ortaya çıkan maliyetlerin %80’ini oluşturan dilim için Satınalma ve Teknik müdürlük ortak çalışarak birim fiyatlarda indirimler alınmasına, alternatif malzemeler geliştirilmesine çalışmaktadır.

4. SONUÇ

Tüm canlı organizmalar doğar, büyür ve bir süre sonra ölür. Doğumdan ölüme dek geçen sürede, bireylerin kendini yenilemeleri, ihtiyaçlarına kulak vermeleri gerekmektedir. Demek istenilen, aslında tedaviye ihtiyacı olan bir organizmanın kendini yenilemesine fırsat verilmesi gerektiğidir. Tabi bu fırsatın da en iyi şekilde kullanılması şarttır. Hastalanan bir canlı için, tedavi ihtiyacının kabul edilmesi ve doğru tanının konulması ve tedavinin uygulamaya geçilmesi nasıl o canlı organizmanın ömrünün uzamasında büyük rol oynarsa; yazılımların da ömrünü bu tip bakımlarla uzatmak mümkündür.

EKS'deki çalışma gösteriyor ki, kullanılan program (prism) satınalma departmanı ihtiyaçları için bugün bakıldığında yetersiz kalmıştır. Elbetteki, şirketin ilk kurulduğu yıllarda, dosya sayısı 20'yi geçmiyor, 5'ten fazla akreditif açılmıyorken prism satınalma departmanı için yeterli, kolay kullanılabilen, şirket ihtiyaçlarına cevap verebilen bir yazılım olabilir. Ancak, bugün departmanda ayda ortalama 100 dosya açılmakta, 1500'ü aşkın malzeme siparişi verilmekte ve minimum 15 akreditif dosyası takip edilmektedir. Prism, işte bu noktada ihtiyaçları karşılayamaz duruma gelmiştir. Tüm bu olanlar göstermektedir ki, yazılımların da tıpkı canlı organizmalar gibi yaşam süreleri bulunmaktadır.

Prism, yeni sürümleri ile güncellenebilir. Ancak, Türkiye'deki ithalat mevzuatı göz önüne alındığında yeni sürüm bile akışı hızlandırmaya ve kapsamaya yetmemektedir. Bu durumda, prismin ithalatı desteklememesinin yanı sıra EKS'deki ithalat artışının da payı göz ardı edilemez. Zira, EKS, bugün için alımlarının %80'e yakın kısmını ithal etmektedir.

Yaşam döngüsünün sonuna geldiğine inanılan bir program ve sayısız ihtiyaç göz önüne alınarak şirket adına önemli bir karar verilmiştir: Yeni bir yazılım geliştirme. Bu karar verilirken öncelikle mevcut durum analizi yapılmıştır. Hangi noktalarda ve neden sıkıntılar yaşandığı saptanmaya çalışılmıştır. Bu çalışma için

departman bünyesinden ve bilgi sistemlerinden ortak çalışabilecek bir ekip kurulmuştur. Ekip üyeleri olarak satınalma işini iyi bilen ve ayrıca sistem konusunda deneyimli bireylerden seçilmiştir. Bu, projenin başarılı olmasındaki en temel unsurlardan biridir. Ekibin üyelerinin iyi seçilmiş olması beraberinde nitelikli sistem çözümlemeyi de getirmiştir. Birbirini iş ortamında da iyi tanıyan proje üyeleri arasında etkili ve başarılı iletişim kaçınılmazdır.

Sistem ihtiyacı belirtildikten ve de kurulacak sistemin neyi, nasıl yapması gerektiği belirlendikten sonra bu sistemin yapısı, parçaları, parçaların birbiriyle olan ilişkisi ortaya konulmuş, kısaca sistemin nasıl kurulacağı tasarlanmıştır. Bu aşamada da diğer aşamalarda olduğu gibi sistem analistleri kullanıcılar ile birlikte çalışmışlardır. Tüm yeni ekranlar kullanıcılar ile birlikte hazırlanmıştır.

Bahsi geçen her bir ekran için, programcı kodu yazdıktan sonra bir diğer programcı veya programcılar tarafından sınamaya geçirilmeden hemen önce gözden geçirilmiştir. Bu çalışmanın hata bulma ve ayıklamada büyük katkıları olmuştur. Sınama aşamasına gelen programın öncelikle her bir birimi ayrı ayrı ele alınmış ve olası her bir işlem en az bir kez sınanmıştır. Her birimin sınanması aşaması başarı ile sonuçlanmasının ardından ortaya çıkan yazılımın tasarıma uygun olup olmadığı araştırılmıştır. Ne de olsa, tasarım çok iyi yapıldığı ve yazılımın tasarıma uygun bulunduğu takdirde yeni yazılımın, müşteri taleplerini bütünüyle cevaplandırması en çok beklenen durumdur.

Bu uygulamada, bilgi sistemleri yaşam döngüsü modellerinden sadece biri ele alınıp projeye baz teşkil eden yaklaşım olarak benimsenmemiştir. Aksine, en büyük başarıyı getireceğine inanılan karma yöntem kabul edilmiştir. Benimsenen bu yönteme göre, çağlayan yönteminde olduğu gibi yapay bir doğrusallık bulunmamaktadır. Tasarımı tamamlandı uygulamaya geçilir ve bir daha tasarıma geri dönülemez gibi bir durum asla söz konusu değildir. Zira, her aşamanın içinde ve aşamalar arasında dönüşler kaçınılmazdır.

Ofislerde, yeni yazılımların, kullanıcıların programa olan yabancılıklarından ve hatta güvensizliklerinden kaynaklanan verimlilik kaybına neden olduğu düşünülür. Bir süre eski sistem ile birlikte çalıştırılarak güven sağlamak amaçlanır. Ancak bu yöntem, çalışanların eski alışkanlıklarını yıkmak için zorlayıcı değildir. Kullanıcı her ne kadar kendini yeni programa yakın hissetse de eski sisteme kayıt

yapma isteği ağır basar. Bu nedenle modülün yavaş yavaş kullanıcılara benimsetilmesi gerekmektedir. Bu yaklaşımdan hareket ederek, satınlama departmanının nakit akışını daha etkin yönetebilmesi için ödemeler modülü son kullanıcılar ile programın tamamı uygulamaya geçirilmeden tanıştırılmış olup halen sorunsuz bir şekilde kullanılmaktadır.

Yeni yazılımın kullanıma geçirilen modülleri için bir kereye mahsus veri aktarımı yapılmış ve eski veri tabanının terk edilmesi sağlanmıştır. Böylece, birden fazla veri tabanı yaratmaktan kaynaklanan zaman kayıplarının büyük ölçüde önüne geçilmiştir. Veri aktarımı yapılmadan önce tüm verilerin doğruluğu kontrol edilmiştir. Bu işlemdeki amaç, yeni modülde hataların devam ettirilmemesidir. Ayrıca belirtilmesi gereken bir konu da tek veri tabanında kayıt tutmanın hata payını azalttığıdır.

Sonuç olarak bu program aracılığıyla etkin satın almalar yapılması, tedarikçi firmalar ile uzun süreli ilişkilerin sağlıklı yürütülebilmesi için bir ortam yaratmak amaçlanmıştır. Elbette ki, geliştirilmesi, iyileştirilmesi gerekmektedir. Ancak program, bu tür kullanıldıkça ortaya çıkacak ihtiyaçlar için hazırlanacak ekranlara, raporlara her zaman izin verebilecek şekilde hazırlanmıştır. EKS için planlanmış olmasına rağmen birçok şirkette ithalat akışını başarıyla götürebilir. Amaç, en başından beri akıştaki darboğazları ortadan kaldırmak, iş tekrarlarını ve zaman kayıplarını önlemek ve ofisteki kağıt kullanımını en aza indirmek olduğundan çalışmanın başarıya ulaştığı düşünülebilir. Zira, kağıtsız ofis kavramının, ithalat süreçleri daha doğrusu Türkiye'deki ithalat kanunları ile pek bağdaşmayan bir yapısı vardır. Gümrükçü dekontları, faturalar, ıslak imza gerektiren banka evrakları yapının tamamen bilgisayar ortamına taşınamamasının başlıca nedenlerindendir.

Programın tamamı ofiste henüz kullanılmadığı için tamamen başarıya ulaşmıştır ya da başarısızdır demek için erkendir. Ancak, çalışanlar, ödeme tabloları, pareto analizleri, gümrükçü performansı için kullanılan kısımlarından memnuniyetlerini dile getirmektedir. Raporlamanın son derece kolay ve sınırsız olması programın kullanım isteğini arttırmakta ve tamamının kullanılmasıyla ithalat sürecinin daha kolaylaşacağına inanılmaktadır. Planlamalara bilgi akışının kolaylaşacağı ve kağıt kullanımının %70-80 ölçüsünde azalacağı tartışılmaz gerçeklerdir. Düne kadar, prismde sadece gümrük lokasyonundaki malzemeleri

planlamadakiler görürken artık her bir malzeme için siparişte, yolda, gümrükte, işlemde vb. pozisyonların takibi ve performanslarının ölçümü yapılabilmektedir.

Yeni yazılım aracılığıyla süreçteki tüm iş tekrarları, veri ve zaman kayıpları ortadan kalkacaktır. Çünkü, artık hiçbir bilgi farklı MS Excel dosyalarına ya da kağıtlara defalarca kaydedilmeyecektir. Tek bir veri tabanı üzerinden sağlıklı bilgilere ulaşmak ve onlar için raporlar hazırlamak mümkün olacaktır.

Yoğun rekabetin yaşandığı günümüzde, farklılık yaratanların pastadan daha büyük payı kapmaları kaçınılmazdır. Üretim teknolojilerinin hemen hemen benzeştiği seramik sektöründe maliyetlerde yaşanan farklılıklar, şirketlerin karlılıklarını doğrudan etkilemektedir. Bu durumda, maliyetlerin çok etkin bir şekilde yönetilmesi gerekmektedir.

EKS Satınalma departmanının ihtiyaçlarına cevap araması, aslında, şirket hedeflerine ulaşmada büyük bir adım sayılabilir. Zira, yeni yazılım ile daha etkin satın almalar gerçekleşecektir. Yeni yazılımın yaşam döngüsü sürecini bakımlar ile destekleyerek, yeni ekranlar ekleyerek, zamanla tedarikçilere doğrudan internet bağlantıları sağlayarak uzatmak, yazılımı daha verimli hale getirmek mümkündür. Bu sebeple, yeni yazılım, daha uzun yıllar EKS hedeflerine hizmet etmek amacıyla kullanılabilecektir.

KAYNAKÇA

Adamany, H.G. ve Gonsalves, F.A.J., 1994. Life Cycle Management: An integrated approach to managing investments, *Journal of Cost Management,* **Summer,** 35-48.

Basili, V.R., Selby,R.W ve Hutchens, D.H., 1986. Experimentation in Software Engineering, *IEEE Trans.Software Engineering*.**July,** 733-743.

Boehm, B., 2000. Spiral Development : Experience, Principles, and Refinements, *USC Spiral Experience Workshop*, California, USA, **February,** 9-11.

Boehm, B. and Ross, R., 1989. Theory W Software Project Management: Principles and Examples, *IEEE Trans.Software Engineering*, **July,** 902-916.

Cavaye, A., 1995. User Participation in System Development Revisited, *Information & Management*, Vol:28, No: 5, **May** 1995

Finkelstein, A., Kramer, J., Nusibeh, B. ve Finkelstein, L., Goegicke. M., 1991.Wiewpoints: A Framework for Integrating Multiple Perspectives in System Development. *International J. Software Engineering and Knowledge Engineering.* **March.** 31-58

Forsberg, K., Mooz, H. ve Cotterman, H., 1996. Visualizing Project Management, John Wiley & Sons, USA.

Genuchten, M.V., 1991. Why is Sofware Late?, *IEEE Trans. Software Engineering,* **17:6,** 582-590.

Gov. Of Hong Kong, 2002. An Introduction to RAD, *Special Administrative Region*, **Febraury**, 5-30

Hart, D., 2002. Information Systems Analysis, INFS 2024, *Rapid Application Development & Course Review*.

IEEE/ANSI 830. 1988.IEEE Recommended ,Practice for Sofware Requirements Specifications IEEE Software Engineering Standards.IEEE Press.

J.D.Blacburn. 1996. Improving Speed and Productivity of Software Developments. A Global Survey of Software Developers.*IEEE Trans Software Engineering.* **22:12.** 875-885

Kitaoka, B., 2000. Yesterday. Today & Tomorrow: Implementations of the Development Life Cycles, *USC Spiral Experience Workshop,* California, USA, **February,** 9-11

Koh, S. ve Heng, M., 1996. users and Designers as Partners: Design Method and Tools for User Participation and Designer Accountability Within the Design Process, *Information System Journal*, Vol:6, No:4, October 1996.

Kusters, R., Cowderoy A., Heemstra F. ve Veenendaal E., 1999. Project Control for Software Quality, Shaker publishing, California. USA.

Landay, J., 2001. Software Life Cycle, CS 169: *Software Engineering*, **Spring**.

Ling, C., 2002, Software Life Cycle Model, Chapter 3, CSE 1401.

Maner, W., 1997. Rapid Application Development, **March**-15

Martin, J., 1991. Rapid Application Development, Macmillan Inc., NewYork.

MIS, 2001, Design Review Team Procedures, *Management Informations Systems*

Parnas, D.L. ve Clements. P.C., 1987. A Rational Design Process:How and Why to Fake It. *IEEE Trans .Sofware Engineering.***12:2.** 251-257.

Rockstrom,A. Ve Saracco,R., 1982. SDL-CCITT Specification and Description Language, *IEEE Trans SoftwareEngineering.***30:6.** 1310-1318.

Taggart, M., 2001, RADical Software Development, Aplication Development Advisor, *Software Engineering*, **December** 2001.

Thorp. J., 1998. The Information Paradox , McGraw Hill. New York.

W. E. Royce, 1998**.** "Software Project Management: A Unified Framework". Addison Wesley.

Zave,P., 1982 . An Operational Approach to Requirements Specification for Embedded Systems.*IEEE Trans* . *Sofware Engineering.* **8:3.** 250-269.

EKLER

EK A.1. **:** Dosya Takip Örneği

EK A.2. **:** Gümrük Dosyası Örneği

EK A.3. **:** İş Akış Şeması

EK A.4. **:** Sipariş Yazısı Örneği

EK A.5. **:** Firma Borçları Raporu Örneği

EK A.6. **:** Aylara Göre Borç Toplamları Raporu Örneği

GÜMRÜK DOSYASI

SAYFA NO

DOSYA NO	FİRMA ADI	MALZEME TANIMI	MİKTARI	SEVK BİLGİSİ	FATURA NO/TARİHİ	SİGOR-TA	ÖDEME ŞEKLİ	VARIŞ GÜMRÜĞÜ	VARIŞ TARİHİ	GÜMRÜK KOM.ADI/ VERİLİŞ TARHİ	FİİLİ İTHAL TARİHİ	BEYANNAME TARİH/NO
	EURO											
29625	365	PG 96	150 KG	34 TC 4642	2.1.2002	✓	MM	ERK	2.12.2002	ÖREN	15/2/02	13/2/02
	COLOROBBIA				625	✓				13/2/02		7566855
				BORGHI INT.								
	EURO											
28522	150	510/001	25 KG		2.3.2002	✓	VM	ESK	15/2/02			
	CEREV				102	✓						
				BORGHI INT.								
	USD 6500											
49000		r-006 t9	2000 AD	MCS	2.6.2002		LC					
	JAVITRADE			SARISKA	65850		120 G.					
				ARKAS								

EKS Eczacıbaşı Karo Seramik
San.ve Tic.A.Ş.

E-5 Karayolu üzeri, Atatürk Cd.
Şifa mah. 81700,Tuzla İstanbul
Tel : +90 216 423 46 00
Fax: +90 216 423 47 10

Purchase Order Number	:	20010
Purchase Order Date	:	5/3/2002
Vendor Name	:	COLOROBBIA
Address	:	41049 Sovigliana
Attention	:	Ms.Ariella
Payment Terms	:	L/C 120 days
Delivery	:	By Vessel

Please, send us the following goods, transmit your proforma invoice and inform us of your exact delivery date. These goods are for our Bozüyük factory. Please do not forget to label all the packages as Bozüyük.

Best Regards,

Elif Uysal

Our Transpoter
Tarros
Tel: +39 536 819211
Fax: +39 536 800096

DUE DATE	CODE	DESCRIPTION	U.M	QUANTITY	U.P	TOTAL
5/4/2002	04908152	MM50-1129(MM40X-0059M)RF 60124	KG	1000	0,5	500
5/4/2002	00201943	K 180 KIL	KG	2000	0,2	400
5/4/2002	00208042	ESV3 K191 RUS KILI	KG	500	0,1	50
5/4/2002	04590787	PR 13 MEDYUM	KG	100	2,1	210

TOTAL(EURO) : 1160

FİRMA BORÇLARI - CERDEC

FATURA TARIHI	SIPARIS NO	P.B.	TUTAR	FATURA NO	ODEME_VADESI	ODEME TARIHI	PLANLANAN TARIH
12/21/1999	27286	LIT	23299000	E/01264	4/19/2000		
12/21/1999	27256	LIT	25356000	E/01264	4/19/2000		
5/17/2000	27783	LIT	28690800	E/00603	9/14/2000	3/30/2001	
5/17/2000	27614	LIT	4850000	E/00603	9/14/2000	3/30/2001	
5/17/2000	27678	LIT	13800000	E/00603	9/14/2000	3/30/2001	
5/31/2000	27783	LIT	16756750	E/00695	9/28/2000	5/7/2001	
6/27/2000	27886	LIT	2007900	E/00829	10/25/2000	6/6/2001	
7/6/2000	27524	LIT	4052000	E/00894	11/3/2000	6/8/2001	
7/6/2000	27678	LIT	6078000	E/00894	11/3/2000	6/8/2001	
7/6/2000	27728	LIT	12156000	E/00894	11/3/2000	6/8/2001	
7/6/2000	27886	LIT	16208000	E/00894	11/3/2000	6/8/2001	
7/6/2000	27824	LIT	2026000	E/00894	11/3/2000	6/8/2001	
7/24/2000	28028	LIT	3100000	E/00980	11/21/2000	6/8/2001	
8/2/2000	27753	LIT	4055000	E/01062	11/30/2000	6/8/2001	
8/2/2000	27783	LIT	283850	E/01062	11/30/2000	6/8/2001	
9/5/2000	28028	LIT	19008500	E/01144	1/3/2001		
9/5/2000	28060	LIT	15500000	E/01144	1/3/2001		
9/5/2000	28121	LIT	11761250	E/01145	1/3/2001		
9/5/2000	28093	LIT	9409000	E/01145	1/3/2001		
9/14/2000	27824	LIT	1256000	E/01180	1/12/2001	7/10/2001	
9/14/2000	27886	LIT	12560000	E/01181	1/12/2001	7/9/2001	
9/14/2000	28165	LIT	6200000	E/01181	1/12/2001	7/9/2001	
10/4/2000	27886	LIT	17250000	E/01276	2/1/2001		
11/3/2000	28332	LIT	12400000	E/01406	3/3/2001		
4/19/2001	28688	EUR	1250	E/446	8/17/2001		
4/19/2001	28384	EUR	1408	E/446	8/17/2001		
6/7/2001	28966	EUR	607.5	E/690	10/5/2001	6/27/2001	
6/20/2001	28990	EUR	982.5	E/736	10/18/2001	7/17/2001	

FATURA TARIHI	SIPARIS NO	P.B.	TUTAR	FATURA NO	ODEME_VADESI	ODEME TARIHI	PLANLANAN TARIH
7/6/2001	29080	EUR	108.5	E/827	11/3/2001		
7/6/2001	29057	EUR	7716	E/830	11/3/2001		
7/13/2001	29106	EUR	1600	E/874	11/10/2001		
7/20/2001	29106	EUR	1215	E/921	11/17/2001		

AYLARA GÖRE BORÇ TOPLAMLARI RAPORU

FİRMA	*SİPARİŞ NO*	*TUTAR*	*PARA BİRİMİ*	*FATURA NO*	*FATURA TARİHİ*	*ÖDEME VADESİ*
COLOROBBIA	28630	5780	EUR	21000289	2/27/2001	6/27/2001
COLOROBBIA	28698	375	EUR	200216	3/2/2001	6/30/2001
COLOROBBIA	28632	6310.9	EUR	200216	3/2/2001	6/30/2001
COLOROBBIA	27890	1084.5	EUR	200216	3/2/2001	6/30/2001
JOHNSON MATTHEY	28631	17746	EUR	20150128	2/20/2001	6/20/2001
JOHNSON MATTHEY	28585	9955.5	EUR	20150128	2/20/2001	6/20/2001
LUNA ABRASIVI	48451	42059000	LIT	1419/00	12/20/2000	6/18/2001
PEM	28564	12644.95	FF	61	2/16/2001	6/16/2001
TG MAC	38617	1449000	LIT	57	2/16/2001	6/16/2001
TG MAC	38637	3390960	LIT	57	2/19/2001	6/19/2001
TG MAC	38617	552000	LIT	56	2/19/2001	6/19/2001
TG MAC	38571	9213745	LIT	56	2/19/2001	6/19/2001
TORRIANA	28626	24117000	LIT	117/E	2/16/2001	6/16/2001
June 2001 Toplamı (EUR)						*84,899.88*
DAM	28742	26028	FF	2000470	4/3/2001	7/2/2001
DAM	28686	52056	FF	2000470	4/3/2001	7/2/2001
July 2001 Toplamı (EUR)						*11,903.83*
Genel Toplam						*96803.7081299*

ÖZGEÇMİŞ

1976 Yılında Ankara'da.dünyaya geldim. İlk öğrenimimi Çorlu, Uncular Süleyman Peker İlkokulu'nda, orta öğrenimimi Eskişehir Anadolu Lisesi'nde tamamladım. 1994-95 öğretim yılında lisans öğrenimime başladığım İ.T.Ü. İşletme Mühendisliği bölümünden 1998 yılı Haziran ayında mezun oldum. 1998-99 öğretim yılında ise Fen Bilimleri Enstitüsünün İ.T.Ü.İşletme Mühendisliği Anabilim Dalı İşletme Mühendisliği Yüksek Lisans Programına kayıt oldum.