Öneri.C.7.S.27.
Ocak 2007.171-176.

BİLİŞİM SİSTEMLERİNİN GELİŞTİRİLMESİNDE DOĞRU YAŞAM DÖNGÜSÜ MODELİNİN SEÇİMİ

Yücel YILMAZ

Marmara Üniversitesi, İ.İ.B.F, Almanca İşletme Enformatiği Bölümü, Dr.

CHOOSING THE RIGHT LIFE –CYCLE MODEL FOR THE DEVELOPMENT OF INFORMATION SYSTEMS

Abstract: *Information Systems provide new possibilities to companies in different areas in time, cost and quality aspects, nevertheless many information system development projects could not result successfully. The life-cycle models, that aim the development of information systems as possible as with fulfilled requirements and economical use of resources, influence with their pros and cons the success of the system development activities. Choosing the right model facilitates the realization of processes on the basis of objectives and increases the benefits obtained from the intensive investments in this area, provides more uses from intensive investments in this area. In this article main life-cycle models used in the development of information systems are introduced and opportunities and threats provided by these models are evaluated in relation to requirements of information systems to be developed.*

Keywords: *Life-cycle Models, Information Systems, System Development.*

BİLİŞİM SİSTEMLERİNİN GELİŞTİRİLMESİNDE DOĞRU YAŞAM DÖNGÜSÜ MODELİNİN SEÇİMİ

Özet: *Farklı sektörlerde faaliyet gösteren her türlü kurum ve kuruluşta kullanılan bilişim sistemleri, zaman, maliyet, kalite alanında önemli kazanımlar sunmakta, buna karşın uygulamada bilişim sistemi geliştirme projelerinden önemli bir kısmının başarısızlıkla sonuçlandığı görülmektedir. Sistemlerin mümkün olduğunca gerekliliklere uygun şekilde ve ekonomik olarak geliştirilmesini hedefleyen yaşam döngüsü modelleri, sundukları çeşitli olanaklar ve riskler ile bilişim sistemi geliştirme çalışmalarının başarısını önemli oranda etkilemektedir. Doğru modelin seçimi, süreçlerin hedeflere uygun şekilde gerçekleştirilmesini kolaylaştırmakta, bu alanda yapılan yoğun yatırımların sağladığı faydaları artırmaktadır. Bu çalışmada, bilişim sistemlerinin geliştirilmesinde kullanılan başlıca yaşam döngüsü modelleri tanıtılmakta, bu modellerin sundukları çeşitli olanaklar ve riskler, geliştirilecek bilişim sistemlerinin ihtiyaçları bağlamında değerlendirilmektedir.*

Anahtar Kelimeler: *Yaşam Döngüsü Modelleri, Bilişim Sistemleri, Sistem Geliştirme.*

I. GİRİŞ

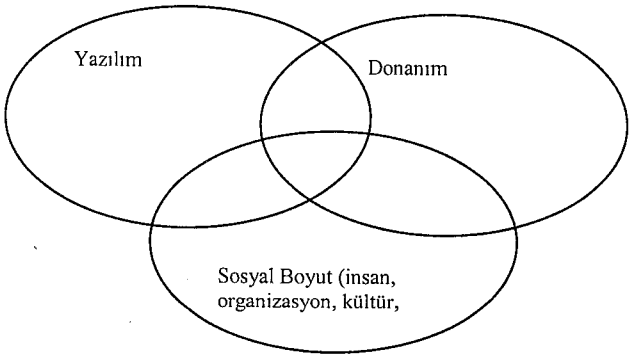
Bilişim teknolojilerindeki hızlı ilerlemeler, her geçen gün hayatımıza yeni kavramlar katmaktadır. Birkaç yıl öncesine kadar kimsenin bilmediği veya çok az kişinin kullandığı kavramlar, günümüzde milyonlarca kişi tarafından kullanılır hale gelmiştir. Veri, enformasyon, bilgi, bilişim gibi kavramlar farklı araştırmacılarca farklı bakış açılarına göre tanımlanmaktadır. Bilgi ve iletişim kavramlarını biraraya getiren bilişim kavramı sistem kavramı ile birlikte kullanıldığında, sadece teknolojiyi değil, insan, organizasyon, kültür ve süreç boyutlarını da kapsamaktadır. Bilişim sistemlerinin verimli şekilde yönetiminde, bu unsurların tümünün önemli rol oynadığı açıktır ancak etkin bir bilişim sisteminin oluşturulması öncelikle teknik çalışmaların başarılı şekilde gerçekleştirilmesini gerektirmektedir.

Bilişim sistemi kavramının literatürdeki tanımlarından biri şu şekildedir: İşe ait veri işleme gerekliliklerini karşılamak; yönetime, planlama, kontrol ve karar verme etkinliklerinde destek olmak için bilgi sağlamak, ihtiyaç duyulduğunda kullanılmak üzere dışsal ve içsel koşullarla ilgili raporlar hazırlayarak, veri işleme operasyonlarını yapan bileşenlerin sistematik ve formel olarak biraraya getirilmesinden meydana gelen sistemlere "Bilişim Sistemleri" denilmektedir [1]. Bu tanımda da görüldüğü gibi bilişim sistemleri öncelikle veriyi işlemekte, veriyi çeşitli işlemlerden geçirerek (Örnek: Sınıflandırma, ayırma, birleştirme, hesaplama, özetleme vb.) enformasyonlara dönüştürmektedir [2]. Bilişim sistemleri, oluşturulan enformasyonları çeşitli şekillerde (Örnek: Raporlar aracılığıyla) sunmakta, bu sayede hem çalışanlara hem de yönetim kademesine alacağı kararlarda önemli oranda yol göstermektedir.

II. BİLİŞİM SİSTEMLERİNİN GELİŞTİRİLMESİNDE MODELLERİN GEREKLİLİĞİ

Bilişim sistemlerinin temel unsurlarını donanım ve yazılım oluşturmaktadır. Donanım kavramı, mikro işlemcilerden monitörlere, sabit disklerden transistörlere kadar birçok kavramı içermektedir. Donanım ürünleri arasında yer alan transistörlerin ve mikro işlemcilerin üretimleri oldukça zor şekilde gerçekleşmekte, üretim süreçleri masraflı ve yorucu olmaktadır. Bu ürünler ancak

çok miktarda üretildiklerinde masraflarını karşılayabilen ürünlerdir. Buna karşın hemen hiçbir sektör, faaliyetlerinde çok sayıda transistöre ve mikro işlemciye ihtiyaç duymamaktadır. Bu ürürünlerin daha fazla sayıda üretilebilmesi ancak diğer sektörlerde de transistör ve mikro işlemci kullanımının yaygınlaşması ile olmuştur. Söz konusu yaygınlaşmada ana rolü, donanım parçalarını farklı alanlardaki ihtiyaçlara uygun şekilde kullanılabilir kılan yazılımlar oynamıştır. Böylece donanım ve yazılım birbirinden ayrılmaz iki unsur olarak karşımıza çıkmıştır.



Şekil.1. Bilişim Sistemlerinde Başlıca Boyutlar

Yazılımları donanımlardan ayıran diğer özellikler aşağıdaki şekilde belirtilebilir [4]:

- Yazılımların bakım zamanları, somut ürünlerdeki gibi sabit koşullara bağlanamaz, çünki yazılımlarda "aşınma" söz konusu olmamaktadır,

- Geliştirilen yazılımın hangi aşamada olduğu maddi ürünlerde olduğu gibi kolayca belirlenememektedir,

Gerek donanımların gerekse yazılımların geliştirilmesi, mühendislik faaliyetlerini gerektirmiştir. Donanım mühendislikleri ile yazılım mühendisliği arasındaki en önemli farkı, ortaya konan ürün oluşturmaktadır. Donanım mühendislikleri maddi ürünlerin üretimiyle ilgilenirken, yazılım mühendisliği fikri ürünlerle ilgilenmiştir, çünkü yazılım "fikri" bir üründür. Dolayısıyla, maddi bir ürün için gereken kaynak tanımlama ve kontrol metodları, burada tam olarak uygulanamamaktadır. Müşterilerin ne istediklerini tam olarak tanımlayamaması, sistemin tamamen oluşturuluncaya kadar ne geliştiriciler ne de müşteriler tarafından tam olarak bilinmemesi, yazılım geliştirme sürecinin en önemli sorunları arasındadır. Başka bir deyişle, yazılım insan zekasının bir ürünüdür. Yazılımı üretmek için oluşturulması gereken entelektüel değer üretimi, geleneksel üretim yöntemleriyle doğrudan karşılaştırılabilecek türden bir unsur değildir [3].

- Yazılımın geliştirilmesi fiziksel kanunlarla sınırlandırılamaz, bu yüzden geliştiricilere önemli şekillendirme olanakları vermektedir.

- Yazılımların yedek parçası yoktur, maddi ürünlerde arızalı parçalar kolayca değiştirilebilirken, yazılımlarda sistemin tekrar oluşturulması gerekmektedir.

- Yazılımlar çok hızlı yaşlanmaktadır. Bu özellik, yazılımlarda aşınma olmaması özelliğiyle çelişkili görülebilir ancak yaşlanma kavramıyla, yazılımın kullanıldığı ortamdaki koşulların değişmesinden bahsedilmektedir. Değişen koşullara uyum sağlayayan yazılımlar kullanıcılar tarafından tercih edilmemekte, böylece yazılım geçerliliğini ve güvenilirliğini kaybetmektedir.

- Teknik ürünler ölçüm aletleriyle mükemmel şekilde ölçülebilmektedir. Böylece ürünler standartlarla ve örneklerle karşılaştırılabilmektedir. Buna karşın yazılımlarda bu değerlendirmeler sınırlı oranda mevcuttur. Yazılımın kalitesi zorlukla belirlenebilir ve sayısallaştırılabilir. Ayrıca ölçülen özellikler ile kalitenin gerektirdiği özellikler arasındaki fark sadece yaklaşımlarla tanımlanamamaktadır.

Yukarıda belirtilen özellikler, yazılımların geliştirilmesinde modellere neden ihtiyaç duyulduğunu ortaya koymaktadır. Yazılımın fikri bir ürün olması, bu alanda modellerin kullanılamayacağı şeklinde yorumlanmamalıdır, çünkü hedeflenen bir kerelik değil, yinelenen başarıdır. Ayrıca yazılımın bir kere geliştirildikten sonra unutulabilecek bir ürün olmadığı da göz önünde bulundurulmalıdır. Yapılan hemen tüm araştırmalar, yazılım geliştirme projelerinde en fazla zaman, emek ve dolayısıyla para harcanan sürecin "bakım – onarım süreci" olduğunu ortaya koymaktadır [4].

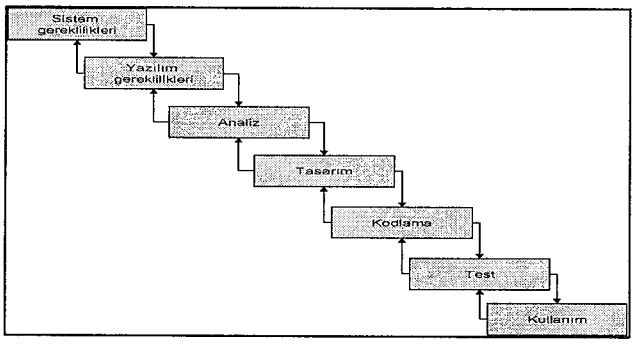
Bütün bunların dışında, bilişim sistemlerinin temel unsurlarından olan yazılımlar, her geçen gün fonksiyonlarını ve büyüklüklerini artırmaktadır. Müşterilerin yeni özellikleri talep etmeleri, üreticilerin müşterilere daha kaliteli ürünler sunma çabaları (özellikle ara yüzler ve on-line yardım menüleri temelinde) yazılımların donanımsal kaynak ihtiyaçlarını artırmaktadır. Yeni versiyonlar daha güçlü mikro işlemcilere ihtiyaç duymakta, yazılımların sabit disklerde kapladığı alan gittikçe artmaktadır. Bu gelişmeler, yazılım geliştirme projelerinin başarısını güçleştirmektedir, çünkü programcılara duyulan ihtiyaç, dolayısıyla masraflar artmaktadır. Ayrıca yazılımların yapısındaki fonksiyonel değişiklikler de göz önünde bulundurulmalıdır. Önceleri bir ana programdan ibaret olan yazılımlar, ihtiyaçların farklılaşması üzerine çeşitli parçalara (modüllere) ayrılmıştır. Böylece temel problem, farklı birimler tarafından kullanılan modüllerin entegrasyonu olmuştur. Yazılımlar başta yalnızca kullanıcılar için destek dokümanları sunarken, günümüzde geliştiriciler için de teknik dokümanlar sunmaktadır. Özetle, yazılım kavramının kapsamı geçmişe göre oldukça artmıştır [3].

III. YAŞAM DÖNGÜSÜ MODELLERİ

Bilişim sisteminin başlıca unsurlarından biri olan yazılımların geliştirilme süreçleri bir yaşam döngüsü (life-cycyle) olarak ifade edilebilir. Geliştirme projelerinin aşamalarının neler olduğu ve bu aşamaların nasıl bir sıralama ile uygulanacağı yaşam döngüsü modellerinde belirtilmektedir. Bu bölümde yaşam döngüsü modellerinden, yaygın olarak kullanılan 5 model ana hatlarıyla ele alınmaktadır.

III.1. Çağlayan Modeli

Çağlayan yaşam döngüsü modeli (kısaca "çağlayan modeli"), 1960'ların sonları, 1970'lerin ilk yıllarında geliştirilmiştir. Winston Royce tarafından 1970'de yayınlanan "Managing the Development of Large Software Systems" başlıklı makale, bu yaşam döngüsü modelini tüm dünyaya tanıtmıştır. Çağlayan modeli, yaşam döngüsü modellerinin en eski örneğidir ve yoğun eleştirilere hedef olmuştur. Söz konusu eleştiriler, yeni yaşam döngüsü modellerinin geliştirilmesinde önemli rol oynamıştır.



Şekil.2. Çağlayan Modeli

Kaynak: Royce, W. (1970). Managing the Development of Large Software Systems. Proceedings of IEEE WESCON, August, 3. (http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf). [03.03.2006]. [6].

Çağlayan modeli, yazılım geliştirmede yukarıdan aşağıya (top-down) bir yaklaşıma göre oluşturulmuştur ve doğrusal bir şekilde sıralanan aşamalar içermektedir. Çağlayan modelinde her aşamanın sonunda bir geliştirme dokümanı oluşturulmaktadır. Geliştirilen bu dokümanın tutarlılığı, önceki sürecin dokümanları temelinde gerçekleştirilen bir onaylama süreciyle güvence altına alınmaktadır. Çağlayan yaşam döngüsü, geri besleme çalışmalarını gerçekleştirerek kalite güvencesiyle ilgili boyutları sisteme entegre etmeyi hedeflemektedir. Bitirilen her aşama bir kilometre taşı olarak belirtilmektedir. Bu yaklaşım, her aşamanın sonunda projenin devamını saptayan yönetim kararını öngörmektedir. Roller prosedürel yapıya göre, aşamalar boyunca tanımlanmaktadır. Bu yüzden, takım üyeleri belirli fonksiyon-spesifik roller üstlenebilirler (Örnek: Sistem analisti, sistem tasarımcısı, programcı vb.) [5].

Çağlayan yaşam döngüsü modeli, farklı kaynaklarda farklı şemalarla ifade edilebilmektedir. Bazı kaynaklar, iki aşamayı tek aşama altında biraraya getirirken, bazıları modelin ilk aşaması olan sistem gerekliliklerinin belirlenmesine yer vermemektedir. Bununla birlikte, modeller farklı şemalarla ifade edilse de ana fikir değişmemekte, işlemler aynı olmasına karşın bu işlemlerin ne şekide yönetildiği modellerde farklılık göstermektedir.

Çağlayan modeli yaygın olarak kullanılsa da uygulamada oldukça önemli eleştirilere hedef olmuştur. Örneğin her aşama, tam anlamıyla kesin olarak tanımlanmış bir sisteme ihtiyaç duymaktadır ve bu sistemin yetkinliği uygulamada henüz kanıtlanmış değildir. Kapsamlı analiz, tanımlama ve tasarım çalışmaları oldukça uzun zaman almaktadır. Yazılım kullanıcılara teslim edildiğinde, kullanıcının ihtiyaçları değişmiş olabilir. Sistemin kullanılabilirliği, gereklilikler analizinin doğruluğuna bağlıdır. Yaklaşımın hiyerarşik yapısından dolayı, değişiklikler oldukça fazla çabayla gerçekleştirilebilmektedir. Hepsinden önemlisi, en kritik tasarım kararları, sadece sınırlı enformasyonların söz konusu olduğu, yazılım geliştirme sürecinin başında alınmaktadır. Ayrıca, bir aşamanın modellenmesinde kullanılan unsurlar, diğer aşamalarınkine uyum göstermeyebilir. Uygun olmayan model unsurları tüm süreçler boyunca ortaya çıkabilecek yanlışlara yol açabilir [5].

Yukarıda belirtilen dezavantajların dışında, yaşam döngüsündeki süreçlerin döngünün adil şekilde bölüşümünü yansıtmadığı, bir aşamada ortaya çıkan sorunların tam olarak çözümü öngörülmeden diğer aşamaya geçildiği, böylece sorunların diğer aşamaya taşındığı ve burada çözüm bekledikleri belirtilmektedir. Çağlayan modelinin izlenmesi halinde, risk faktörünün yazılımın başarısını etkileyen en önemli unsur olacağı ifade edilmektedir [7].

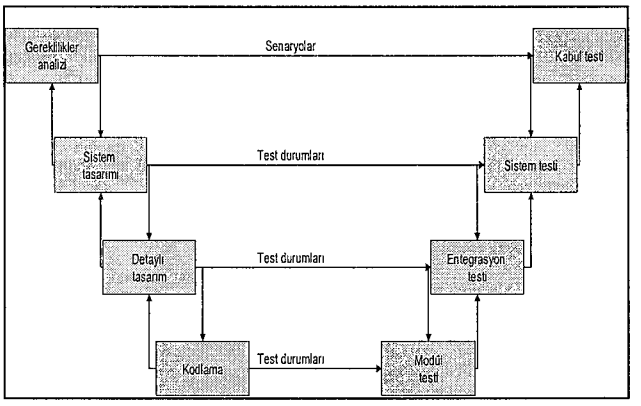
III.2. V-Modeli

Boehm tarafından geliştirilmiş olan bu modelde sıralı bir yaşam döngüsü modeli söz konusudur ve doğrulama / onaylama ölçüleri her sürecin sonunda süreç çıktılarına uygulanmaktadır. Modelin ismindeki V harfi, Almanca'da hareket tarzı anlamına gelen "Vorgehensweise" kelimesinin ilk harfinden gelmektedir.

V-Modeli Alman Hükümeti tarafından hem askeri alandaki hem de diğer alanlardaki yazılım geliştirmelerinde temel olarak kullanılmaktadır. Bunun dışında birçok firmada da standart yazılım geliştirme modeli olarak temel alınmaktadır. Model, yazılım geliştirmede ve ilgili çalışmalarda kalite güvencesi, konfigürasyon yönetimi ve teknik proje yönetimi için

gerçekleştirilecek faaliyetleri ve elde edilecek ürünleri (sonuçları) başlangıçta belirlemektedir. Bu modelin ulaşılmasına katkı sağladığı hedefler şunlardır:[8]

- Yazılım kalitesinin sağlanması ve iyileştirilmesi,
- Yaşam döngüsü üzerinde yazılım masraflarının azaltılması,
- Projenin tüm katılımcıları arasındaki iletişimin iyileştirilmesi, ayrıca projeyi veren kişilerin projeyi gerçekleştiren kişilere olan bağımlılığının azaltılması.



Şekil.3. V - Modeli

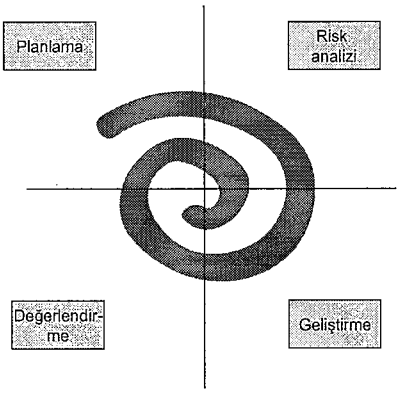
Kaynak: Knöll, H.D.; Kuhl, R.W.A.; Kuhl, L.W.H. & Moreton, R. (2001). Optimising Business Performance with Standard Software Systems. Braunschweig / Wiesbaden: Der Verlag Vieweg. 135 [5].

V-Modeli, organizasyon(lar)a bağlı saptamalarda bulunmamaktadır. Başka bir deyişle model organizasyon-bağımsızdır. Bu yüzden, farklı alanlardaki farklı süreç yapılarında kolayca uygulanabilmektedir. Söz konusu uygulamanın gerçekleştirilmesi amacıyla, modeldeki faaliyetler gruplandırılmış ve her grup için roller belirlenmiştir.

III.3. Helezonik Model

Helezonik model (spiral modeli), çağlayan modelinin dezavantajlarını gidermek üzere geliştirilmiştir. Boehm tarafından geliştirilen helezonik model 4 aşama içermektedir. Bunlar; planlama, risk analizi, geliştirme ve değerlendirme aşamalarıdır. Bu aşamalar, tüm sorunların giderilmesi amacıyla sırayla birbiri ardına tekrarlanmaktadır. Aşamaların tekrarlanması, bir aşamada ortaya çıkan sorunların daha iyi anlaşılmasına ve bu aşama tekrarlandığında sorunlarla daha iyi başa çıkılmasına yardımcı olmaktadır. Planlama ve geliştirme stratejileri, aşamaların tekrarlanması sırasında takip edilmektedir [9].

Bu modelin en önemli özelliği, her döngüde risk analizinin gerçekleştirilmesidir. Risk analizi temel alınarak, sonraki adımlar belirlenmektedir. Helezonik modelde, gereklilik tanımlaması odaklı, ön ürün odaklı, kıyaslama odaklı ve simülasyon odaklı yaklaşımlar mevcuttur. Her döngünün sonunda, tüm sonuçların değerlendirildiği bir gözden geçirme çalışması gerçekleştirilmektedir. Kullanıcılar sistem geliştirme çalışmalarına belirli aşamalarda katılmaktadır [7].



Şekil .4. Helezonik Model

Kaynak: Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. Computer, 21(5), 3. [7].

III.4. Nesne Yönelimli Model

Nesne yönelimli anlayışın ve nesne yönelimli programlama dillerinin ortaya çıkışı, 1970'lere kadar uzanmaktadır. Nesne yönelimli modelleme kapsamında geliştirilen ilk metodlar, sadece belirli alanlardaki uygulamaları desteklemiş, zamanla metodların kapsamları genişletilmiştir. UML (Birleştirilmiş Model Oluşturma Dili-Unified Modelling Language), 1997'de OMG (Object Management Group) tarafından sahiplenilmiş ve açık standart olarak geliştirilmeye başlanmıştır. OMG kar gütmeyen, bilgisayar endüstrisi standartlarının oluşturulmasında faaliyet gösteren bir organizasyondur. Dolayısıyla UML herhangi bir şirkete veya kişiye ait değildir, günümüzde Rational, MS Visio, Together Soft, vb. birçok modelleme aracı tarafından sunulmaktadır. UML nesne tabanlı sistemleri modellemede kullanılmaktadır, görsel bir modelleme dilidir ve modellemedeki başarısını sıkça kanıtlamıştır [10].

Nesne yönelimli metodlar, enformasyonu ve bu enformasyonun işlendiği süreci, enformasyonun betimlediği gerçek dünyadaki nesnelere göre organize etmektedir. Yeni olmasına karşın, nesne yönelimli metodlama bilişim sistemleri endüstrisinin ilerleyeceği yön olarak değerlendirilmektedir. Bu modelleme türü deneme safhasını geride bırakmıştır ve bilişim sistemlerinin geliştirilmesi için kullanılmaya başlanmıştır. Bu yönelim iki etken tarafından hızlandırılmıştır:

- Nesne yönelimli programlamanın başarısı ve hızlı şekilde yaygınlaşması,

• Nesne yönelimli metodların; daha etkin, verimli, esnek ve kalıcı bilişim sistemlerini oluşturması.

1960'lı yıllarda analiz ve problemin tanımlanması süreçleri için kullanılan herhangi bir yöntem yoktu. Yapılabilen, müşteriler ile gerçekleştirilen görüşmeler yoluyla problemin tanınmasına ve çözümün geliştirilmesine ilişkin yeterli bilgiye ulaşmaya çalışmaktı. 1960'lardan 1990'lara kadar birçok metod geliştirildi, ancak bu metodlar çeşitli sorunlar nedeniyle hala yeterince etkin değildi. Bu sorunlardan en önemlileri şunlardır:

1. Bakım

Yazılımların büyüklüğü ve yapılarının karmaşıklığı her geçen gün artmaktadır. 60'lı yıllarda bir yazılım en fazla 15.000 kod satırına sahipken günümüzde yazılımların içerdiği kod satırları on milyonlarla ifade edilmektedir. Ayrıca günümüzde kullanılan yazılımlar diğer yüzlerce yazılım türü ile etkileşimde bulunmalı, haberleşmeli ve ortak operasyonlar gerçekleştirmelidir.

2. Yapılması Gereken Çalışmalar

Geliştirilen yeni teknikler ve araçlar sayesinde süreçler daha iyi anlaşılsa da kullanıcıların ihtiyaçları büyük ölçüde artmıştır. Neyin, nasıl yapılması gerektiği daha iyi kavranmasına karşın, yapılması gereken çalışmaların kapsamı azalmış değildir.

3. Güvenilirlik

Tarih boyunca yazılımların tasarımı aşamasında yapılan hatalar çok sayıda probleme yol açmıştır. Bu problemlerin çözümü için yapılan çalışmalar ise bakım faaliyetlerini daha zor, daha zaman-yoğun ve daha pahalı kılmıştır.

Nesne yönelimli tekniklerin sağladığı faydalar ise şöyle özetlenebilir [11]:

1. Değişikliklere Uyum

Bir kere kurulmuş olan ve çalışan bir bilişim sistemi, yıllar sonra, kullanıcılarının ihtiyaçlarında yaşanan değişikliklere paralel olarak tekrar biçimlendirilebilmelidir. Sistemin esnekliği iki temel faktöre bağlıdır. Bunlardan birincisi, sistemdeki değişikliklerin en az zaman ve çaba harcanarak, önemli aksaklıklara yol açmadan yapılabilmesidir. İkincisi ise değişiklikler yapılırken, sistemdeki diğer unsurların zedelenmesi riskinin en az seviyede olmasıdır.

2. Bakım Kolaylığı

Önceki yöntemler, raporların şu anki ihtiyaçlarını karşılama üzerine temellendirilmişlerdir. Raporlarda değişiklikler gerektiğinde, yazılımların da sürekli değiştirilmesi gerekmekte, bu durum sistemin bakımı alanında önemli sorunlara yol açmaktadır. Nesne yönelimli metodlarla oluşturulan sistemlerin bakımı ve değiştirilmesi / genişletilmesi ise kolayca gerçekleştirilebilmektedir.

3. Tekrar Kullanılabilirlik

Nesne yönelimli tekniklerin en önemli faydalarından biri, program kodunun ve analiz sonuçlarının tekrar kullanılabilmesidir.

4. Gerçeğe Dayalı Sistemler

Nesne yönelimli metodlarda kullanılan teknikler, kullanıcıların iş hareketleri ve bu hareketlerin enformasyon ihtiyacı ile ilgili daha doğru değerlendirmeleri mümkün kılmaktadır.

5. Verilere Erişilebilirlik

Verileri dosyalamak sadece tek bir işlemdir, buna karşın aynı verilere tekrar erişmek oldukça farklı işlemleri gerektirebilir. Veri tabanının tasarımı, iyi bir veri tabanını kötü bir veri tabanından ayıran başlıca unsurdur. Bu tasarım, kullanıcı verilerinin ve bu veriler arasındaki ilişkinin tam olarak anlaşılmasına bağlıdır.

6. Kullanıcıların Katılımı

Nesne yönelimli teknikler yoluyla, kullanıcıların sistem geliştirme projesine doğrudan katılımı sağlanmaktadır. Sistem geliştiricilerin işlerini verimli olarak yapıp yapmamaları, oluşturulan sistemin kullanıcıların işlerini kolaylaştırmasına bağlıdır. Sonuç olarak kullanıcılar sistemi sahiplenmekte, faaliyetlerinde daha etkin şekilde kullanmaktadır.

III.5. Hızlı Uygulama Geliştirme Modeli

Hızlı uygulama geliştirme (HUG) yaklaşımı, iş dünyasında artan değişim baskısının, bilişim sistemlerinin oluşturulmasında dikkate alınması amacıyla geliştirilmiştir. Hızla değişen pazarlar, şirketleri hızla değişen gereklilikler ile karşı karşıya bırakmaktadır. Bu hızlı değişiklikler bilişim sistemleri tarafından karşılanmalı ve geliştirilen uygulamalar hızla kullanıma sunulmalıdır. HUG'nin başlıca amacı şöyle özetlenebilir: Yüksek kalitedeki sistemlerin en az masrafla, hızlı şekilde geliştirilmesi ve teslimi. HUG yaklaşımında vurgulanması gereken bir nokta, bu yaklaşımda kullanıcıların ve müşterilerin katılımının önemli rol oynadığıdır. Söz konusu katılımı gerçekleştirmede çeşitli zorluluklarla karşılaşılabilir, çünki "temel kullanıcılar"ın sistem geliştirme sürecine dahil edilmesi gerekmektedir. Bu ise

temel kullanıcılar ile diğer kullanıcıların ayırt edilmesini gündeme getirmektedir [5].

HUG yaklaşımında, sistemin kalite kriterleri tespit edilmekte ve sistem sınırları belirlenmektedir. Ayrıca belirlenen kalite kriterleri ve fonksiyonel özellikler için öncelikler belirlenmektedir (Örnek: Zorunlu özellikler, zorunlu olmayan ancak fayda sağlayacak olan özellikler, istenmeyen özellikler vb.). Herhangi bir zaman kısıtı söz konusu olduğunda, yukarıda belirtilen öncelikler temel alınmakta ve yüksek öncelikli olan özellik tercih edilmektedir. HUG yaklaşımı, hızlı ön ürün ve evrimsel sistem geliştirme yaklaşımlarını kendi süreçlerinde içermektedir. Ön ürün, belirli sistem özelliklerinin (Örnek: Kullanıcı ara yüzleri) analizi ve kolayca anlaşılması amacıyla tasarlanmaktadır [12].

IV. UYGULANACAK YAŞAM DÖNGÜSÜ MODELİNİN BELİRLENMESİ

Geliştirilecek olan bilişim sisteminde hangi yaşam döngüsü modelinin kullanılacağı, sistemin sahip olması gereken özellikler ve geliştirme sürecinde hangi önceliklerin söz konusu olduğuyla yakından ilgilidir. Oluşturulacak olan sistemin özellikleri bağlamında, sistemin kullanılacağı ortamın ve sistemin özelliklerinin kesin olarak tanımlanabilmesi büyük önem taşımaktadır. Eğer sistemin hangi koşullar altında kullanılacağı ve hangi özellikleri içermesi gerektiği biliniyorsa kullanıcı katılımına fazla ihtiyaç duyulmuyorsa ve sistemin bir an önce kullanıma sunulması beklenmiyorsa çağlayan modeli temel alınabilir. Yukarıdaki koşulların yanında, her aşamanın sonundaki çıktılara uygulanacak değerlendirme / onaylama kriterleri önceden belirlenmişse V-modeli sistem geliştirmede kullanılabilir. Sistemin geliştirilmesinde önemli yatırımların yapılması ve buna bağlı olarak risk faktörünün minimize edilmesi isteniyorsa ayrıca sistem geliştirme sürecinde önemli sorunlarla karşılaşılacağı ön görülüyorsa helezonik modelin kullanımı gündeme gelmektedir. Geliştirme sürecinde çok sayıda fonksiyonun ve bu fonksiyonların içerdiği çok miktarda enformasyonun modellenmesi gerekiyorsa söz konusu modellemeleri tek çatı altında biraraya getiren nesne yönelimli modelleme önemli katkılar sunabilir. Bilişim sistemlerinin geliştirilmesine kullanıcıların aktif katılımı isteniyorsa ve sistemin bir an önce kullanıma alınması hedefleniyorsa hızlı uygulama geliştirme yaklaşımı ön plana çıkmaktadır.

V. SONUÇ

Bilişim sistemlerinin geliştirilmesinde, yaşam döngüsü modelleri önemli katkılar sunmaktadır. Geliştirme çalışmalarının başarılı şekilde yürütülmesinde ve sonlandırılmasında, doğru modelin seçimi en kritik aşamalardan biridir. Bu seçimde, gerek geliştirilecek olan bilişim sisteminden beklentiler gerekse geliştirme süreciyle ilgili saptamalar belirleyici olmaktadır. Özetle her durum için doğru kabul edilebilecek bir yaşam döngüsü modeli bulunmamakta, sistem geliştirme çalışmalarında hangi modelin temel alınacağı ilgili koşullar bağlamında değişiklik göstermektedir.

YARARLANILAN KAYNAKLAR

[1] Erkut, H. (2000). Sistem Yönetimi. İstanbul: İrfan Yayımcılık.

[2] Barutçugil, İ. (2002). Bilgi Yönetimi. İstanbul: Kariyer Yayınlık.

[3] Thaller, G.E. (2003). Software-Projekt Management. Frankfurt: Software & Support Verlag.

[4] Balzert, H. (2001). Lehrbuch der Software Technik. Berlin: Spektrum Verlag.

[5] Knöll, H.D.; Kuhl, R.W.A.; Kuhl, L.W.H. & Moreton, R. (2001). Optimising Business Performance with Standard Software Systems. Braunschweig / Wiesbaden: Der Verlag Vieweg.

[6] Royce, W. (1970). Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON,* August, 1-9. (http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf). [03.03.2006].

[7] Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, 21(5), 61-72.

[8] Fraunhoferinstitut. (2005). Experimentelles Softwareengineering. (http://www.iese.fhg.de/VModell/Intro/vm.introMain.html). [03.03.06].

[9] Lehner, F.; Auer-Rizzi, W.; Bauer, R.; Breit, K.; Lehner, J.M. & Reber, G. (1991). *Organisationslehre für Wirtschaftsinformatiker.* Münich/Viyana: Carl Hanser Verlag.

[10] Tokgöz, G.M. (2004). UML ile Yazılım Modellenmesi. *EMO Bilgisayar Mühendisliği Dergisi,* 2003-2004. (http://bm-dergi.emo.org.tr/index.php?option=com_content&task=view&id=18&Itemid=74) [04.03.06].

[11] Brown, D.W.M. (1997). *Introduction to Object-oriented Analysis.* John Wiley and Sons: New York.

[12] Fischer, J.; Herold, W. & Dangelmaier, W. (2002). *Bausteine der Wirtschaftsinformatik.* Hamburg: Erich Schmidt Verlag.

Yücel YILMAZ (yucelyilmaz@marmara.edu.tr) has a PhD in Informatics from Marmara University. Presently he is a lecturer in the Department of Business Informatics at Marmara University. His research areas are information systems and knowledge management.