KAD İle Finans Sektöründe Yazılım Hatalarını En Aza İndirgeme Çözüm Süreci

[1]Ayşe Betül Karagöz, [2]Fatma Molu
[1,2]Kuveyt Türk Katılım Bankası Ar-ge Merkezi, Kocaeli, Türkiye

Abstract

The most effective way to finance and sustain economic growth is possible with a vast, powerful and sound structure in financial sector. It is crucial for individuals and companies to have availability for credit and sustenance in financing investment, production, consumption and commerce. Operating in order to meet the financial services required by customers and the economy and to ensure that the payments system work, financial institutions are crucial actors in economic activity.[1]
It has been clearly understood from certain examples of software products employed in such a critical field of profession that errors arising from these products caused material and immaterial damages, lead to loss of customer trust and resulted in damage of corporate reputation. Studies have shown that distribution channels like internet banking started to be frequently used especially by intellectuals with the spread of internet and the more widespread the service became, the more criticism it received, inevitably leading to competition.[2] These circumstances lead financial sector to search new methods to minimize software errors.
Critical and analytical thinking methods such as critical approach, hypotheses testing and prejudice detection were employed starting from problem detection phase to final solution phase in order to understand the problem. Experience of this phase and period of change were explained and critical and analytical thinking and properties of them were emphasized.[3]

Key words: CAT, finance, software, software testing

Özet

Ekonomik büyümeyi finanse etmenin ve sürdürmenin en etkin yolu finansal sektörün büyük, güçlü ve sağlıklı bir yapıya sahip olmasıdır. Yatırım, üretim, tüketim ve ticaretin finansmanında, şirketler ve bireyler için kredinin bulunabilirliği ve sürekliliğinin sağlanması hayati öneme sahiptir. Temel işlevleri müşterilerinin ve ekonominin ihtiyaç duyduğu finansal hizmetleri sunmak ve ödeme sisteminin çalışmasını sağlamak olan finansal kurumlar ekonomik faaliyetin çok önemli bir parçasıdır.[1]
Böylesine kritik bir alanda kullanılan yazılım ürünlerinin kalitesinin ne kadar önemli olduğu yaşanılan örneklerde görülmüştür ki, yazılım ürünlerindeki hatalar maddi, manevi zararların olmasına, müşteri güven ve sadakatinin yitirilmesine, kurum itibarının zedelenmesine sebep olmuştur. Yapılan araştırmalar göstermiştir ki internetin yaygınlaşmasıyla birlikte İnternet Bankacılığı gibi alternatif dağıtım kanalları özellikle entelektüel çevrelerce sıkça kullanılmaya başlamış ve verilen hizmet sektöründe yaygınlaşmasıyla birlikte eleştiriler almayabaşlamış ve rekabeti kaçınılmaz noktaya taşımıştır.[2] Bu durum finans sektörünü yazılım ürünlerindeki hataları en aza indirgeme konusunda çözüm metotları arama yoluna itmiştir.
Makalede bu arayış sürecinin en başından yani sorunun tespit aşamasından başlayarak son aşaması olan nihai çözüm aşamasına kadar sorunu anlamada kritik analitik yaklaşım, varsayımların sorgulanması, önyargıların

tespit edilmesi gibi KAD yöntemleri ile ele alınması, tecrübe edilmesi ve değişim süreci anlatılmış, sürecin başarısına katkı sağlayan KAD avantaj ve KAD özelliklerine vurgu yapılmıştır.[3]

Anahtar Kelimeler:KAD, finans, yazılım, yazılım test

1. Giriş

Tarih boyunca görülmüştür ki yaşayan süreçler çoğu zaman tecrübelerle birlikte kendini kaliteli ve kullanılabilir hale getirirler. Tecrübelerden edinimlerin hayata geçirilmesi zaman kaybı, fazzla efor harcanmasına sebep olmaktadır ve herşeyin tecrübe edilememe ihtimali veya tecrübe edilmesinin beklenmesi süreçteki risklerin devam etmesine sebep olacaktır. Tüm bu durumlar yazılım süreçleri için de geçerlidir. Yazılım süreçlerindeki hatalar bir çok riske, can mal kaybına, itibar kayıplarına sebep olmuştur. Yazılım süreçlerinde yapılan hatalarla ilgili birçok tarihi olay kayıtlara geçmiştir. Tüm bunların sonucu olarak yazılım süreçlerinde analiz ve test süreçleri yazılım sürecine dahil dilmiştir. Makalede ağırlıklı olarak yazılım süreci modellerinden ve kritik analitik düşünceyi içinde barındıran yazılım test sürecinin de dahil edildiği yazılım süreci modelinden bahsedilmiştir.

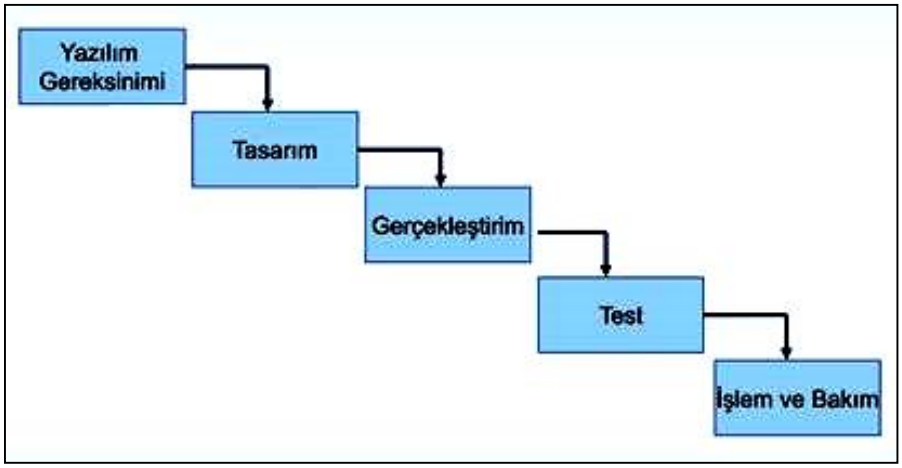
2. Yazılım Geliştirme Sürecinin Oluşumu ve Gelişimi

1837 yılında ilk olarak duyurulan "Analytical Engine Code Order" ve peşinden geliştirilen diğer makine dilleriyle başlayan dünya programlama süreci gelişimi içinde bulunduğumuz son yıllarda nesne yönelimli programlamanın gelişmesiyle birlikte çok hızlı ve pratik hale gelmiştir. Bu gelişmelerle birlikte giderek serileşen yazılım üretiminde metodoloji kullanımı ihtiyacı doğmuş ve 1960'lı yıllarda ilk defa Gelişigüzel Geliştirme Modeli uygulanmıştır. Gelişigüzel Geliştirme Modeli basit programlamanın olduğu ve üretimde sadece yazılımcının olduğu bir geliştirme modeliydi.Yazılımcı bir yazılım ürününü üretmek için sadece yazılım ürününü talep eden müşteri ile görüşür ve ürünü ortaya çıkarırdı. Bu süreçte yazılımcı sadece müşterinin ne istediği bilgisini alır ve müşteri ve yazılımcı müşterinin istedikleri üzerinde krtikanalitik yapmadıysa çoğu zaman ortaya müşterinin istediği veya istemediği, risklere açık, kullanılabilir olmayan bir ürün üretilirdi.Programlama dillerinin hızla gelişmesine ve serileşmesine rağmen süreçteki bu aksaklık devam etmiştir. Tüm bu yaşanan tecrübelerin sonunda programlama süreci fazlara ayırma ve ayrılan fazlarda da yazılımcı haricinde analist, testuzmanı ihtiyacı farkedilerek, yeni görev ve iştanımları oluşturularak ve farklı yazılım modelleri oluşturulmuştur. Bu geliştirme modellerinden biri olan Şelale Modeli 1970'li yıllardan bu yana dünyada en yaygın olarak kullanılan bir yazılım geliştirme modelidir.

3. Şelale Yazılım Geliştirme Modeline Kritik Bakış

Dünyada en yaygın kullanılan bu geliştirme modeli sürei analiz, tasarım, kodlama, sınama ve bakım gibi adımlardan oluşup, bir adımın tamamlanmasıyla bir sonraki adıma geçilebilmesi mümkün olabilmektedir. Herhangi bir adımda oluşan hataya da farkedilen eksiklik için ancak bir adım geriye gidilirse düzeltilebimektedir. [4]

Şelale Modeli 1970'li yıllarda Winston W. Royce tarafından yazılan bir makalede oluşturulmuştur. Şelale modelinin bu adı almasının sebebi ise birbiri içine geçmemiş ardışık aşamalardan oluşmasıdır, süreç yukarıdan aşağıya doğru akan şelale gibi peşpeşe devam eder. Her adım bir önceki adımda üretilenleri kullanarak kendi safhasını gerçekleştirir.



Şekil 1 – Şelale (Waterfall) Model Yazılım Süreci

3.1. Şelale *Modelinin* Avantajları

Gereksinimlerin en başta net olarak ortaya konulması projenin kapsamının, süresinin belli olmasını sağladığından netleşmiş bir iş planı ile proje başlar.
Projenin safhaları ayrı olduğundan iş bölümü ve iş planı projenin en başında net bir şekilde bellidir. Bu durum projenin yönetimini de oldukça kolay hale getirir.
İş akışı bir önceki adıma dönüş olması yapılan dokümantasyon ve raporlama süreçleri kolay bir şekilde eksiksiz tamamlanır.[5],[6]

3.2. Şelale Modelinin Dezavantajları

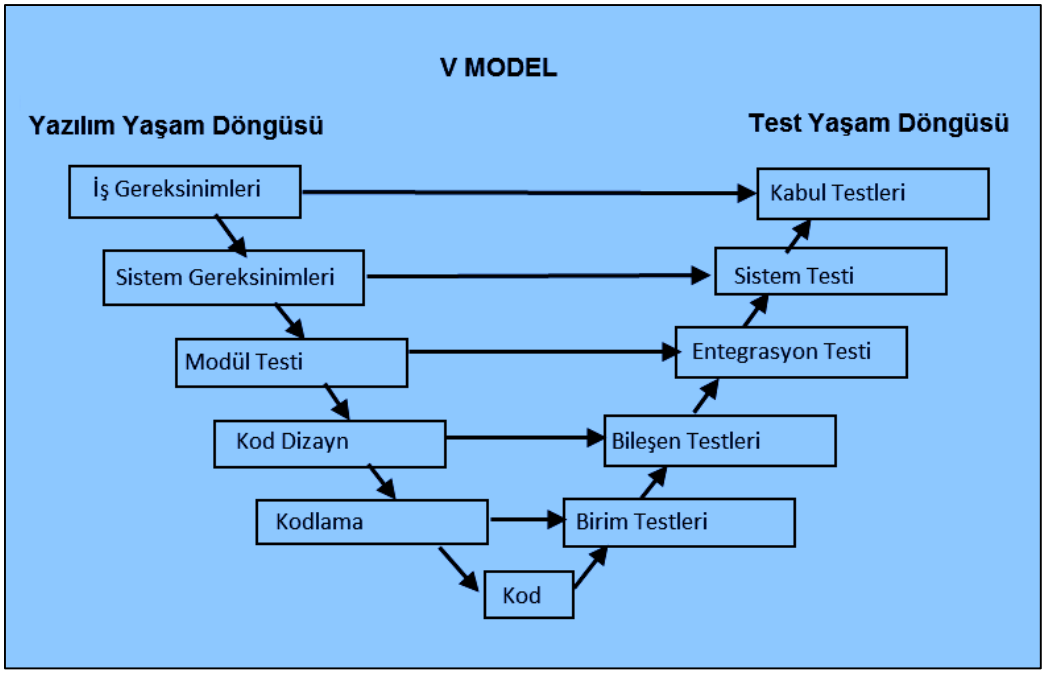
Projede olabilecek her türlü değişime karşı elverişsiz, katı bir modeldir. Yapılan her değişiklik maliyeti büyük oranda artırır.
Müşteri memnuniyetini sağlamak çok zordur çünkü gelişim ve değişime açık bir model değildir.
Model safhalardan oluştuğu için ürün son safhada tamamlanır, gereksinimlerin iyi tanımlanmadığı müşterinin ne istediğinin anlaşılmadığı bir projede bu durum projenin bittikten sonra iptal edilmesine ve başka gerginliklere sebep olmaktadır.

3.3 Şelale Modelinde Yazılım Testinin Dezavantajları

Test safhasının yazılım yaşam döngüsünün başında yer almayıp son aşamada olması hataların çözüm maliyetini ciddi oranda artırmaktadır.
Kullanıcı kabul testine son safhada çıkarılan uygulamanın kullanıcı ihtiyaçlarına cevap vermeme riski her zaman vardır. Kullanıcı çalışan uygulamayı sadece son safhada görür.
Test sürecinde tespit edilen, değişiklik gerektiren durumlar projeye yaşam döngüsüne kolay adapte edilemez.
Test sürecinde tespit edilen gereksinim kaynaklı hataların çözüm süreci proje kaynak ve bütçesi için sorun teşkil ettiğinden uygulamada değişiklik yapılamayabilir.

4. Her Adımında KAD ile Test Olan Bir Yazılım Süreci Modeli

Yazılım yaşam döngüsü boyunca her aşamada testin yer aldığı yazılım geliştirme modelidir.
Test aktiviteleri (planlama, dizayn gibi) kodlamadan önce başlar.
Bu yüzden hatalar iş gereksinimleri belli olduğunda bulunmaya başlar.
Bu zaman kazanılmasını sağlar ve daha ileri safhalarda çıkması durumunda maliyeti yüksek hataların önlenmesini sağlar.



Şekil 2 – V Model Yazılım Süreci

5. Yazılım Test Mühendisi

Yazılım Test Mühendisi özelliklerine baktığımızda görülmüştür ki yazılım test mühendisinde olması gereken özellikler KAD kültürüne sahip bir kişide olması gereken özelliklerdir.
Yazılım test mühendisi şu özelliklere sahiptir,

- Dikkatlidir, detayları önemser.
- Analizi bir varsayım kabul edip kritik eder ve sorgular böylelikle analiz aşamasında bir çok hatanın önüne geçer.
- Şüphecidir, bilgi kaynaklarını, alternatifleri araştırır ve dener.
- Tarafsızdır ve önyargılarını projeden uzak tutar.
- Planlı ve nettir.
- Süreci iyi bilir, testin konusuna iş ve teknik açıdan hakimdir.
- Test araçlarını iyi kullanır.
- İletişimi iyidir ve iletişim araçlarını en etkin şekilde kullanır.

Yazılım test mühendisi başından sonuna kadar yer aldığı tüm projelerde proje ekibinin tarafsız şekilde kritik ve analitik bakmalarını sağlar, bulundukları kurumdaki kalite seviyesine en üst seviyeye çıkarır. Test ekibi kalite organizasyonunun bir parçasıdır. Kalite yönetimi ve gelişimi için süreçlerin KAD ile analiz edilmesi, geliştirilmesi ve test edilmesi hataların en aza indirgenmesini sağlar. KAD bakış açısına sahip bir ekip kalitenin bir kültür haline getirilmesini sağlamış demektir.

5. Sonuç

Yılların deneyimi sonucu finans sektörü gibi önemli bir çok sektörde firmalar organizasyonlarında

bir kalite ve test ekipleri bulundurmaya başlamışlar ve böylelikle rekabetin had safhaya ulaştığı günümüzde kaliteye verdikleri önem derecesinde boy göstermişlerdir.
Sistematik hale getirilmiş her yazılım test süreci projenin başında yer almışsa o proje KAD ile yürtülmüş demektir.
Her safhasında KAD yaklaşımı içeren bir süreç elbette tecrübe edinmeden bir çok köklü değişimin yapılmasını ve tecrübe ile olgunlaşan bir sürece kıyasla çok daha kısa sürede olgunlaşmasını sağlamıştır. KAD gerekli öngörüyü sağlayarak tüm sistemleri ve süreçleri KAD yapılmayan süreç ve sistemlere kıyasla gözle görülür bir farkla geliştirmiş ve ileriye götürmüştür.
Kuveyt Türk Arge Yazılım projelerine destek veren on kişilik bir test ekibi bulunmaktadır. Bu ekip yazılım projelerinin %30'una dahil olmaktadır. Yazılım test ekibinin dahil olmadığı projelerde test ekibinin görevini analistler üstlenmiştir ancak birçok projede test ekibinin verimine ulaşılamadığı görülmüştür. Sonuç olarak kritik analik düşünceyi sistematik hale getirmiş test ekibi büyütülmeye devam etmektedir.

Referanslar

[1] TürkiyeBankalarBirliğiYön. Kur.Başk. Hüseyin Aydın, BankacılarDergisi, Sayı 84,2013
[2]SalihBarışık, HalimeTemel, http://dergi.kmu.edu.tr/userfiles/file/aralik2007/PDF/7.pdf
[3] http://www.kritik-analitik.com/ArticlesPopUp.aspx?Id=10
[4] Pressman, R. S., "Software Engineering: A Practitioner's Approach" cgraw/Hill2005
6th Ed. (2005)
[5]Beck K., "Embrace change with extreme programming", IEEE Computer, Ekim 1999 s.70–77
[6] Royce W.W., "Managing the development of large software systems: Concepts and techniques" Proc. WESCON, 1970, s. 1–9