# YAZILIM KALİTE GÜVENCESİNDE İSTATİSTİKSEL SÜREÇ KONTROLÜ

# STATISTICAL PROCESS CONTROL IN SOFTWARE QUALITY ASSURANCE

**ÖZDEN GÜR ÇİVLİK**

Hacettepe Üniversitesi
Fen Bilimleri Enstitüsü Yönetmeliğinin
İSTATİSTİK Ana Bilim Dalı İçin Öngördüğü
YÜKSEK LİSANS TEZİ
olarak hazırlanmıştır.

2006

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından İSTATİSTİK ANABİLİM DALI 'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

| | |
|---|---|
| Başkan | :…..<br>(Doç.Dr. Turhan Menteş) |
| Üye | :…..<br>(Yrd. Doç. Dr. Güvenç Arslan) |
| Üye (Danışman) | :…..<br>(Yrd. Doç. Dr. Canan Hamurkaroğlu) |

ONAY

Bu tez ...../...../..... tarihinde Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından tarihinde kabul edilmiştir.

...../...../.....

Prof.Dr. Ahmet R. ÖZDURAL
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

kızım ***NEHİR*****e,**

**YAZILIM KALİTE GÜVENCESİNDE İSTATİSTİKSEL SÜREÇ KONTROLÜ**

**Özden Gür Çivlik**

## ÖZ

Yazılım süreçlerinin yeterliliği ve etkinliği üzerine artan ilgi, klasik tekniklerin ötesinde, ölçümün önemi üzerine odaklanmaya başlamıştır. İstatistik ve süreç tabanlı düşünceler yazılım geliştirmenin tutarlılığı ve süreçlerin yeteneğini ölçmek için istatistiksel süreç kontrol tekniklerinin kullanımını ön plana çıkarmaktadır.

Bu çalışmanın amacı, belli bir olgunluk seviyesine ulaşmış yazılım firmalarına, süreç iyileştirmenin önemini ve süreçlerini iyileştirmeleri için kullanılabilecekleri İSK tekniklerini tanıtmak ve bir uygulama ile yol göstermektir.

Bu çalışmada, yazılım kalitesi üzerine ölçümün öneminden yola çıkarak mevcut İSK teknikleri kısaca tanımlanmıştır. Bu tekniklerden, kontrol kartları, pareto diyagramı, histogram ve gruplandırma kullanılarak, bilişim sektöründe faaliyet gösteren bir firmanın, süreçlerini iyileştirmesi için matematiksel sonuçlar ortaya çıkartılmıştır. Bunun için firmanın son birkaç yıl içerisinde gerçekleştirdiği 30 projeye ait hata sayıları, bu hataların tiplere göre dağılımı, bu hataların giderilmesindeki öncelikleri ve testten dönen hata sayıları incelenmiştir. Sonuç olarak, daha etkin süreç iyileştirme çalışması gerçekleştirmeleri için yazılım firmalarının dikkat etmeleri gereken ölçüm metrikleri belirtilmiştir.

Firmaların, süreçlerini iyileştirme çalışmalarında kılavuz görevi üstlenen yazılım yetenek olgunluk modellerinden, Yetenek Olgunluk Modeli (Capability Maturity Model - CMM), Yetenek Olgunluk Modeli Entegrasyonu (Capability Maturity Model Integration - CMMI) ve Yazılım Süreç İyileştirme ve Yetenek Belirleme (Software Process Improvement and Capability dEtermination, SPICE - ISO/IEC 15504) hakkında kısa bilgiler verilip bu modeller içerisinde İSK' nın kullanımı açıklanmıştır.

**Anahtar Kelimeler:** Yazılımda kalite güvence, yazılımın iyileştirilmesi, istatistiksel süreç kontrol teknikleri, yetenek olgunluk modelleri, kontrol kartları.

Danışman: Yrd. Doç. Dr. Canan HAMURKAROĞLU, Hacettepe Üniversitesi, Fen Fakültesi, İstatistik Bölümü Anabilim Dalı

**STATISTICAL PROCESS CONTROL IN SOFTWARE QUALITY ASSURANCE**

**Özden Gür Çivlik**

**ABSTRACT**

The interest rising on the adequacy and effectiveness of software processes starts to focus on the importance of measurement beyond classical techniques. Views based on statistics and processes bring forward the use of statistical process control techniques for measuring the consistency of software development and processes capability of processes.

The purpose of this study is to present to software firms having reached a certain maturity level, the importance of process improvement and the İSK techniques they can use in improving their processes and guide them with application.

In this study, the existing Statistical Process Control (İSK) techniques were shortly defined, starting with the importance of measuring software quality. Of these techniques, mathematical results for the improvement of processes of a firm operating in the information technologies sector were presented using control cards, pareto diagram and grouping. For this, the number of errors in the 30 projects the firm has realized in the last few years, the distribution of these errors according to types, the priorities of eliminating these errors and the number of errors returning from the tests were examined. In conclusion, the measurement metrics that software firms should pay attention to for the realization of more effective process improvement studies were mentioned.

Of the software capability maturity models that take on the task of a guide to the firms for improving their processes, short information was given about Capability Maturity Model - CMM, Capability Maturity Model Integration - CMMI and Software Process Improvement and Capability Determination, SPICE - ISO/IEC 15504 and the use of İSK among these models was explained.

**Keywords:** Software quality assurance, software improvement, statistical process control techniques, capability maturity models, control charts.

Consultant: Asst. Prof. Canan HAMURKAROĞLU, Hacettepe University, Faculty of Science, Statistics Department Science Branch

**TEŞEKKÜR**

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren danışmanım Yrd. Doç. Canan HAMURKAROĞLU' na, tez çalışmalarım için kayıt yaptırdığım andan itibaren manevi desteğini esirgemeyen İstatistik Bölüm Başkanı Prof. Dr. Süleyman GÜNAY'a, uygulamamda veri elde etmem için uygun firmanın bulunmasında beni yönlendiren ve yardımcı olan Doç. Dr. Turhan MENTEŞ'e, uygulamamı gerçekleştirmek için veri aldığım ve bana bu konuda yardımcı olan firma çalışanlarına, manevi desteklerini esirgemeyen İstatistik Bölümündeki tüm öğretim görevlilerine, tezimle ilgili danışmanımla yapacağım çalışmalarımda çıktı almamda yardımcı olan Kemal BİRİNCİ'ye, çalışmam için her aşamada desteğini esirgemeyen eşim Şevket Serkan ÇİVLİK'e, çalışmama zaman ayırabilmem için desteğini esirgemeyen annem Necla GÜR'e, tez çalışmamda bana yol gösteren babam Prof. Dr. Ali Rıza GÜR'e, ilgili kaynakları elde etmede yardımcı olan kardeşim Özge GÜR'e ve sevimliliğiyle çalışmamda en sıkıntılı anlarımı bile tatlı bir gülümsemeyle bakmamı sağlayan kızım Nehir'e teşekkürlerimi sunarım.

**İÇİNDEKİLER**

**Sayfa**

## ŞEKİLLER

**Sayfa**

## ÇİZELGELER

**Sayfa**

**SİMGELER VE KISALTMALAR**

| | |
|---|---|
| σ | Kitle standart sapması (Sigma) |
| R | Dağılım genişliği |
| $\overline{X}$ | Ortalama |
| S | Standart sapma |
| p | Hatalı oranı |
| np | Hatalı sayısı |
| c | Hata sayısı |
| u | Birim başına hata sayısı |
| JIS | Japon Standartları Enstitüsü |
| ISO | Uluslararası Standartlar Örgütü |
| İSK | İstatistiksel Süreç Kontrolü |
| OT-VT | Otomatik Tanıma / Veri Toplama |
| ANSI | Amerikan Ulusal Standart |
| ASQC | Kalite Kontrol için Amerikan Topluluğu |
| CMM | Yetenek Olgunluk Modeli |
| CMMI | Yetenek Olgunluk Modeli Entegrasyonu |
| SPICE | Yazılım Süreç İyileştirme ve Yetenek Belirleme |
| SEI | Yazılım Mühendisliği Enstitüsü |
| DoD | Amerikan Savunma Bakanlığı |
| IEC | Uluslararası Elektroteknik Komisyonu |
| ÜKL | Üst Kontrol Limit |
| AKL | Alt Kontrol Limit |

# 1. GİRİŞ

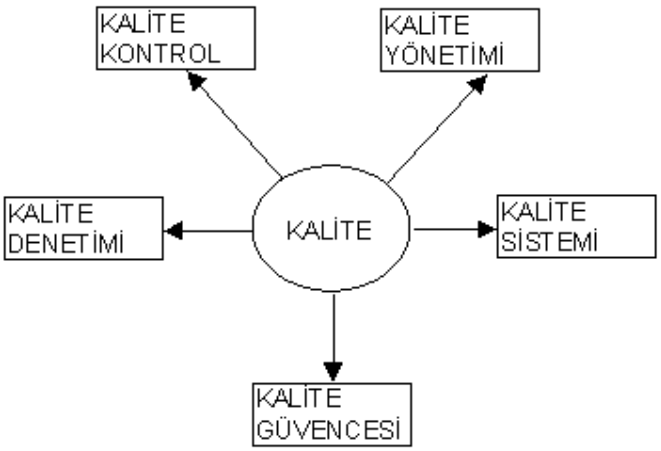
Kalite kavramı, ortaya çıkışından bu yana içinde bulunduğu koşullardan da etkilenerek değişik tanımlara sahip olmuştur.

Toplam Kalite Yaklaşımı ya da Japonların deyimiyle firma boyutunda kalite anlayışı, kalitenin, ürünün üretilmesi sırasında oluşturulmasını amaç edinmektedir. Bu fikir altında aşağıda değişik kalite tanımları yer almaktadır (Gençyılmaz ve Zaim, 1999):

- Kalite, kullanıma uygunluktur (Dr. Joseph M. Juran).
- Kalite, şartlara uygunluktur (Philip B. Crosby).
- Kalite, bir ürün ya da hizmetin belirlenen ya da doğabilecek gereksinimleri karşılama yeteneğine dayanan özelliklerin toplamıdır (ISO–8402).
- Kalite, mükemmeli arayışın sistematik bir yaklaşımıdır (Kalite Kontrol için Amerikan Topluluğu - ASQC).
- Kalite, ürün ya da hizmeti ekonomik bir yoldan üreten ve tüketici isteklerine cevap veren bir üretim sistemidir (Japon Standartlar Enstitüsü - JIS).
- Kalite kontrolü uygulamak, en ekonomik, en kullanışlı ve tüketiciyi daima tatmin eden kaliteli ürünü geliştirmek, tasarımını yapmak, üretmek ve satış sonrası hizmetlerini vermektir (Prof. Dr. Kaoru Ishikawa).
- Bir ürün ya da hizmetin belirlenen ya da olabilecek ihtiyaçları karşılama kabiliyetine dayanan özelliklerinin toplamıdır (TS EN ISO 8402:1998).

Kalite dünyası içerisinde yer alan kalite kontrol, kalite güvence, kalite yönetimi, kalite sistemi ve kalite denetimi kavramları birbirleriyle karıştırılmaktadır. Şekil 1.1. “kalite” kavramından türeyen diğer kalite kavramlarını göstermektedir. Bu kavramlar kısaca; kalite kontrol “kalite isteklerini sağlamak için kullanılan uygulama teknikleri ve faaliyetleri”, kalite güvencesi “ürün ya da hizmetin kalite için belirlenen istekleri karşılamak maksadıyla yeterli güveni sağlaması için gereken planlı ve sistematik faaliyetlerin bütünü”, kalite yönetimi “genel yönetim fonksiyonunun kalite politikasını belirleyen ve uygulayan bölümünü”, kalite sistemi "kalite yönetiminin

uygulanması için gerekli olan kuruluş yapısı, sorumluluklar, prosedürler, işlemler ve kaynakları", kalite denetimi "kalite ile ilgili faaliyetlerin ve sonuçlarının, planlanan düzenlemelere uyup uymadığının, bu düzenlemelerin etkili olarak uygulanıp uygulanmadığının ve amaca ulaşmak için uygun olup olmadığının sistematik ve tarafsız olarak incelenmesi" dir.




Şekil 1.1. Kalite Kavramları

Japon işletmelerinin kalite düşüncesiyle hareket etmesi ve yüksek satış rekorları kırmasının ardında toplam kalite anlayışının büyük katkısı olmuştur. 1980'lerden günümüze kadar tüm dünyada ve ülkemizde kalite kavramı hızla gelişmektedir. Kalitenin gelişiminde Shewhart, Deming, Juran ve Crosby gibi kalite guruları önemli roller oynamışlardır (Aksoy, 2005).

## 2. YAZILIM VE KALİTE

Uluslararası Standartlar Örgütü (International Organization for Standardization - ISO) tanımlarına göre yazılım, bir veri işleme sistemi operasyonuna bağlı olan programlar, yordamlar ve bunların dokümantasyonudur. Yazılım ürünü ise kullanıcıya ulaştırılmak üzere tanımlanmış bilgisayar programları kümesi, yordamları ve dokümanlarıdır.

Kalite kavramı, üretim sektöründe tasarımdan başlayarak son kontrole kadar önemli bir rol almıştır. İçerisinde ürün olma özellikleri, süreç olma özellikleri ve geliştirme aşamasında teknoloji yönetimi, proje yönetimi, süreç yönetimi, kalite yönetimi, insan kaynakları yönetimi vb. gibi farklı disiplinleri kullanan yazılım sektöründe de kalite yer almaktadır (Özkan, 2001).

Bilgisayar tabanlı sistemlerin istenildiği gibi çalışabilmesi, insanlara yardım edilebilmesi için tüm öğelerin uygun kalitede (nitelikte) olması gerekir.

Yazılım firmalarının misyonu, müşterilerine, rakiplerinin önünde yer alarak yenilikçi ürün ya da hizmeti rekabetçi bir fiyatta sağlamaktır. Bunu gerçekleştirmek için açık bir iş vizyonuna, yenilikçi bir kültüre ve mükemmel bir ürün geliştirme yöntemine ihtiyaç duyar.

Hatta istenilen kalitenin yazılım ürününe uygulandığından ve müşterilerin firmadan memnun olduğundan emin olmak için müşteri memnuniyeti ve yazılım kalitesi üzerine odaklı bir sisteme ihtiyaçları vardır. Bu yeni yüzyılla birlikte müşteriler, zamanında teslim edilen yüksek kalite seviyesinde bir yazılım ürünü beklemektedirler. Kalite üzerine bu odaklanma, üretilecek olan yazılımın kaliteli olmasını sağlamak için mükemmel bir yazılım geliştirme altyapısına ihtiyaç duymaktadır (O'Regan, 2002).

Yazılım kalitesinin iki farklı boyutu vardır. Birincisi, son üründe kullanıcı anlayışının kalitesini oluşturan karakter ve alt karakterler gibi dış (external) ürün görünüşü ki bu genellikle kullanımda-kalite olarak adlandırılır. Kullanımda-kalite, sadece yazılım ürünü tamamlandığında yazılımın dış özellikleri ölçülerek hesaplanır.

İkincisi, üretilmekte olan ürünün kalitesini kontrol altında tutabilen kriterleri içiren yani iç ürün görünüşüdür (internal product view). İkinci boyut ürün kalitesini oluşturur (Fenton et al., 2002).

Yazılım geliştirmede kalitenin hedefi “sıfır hata” yaklaşımıdır. Sıfır hata ile bir ürün üretilmesi için, sürekli geliştirilen süreçler ve sürekli eğitilen nitelikli çalışan kişiler ön koşullardır.

Geliştirme süreci aşağıda yazan maddelerle birlikte ürünün teslim edilmesidir: (Burr and Owen, 1996)

- zamanında
- bütçesinde
- özelliklerini karşılayarak

Yazılım geliştirmenin temel amacı yüksek nitelikli, diğer deyişle yüksek kaliteli yazılım üretmektir. Yazılım kalitesi, kullanım amaçlarına göre açıkça tanımlanmış işlev ve başarım gereksinimlerine uyum, kullanıcı isteklerine yanıt verebilme, açıkça belgelendirilmiş yazılım geliştirme standartlarına uygunluk, yüksek güvenilirlik sağlama, üretilen yazılımda çeşitli teknik özelliklere sahip olma ve teslim sonrası destek olarak tanımlanır.

Dolayısıyla, gereksinimler ya da kullanıcı istekleri kalitenin ölçülmesine temel oluştururlar. Onlardan uzaklaşmak aslında işlevsel nitelikten uzaklaşmak demektir. Belirlenmiş standartlara uyularak geliştirilen yazılım belirli bir teknik nitelik taşır; ancak, işlevsel olarak işini göremeyen yazılım kaliteli olarak kabul edilemez. Geliştirme standartlarının dışına çıkılması durumunda ise teknik kaliteden ödün verilmiş olur. Yazılımın sahip olması gereken doğruluk, sağlamlık, modülerlik, anlaşılabilirlik, bakım kolaylığı gibi özelliklerin eksikliği işlevsel olarak çok iyi olan bir yazılımın kalitesinin de eksikliği anlamına gelir.

### 2.1. Yazılım Kalitesinde Kalite Kriterleri

Yazılımın taşıması gereken önemli özellikler iyi bir yazılımda aranan ve ürünün kalitesini belirleyen etmenler, yani faktörlerdir. Yazılımda kalite kriterlerini farklı ürünleri birbirleriyle kıyaslayabilmek amacıyla çeşitli gruplara ayırmak mümkündür. Ürün özelliklerini dikkate alarak şu üç grupta inceleme yapılabilir:

- Ürünün kullanımına yönelik özellikler

- Ürünün taşınmasına yönelik özellikler
- Ürünün yenileştirilmesine yönelik özellikler

### 2.1.1. Kullanıma yönelik özellikler

Bir yazılım ürününün yalnızca kullanımının dikkate alınması, kullanıcı memnuniyetini sağlayacak özellikleri listelemeyi gerektirir. Bunun için işlevsellik, doğruluk, sağlamlık, güvenilirlik, verimli çalışma, korunmalı olma, kullanım kolaylığı, ekonomiklik ve işletim sürekliliği özellikleri verilebilir.

- İşlevsellik

Yazılım, kullanıcının tüm isteklerini karşılayabilmelidir. Ana işlevler yanında yardımcı işlevlerle zenginleştirilmiş olmalıdır.

- Doğruluk

Yazılım, öngörülen tüm işlevleri istenildiği şekilde, doğru ve yeterli hassaslıkla yerine getirebilmesidir.

- Sağlamlık

Yazılım, normal olmayan çalışma koşullarında da güvenle çalışabilmeli, sisteme hatalara karşı hoşgörülü bir yapı kazandırılmalıdır. Tasarım ve geliştirme aşamasında belirlenemeyen bazı koşullar sonradan ortaya çıksa da sistem işlevlerini yerine getirebilmelidir.

- Güvenilirlik

Yazılım, olgun bir sürüme sahip olmalı, sürekli ve hatasız çalışabilmeli, tüm işlevleri doğru yapmalı, hatalı girdilere ve kullanıcı yanlışlıklarına karşı korumalı olmalıdır.

- Verimli çalışma

Yazılım, işlevlerini yerine getirirken sistem öz kaynaklarını uygun şekilde kullanmalıdır. İşlemci, bellek, disk ve diğer kaynakların kullanımı etkinlikle yapılmalı, iletişim, grafik sergileme ve yazıcı çıktıları yüksek başarıma sahip olmalıdır.

- Korunmalı olmalı

Yazılım, yetkisiz kişilerin yapabilecekleri değişikliklere izin vermeyecek şekilde nesne kodunu, veri yapılarını, yapılandırma dosyalarını ve belgeleri elektronik ortamda koruma altına alabilmelidir.

- Kullanım kolaylığı

Yazılım, insanların kullanım amaçlarına hizmet ettiğinden birinci öncelik her zaman insan unsuru olmalıdır. Üretilen yazılımı insanlara rahatlıkla kullanabilmesi için gerekli kolaylıklar sağlanmalı, özellikle grafiksel kullanıcı arayüzü düzenli, estetik ve kullanımı kolay olmalıdır.

- Ekonomiklik

Yazılıma sahip olmak, sürekli kullanabilmek, eldeki donanım olanaklarını değerlendirebilmek kullanıcı için önemli ölçütler olabilmektedir.

- İşletim sürekliliği

Yazılımın kapatma açma gerektirmeden çok uzun süre aynı başarım ve doğruluk seviyesinde çalışabilmesidir.

### 2.1.2. Taşınmaya yönelik özellikler

Geliştirilen yazılımın bir başka yerde kullanılabilmesi, bir başka ortama taşınması ya da bir başka geliştirici gruba aktarılması gerektiğinde zorluklarla karşılaşılmaması için yazılımı geliştirirken dikkate alınması gereken özellikler tekrar kullanılabilirlik, uyumluluk ve taşınabilirliktir.

- Tekrar Kullanılabilirlik

Yeni geliştirilen bir yazılımın daha sonra kısmen ya da tamamen yeni bir uygulamada kullanılabilmesidir. Tasarım ve gerçekleştirim bu amaca hizmet edecek şekilde yapılandırılmalıdır.

- Uyumluluk

Bir yazılım ürünü daha önce üretilmiş olan ya da beraber çalışan diğer ürünlerle tam uyumlu olmalıdır. Birbiriyle etkileşen sistemler ortak özelliklere sahip olarak yaratılmalıdır. Bunun için dosya biçimleri, veri yapıları, kullanıcı arayüzleri üzerinde

belirli bir standart oluşturulmalı, tüm yazılımlar bunlara uygun olarak tasarlanıp geliştirilmelidir.

- Taşınabilirlik

Yazılımlar farklı bilgisayar donanım ve yazılımın kullanıldığı ortamlara aktarılabilir olmalıdır. Yeni işletim sistemi sürümlerinin kullanılması, bilgisayar değiştirilmesi gibi durumlarda, daha önce geliştirilmiş olan yazılım en az çaba ile tekrar kullanılabilmelidir.

### 2.1.3. Yenileştirmeye yönelik özellikler

Yazılımın bir kez üretildikten sonra kullanıma sunulması ve kullanımdayken de değişikliğe gereksinim duyması doğaldır. Bu tür yenileştirmenin kolaylıkla yapılabilmesi için yazılımın sahip olması gereken özellikler bakım kolaylığı, genişleyebilirlik ve doğrulanabilirliktir.

- Bakım kolaylığı

Başka bir yazılım geliştirici kişi ya da grup tarafından yazılımın bakımının yapılabilmesi için kaynak kodun anlaşılabilir şekilde yazılmış olması, iyi belgelendirilmesi, sorun çözümlemesinin ve testinin kolay olması gereklidir.

- Genişleyebilirlik

Bir yazılım her zaman için kullanıcı isteklerine göre yeniden uyarlanabilir özelliğe sahip olmalıdır. Ancak yazılımlar büyüdükçe bunu sağlamak güçleşir. Sistemde gerekli değişiklikleri uygun bir şekilde ve kolayca yapabilmek, ona yeni işlevler ekleyebilmek için tasarım ve gerçekleştirimin uygun şekilde yapılması gerekir.

- Doğrulanabilirlik

Yazılımın doğru çalıştığının test edilebilme kolaylığıdır. Testi mümkün olmayan yazılımın gerektiği zaman doğru çalışması da garanti edilemez.

## 2.2. Yazılım Kalitesinde Kalite Ölçümleri

Kalitenin iyileştirilmesi, yol gösterici ya da yenilikçi teknolojinin transfer edilmesiyle sağlanır.

Kalite kriterlerinin ölçülmesi çoğu zaman zordur, hatta bazen de olanaksızdır. Bu nedenle kalite kriterleri sınıflandırılarak ve ölçeklenerek değerlendirilir. Bu şekilde elde edilen sayısal bilgiler kalitenin sağlanmasında ve geliştirilmesinde temel oluşturur. Değerlendirmede dikkate alınabilecek ölçülebilir özelliklerin bir kısmı şunlardır:

- Denetlenebilirlik: Yazılımın standartlara uyum derecesinin denetiminin kolaylığıdır. (Denetim için harcanan süre, denetimde tespit edilen uygunsuzluk sayısı, denetim performansı)
- Doğruluk: Yazılımın arzu edilen işlevleri eksiksiz ve doğru olarak yerine getirebilmesidir (Doğrulama (Verification) – Sağlama (Validation). (Test sürecinde ve müşteriye yayımlama sonrasında tespit edilen uygunsuzluk sayısı)
- Hassaslık: Hesaplamaların ve kontrol işlemlerinin sayısal doğruluk derecesidir.
- Başarım: Yazılımın arzu edilen işlevleri istenen hızda yerine getirebilmesidir. (Performans testi sonuçları)
- Arayüzün yaygınlığı: Arayüzlerin, veri aktarım protokolünün, aktarım hızlarının standartlara uyumluluğudur.
- Verilerin yaygınlığı: Yazılımın tamamında standart veri yapılarının ve veri tiplerinin kullanılmasıdır.
- Bütünlük: İstenen tüm işlevlerin gerçekleştirilme derecesidir.
- Büyüklük: Yazılımı oluşturan kaynak kod satır sayısı, modül sayısı, özkaynak gereksinimi gibi değerlerdir. (Harcanan çaba başına üretim miktarı)
- Tutarlılık: Yazılımın geliştirme projesi boyunca aynı tasarım ve belgeleme teknikleri kullanılmasıdır.
- Hata dayanaklılığı: Yazılımın bir hatayla karşılaşılması durumunda oluşan hasarın büyüklüğüdür.
- Verimlilik: Çalışma sırasındaki başarım derecesidir.

- Genişleyebilirlik: Mimarinin, veri yapılarının ve yordamsal tasarımın genişleyebilme derecesidir (Gereksinimin durağanlığı).
- Donanım bağımlılığı: Yazılımın üzerinde çalıştığı bilgisayar donanımına olan bağımlılık derecesidir.
- İzlenebilirlik: Program çalışırken kendi kendini izleyebilmesi, hatalarını gösterebilmesidir.
- Modülerlik: Program bileşenlerinin işlevsel olarak birbirinden ayrık olmasının derecesidir.
- Kullanım kolaylığı: Programın öğrenilebilmesinin ve işletiminin kolaylık derecesidir. Kullanıcı arayüzünün niteliği en büyük etmendir.
- Bakım kolaylığı: Yazılıma sonradan uygulanabilecek iyileştirici ve düzeltici bakımın ne derece kolay ve kısa sürede yapılabileceğidir. (Yeniden çalışma yüzdesi)
- Belgelendirme: Yazılımın tamamının tasarımının ve kodlarının anlaşılır şekilde belgelendirilmesinin derecesidir. (Firmanın prosedürlerine uyumluluk)
- Müşteri tatmini: Tüm yazılımların belirli bir kullanıcısı ya da müşterisi vardır. Amaç onun tatmin olmasıdır. (Sistem testinde tespit edilen uygunsuzluk sayısı) (Sarıdoğan, 2004).

### 2.2.1. Aday ölçümler

Çizelge 2.1 Kullanılabilecek Olası Yazılım Kalite Ölçümleri

| No | Ölçülebilir Özellikler | Açıklama |
|---|---|---|
| 1 | Gözden Geçirme Etkinliği | Gözden geçirme sürecinin nasıl etkin olduğu (gözden geçirme zamanı başına bulunan hatalara dayanarak) |
| 2 | Hata Yoğunluğu | Bir üründeki ilişkili hata sayısı |
| 3 | Üretkenlik | Harcanan çaba başına üretim miktarı (ürünün büyüklüğü) |
| 4 | Test Performansı | Test için harcanan çaba başına hata sayısı |
| 5 | Gereksinimlerin Durağanlığı | Eklenen, silinen ve değiştirilen gereksinimlerin yüzdesi |
| 6 | Müşteri Destek Süresi | Kabul edilen müşteri probleminin alınmasından düzeltilen kodun hedef bölge için yerleştirilmesine kadar geçen süre |
| 7 | Birikmiş Yönetim İndeksi | Kapatma oranıyla ilişkili hata raporu yaratılma oranının miktarı |
| 8 | Denetim için Harcanan Çaba Yüzdesi | Toplam harcanan çaba ile ilgili olarak denetim için harcanan çabanın miktarı |
| 9 | Denetim Performansı | Toplam harcanan çaba ile ilgili olarak denetim boyunca tespit edilen uygunsuzluk sayısı |
| 10 | Firma içi Gözden Geçirme için Harcanan Çaba İndeksi | Ürün büyüklüğüyle ilgili olarak firma içi gözden geçirme için harcanan çaba miktarı |
| 11 | Hata Raporlama Gelişimi | Bir hata raporunun başlangıcı ile çözülmesi arasında geçen süre |

## 3. İSTATİSTİKSEL KONTROL

Ürünün kalitesini, onu üreten sürecin kalitesi belirler, dolayısıyla süreç odaklı kalite yaklaşımı egemendir. Müşteriye sağlanan ürün/hizmet, yönetilen süreçlerin çıktılarıdır. Sürecin normal sapmasının ölçüm yöntemine ihtiyaç vardır ve bu nedenden dolayı erken tespit sistemi, sürecin üzerine kurulabilir. Bu noktada süreç yönetimi temelli düşünce, bir yöntem kullanımını öne çıkarmaktadır.

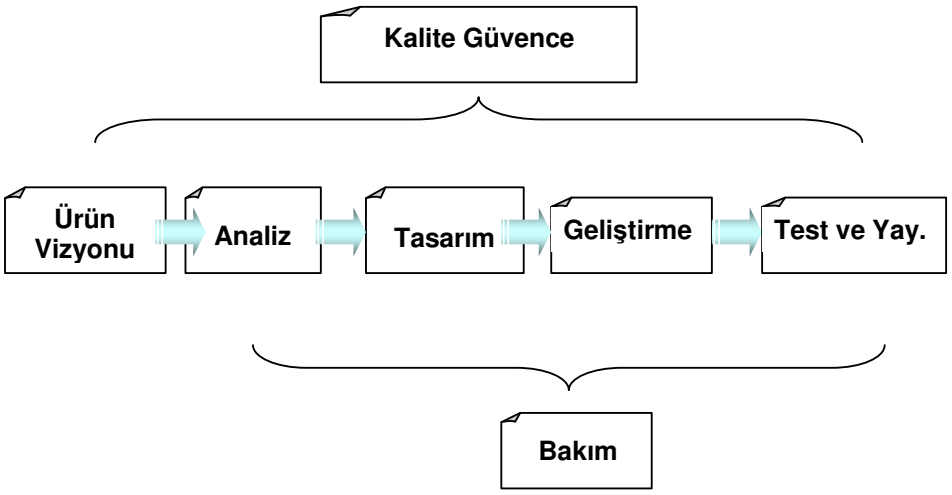
Kullanılacak yöntemi oluşturan faktörlerden biri de gerçekleştirilen ölçümdür. Ölçüm, hem değişkenlere hem de özelliklere uygulanabilir. Değişkenler için ölçüm, herhangi bir değere sahip olabilen bir ürünün ölçülmesi; özellik verisi için ölçüm ise üründe şuan itibariyle mevcut olabilen şeylerin sayılması olarak tanımlanabilir. Bu iki ölçüm kategorisinin doğası gereği farklı istatistiksel analizlerin uygulanması gerekir.

Bir süreç davranışı içerisindeki değişkenlik istatistiksel terimler altında tahmin edilebilirse, bu sürecin kontrol altında olduğu söylenebilir. Bu durum, (belirli limitler içerisinde) aynı süreci uyguladığımız bir sonraki seferde çıktının ne olacağını tahmin edebilmemiz demektir. Bu gerçekleştirildiğinde, oldukça gerçekçi proje planları hazırlanabilir, daha iyi fiyat tahminleri ve daha gerçekçi faaliyet planları yapılabilir.

Süreç, istatistiksel kontrol altında geliştirilirse, sapmalar kontrol altına alınabilinir. Bununla şu ima edilmektedir: Sürecin geçmiş performansı kullanılarak, gelecekteki performansı tahmin edilebilir ve hatta müşteri memnuniyetiyle ilişkili olarak sürecin uygulanabilirliği de gözlenebilir (Jacob and Pillai, 2003).

Süreç davranışındaki sapmaları hesaplamak için sürecin çıktısı olarak sunulan özellik ve değişkenler tanımlanabilir. Nelerin istatistiksel kontrol altında inceleneceğine geçmeden önce yazılım yaşam süreci ve bu döngü içerisindeki safhaları kısaca tanımlamak gerekir. Yazılım yaşam (geliştirme) süreci, bir bilgi sistemi için sistem gereksinimlerinin ortaya konulmasından, ilgili sistem gerçekleştiriminin yerine getirilmesine ve hatta işletim ve bakım safhasına kadar geçen süreci ve bu süreçte yapılması gereken aktiviteleri tanımlayan işlevler bütünüdür. Yazılım yaşam süreci Şekil 3.1.'de gösterildiği gibi basit olarak aşağıdaki safhalardan oluşmaktadır:

YAZILIM YAŞAM (GELİŞTİRME) SÜRECİ




Şekil 3.1. Yazılım Yaşam (Geliştirme) Süreci Safhaları

1. Gereksinim Analizi safhasında aşağıdaki maddeler hedeflenir:
    a. Uygulamanın çözüm getirmesi beklenen gereksinimlerin anlaşılması,
    b. Uygulamanın neler yapacağının belirlenmesi,
    c. Uygulamanın neler yapmayacağının belirlenmesi,
2. Tasarım safhasında, fiziksel özellikler, performans kriterleri, veri kısıtları, programlama dilinin özellikleri ve kullanılacak sistem mimarisi belirlenir.
3. Geliştirme safhasında, yazılımın kodlanması gerçekleştirilir.
4. Test ve Yayımlama (Deployment) safhasında yazılım testleri ve testlerin başarıyla tamamlanması sonrasında ürünün müşteriye yayımlanması gerçekleştirilir.
5. Bakım safhasında, müşteriden gelen istek ve / yada hatalar belirlenen bakım süreci içerisinde yazılım yaşam sürecinin herhangi bir safhası tekrarlanarak karşılanır.

Yazılım ürününün iyileştirilmesi, Şekil 3.1'de de görüldüğü gibi tüm süreçleri içine alan Kalite Güvence (Quality Assurance) faaliyetleri ile gerçekleştirilir. Yazılımın

iyileştirilmesi, sürekli olarak yazılım kalitesinin iyileştirmesini destekler. Yazılımda kalite güvencenin amacı, ürün veya üretim süreçlerinin firma süreçleri veya proje gereksinimleri ile uyumlu olduğunu kontrol etmek ve sağlamaktır. Bunu gerçekleştirmek için kullanılan aktiviteler, ürünün iyileştirilmesi daha doğrusu nitelikli ürün üretilmesi için oldukça önemlidir. Bu aktiviteler:

- Doğrulama (Verification),
- Sağlama (Validation),
- Denetimdir (Auditing).

Tüm bu aktiviteler, yazılım yaşam sürecinin en başından itibaren tüm süreçlerinde kullanılır.

Doğrulama (verifikasyon), ürün/hizmetlerin doğru olduğunu temin etme çabasıdır. Doğrulamanın standart tanımında şu soru sorulur "Ürünü doğru olarak ürettik mi?". Doğrulama, yazılım ürününün doğru yollarla geliştirildiğinden emin olma sürecidir. Doğrulama boyunca, yazılım ürünü, bir veya daha fazla kişi tarafından uygunsuzlukları tespit etmek ve ortaya çıkarmak amacıyla gözden geçirilir / incelenir. Böylece, projenin başarısızlığa uğrama olasılığı, potansiyel hataların önlenmesiyle giderilmesi sağlanır. Doğrulama için kullanılan aktiviteler, gözden geçirme ve testtir. Yazılım yaşam süreci boyunca veya sonunda doğrulama faaliyetleri gerçekleştirilir (Gereksinim Analizi Doğrulaması, Tasarım Doğrulaması, vb.).

Gereksinim aşaması doğrulaması, aşağıdaki soruların sorulmasıyla gerçekleştirilir:

- Yazılım gereksinimleri tutarlı mı?
- Gereksinimler güvenlik ihtiyaçlarını karşılıyor mu?
- Gereksinim dokümanı, firmanın belirlediği gereksinim analizi dokümanı standardına uygun hazırlanmış mı?

İhtiyaç duyulursa gereksinim doğrulama amacıyla prototipleme yöntemi de kullanılabilir. Bu durumda, hazırlanan prototip gözden geçirilir.

Tasarım aşaması doğrulaması, aşağıdaki soruların sorulmasıyla gerçekleştirilir:

- Gereksinimler ve tasarım arasında izlenebilirlik sağlanmış mı?
- Tasarım, güvenlik ve diğer kritik ihtiyaçları göz önünde bulundurmuş mu?
- Tasarım dokümanı, firmanın belirlediği tasarım dokümanı standardına uygun hazırlanmış mı?

Kodlama aşaması doğrulaması, aşağıdaki soruların sorulmasıyla gerçekleştirilir:

- Tasarım ve gereksinimler arasında izlenebilirlik sağlanmış mı?
- Kod, kodlama standartlarına uygun olarak yazılmış mı?
- Kod, girdileri, çıktıları, arayüzleri, mantıksal akışı, hata tanımlarını, önlemlerini ve hataların ele alışlarını içeriyor mu?
- Kod, güvenlik ve diğer kritik ihtiyaçları karşılayan parçaları içeriyor mu?

Doğrulama kapsamında gerçekleştirilen gözden geçirme ve test faaliyetleri yazılım gerçekleştirim çalışmalarında görev almamış tarafsız personel tarafından yapılır.

Doğrulama sonucunda elde edilen veriler bu sürecin iyileştirilmesinde kullanılır (Gözden geçirme faaliyeti sonucunda dönüş sayısı, Gözden geçirme faaliyeti sonucunda değişiklik sayısı, Doğrulama sonucunda geri dönen test durumu sayısı vb.).

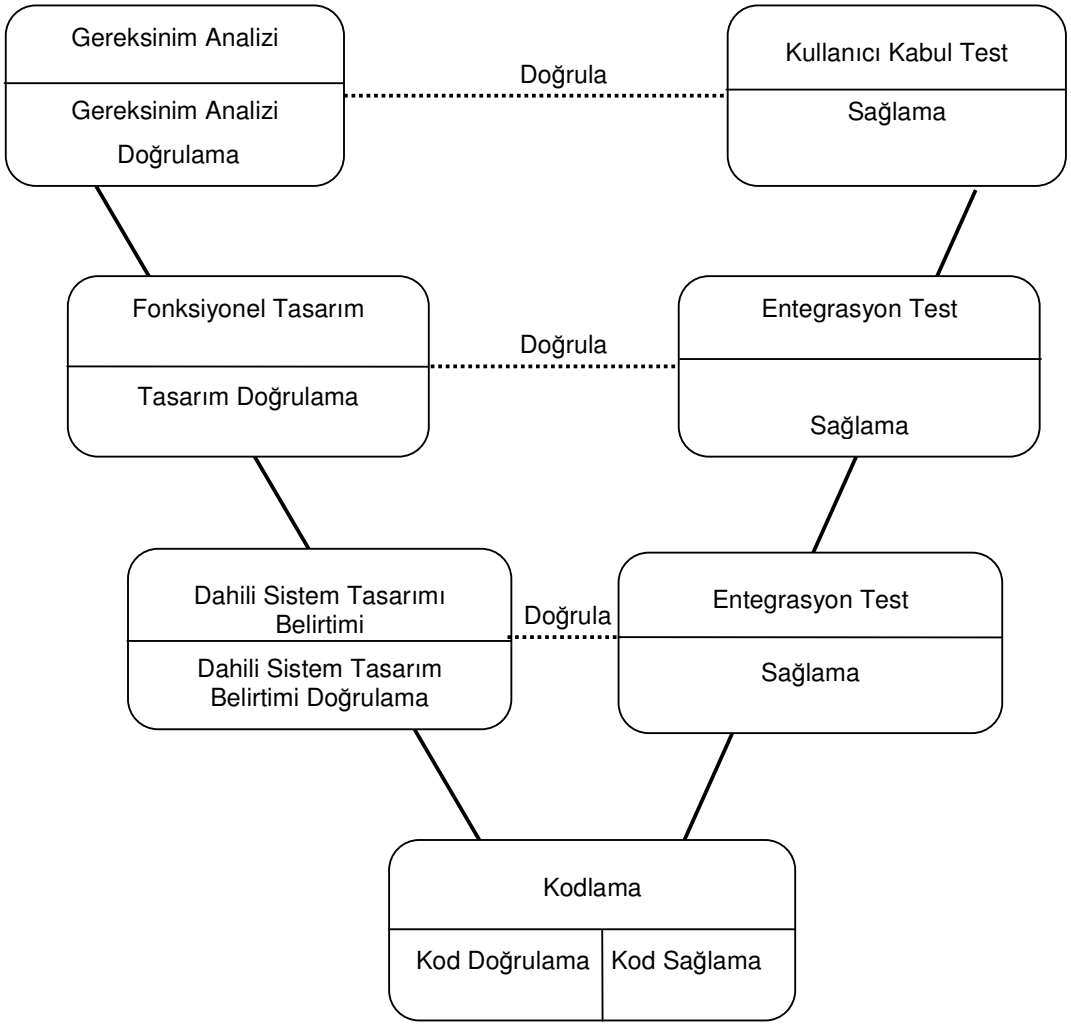
Sağlama (validasyon), ise amaca uygunluk, kullanıcıya sağlanan değer, yani doğru ürün/hizmetin sağlanmasıdır. Yazılım ürününün, kullanıcı tarafından belirlenen fonksiyonel gereksinimlerin tümünü karşılayıp karşılamadığı incelenir. Sağlama, doğrulamanın hemen sonrasında, her bir yazılım yaşam süreci safhası sonunda gerçekleştirilir (Gereksinim Analizi Sağlaması, Tasarım Sağlaması, Kod Sağlaması, Test Sağlaması, vb. ) (Parekh, 2005).

Sağlama aktivitesi sırasında, müşteriden, dokümanların yeterliliği, tamamlığı, tutarlılığı, muğlâk olmama gibi kriterleri değerlendirmesi istenmelidir. Uygulama için gerçekleştirilecek sağlama faaliyetleri, daha önce müşteri tarafından onaylanan “Test Planında” yer alan test durumları kullanılarak yine müşteri tarafından kabul testleriyle gerçekleştirilir.

İhtiyaç duyulursa uygulamanın prototipleri hazırlanarak sağlama amacıyla müşteriye sunulabilir.

Sağlama sonucunda elde edilen veriler bu sürecin iyileştirilmesinde kullanılır (Dokümanların müşteriden dönüş sayısı, Dokümanların müşteriden dönüşü sonucunda değişiklik sayısı, Sağlama sonucunda geri dönen test durumu sayısı, vb.).

Yazılım proje geliştirme yaşam süreci içerisinde gerçekleştirilen doğrulama ve sağlama aktiviteleri Şekil 3.2'de Doğrulama & Sağlama modeli ile gösterilmektedir.




Şekil 3.2 Doğrulama & Sağlama Modeli

Denetim ise yazılım kalite güvencesinde kullanılan, ürünün ve sürecin etkin olarak uygulandığını ve sürdürüldüğünü teyit etmek ve standartlara uygunluğunu kontrol etmek için planlı aralıklarla sürdürülen üçüncü aktivitedir. Denetimler, iç denetim

eğitimi almış firma çalışanları tarafından gerçekleştirilir. Denetim sonunda elde edilen veriler denetim sürecinin iyileştirilmesi için kullanılır (Gerçekleştirilen denetim sayısı, denetim için harcanan süre, denetim sonunda bulunan uygunsuzluk sayısı vb.).

En az hatayla kaliteli ürünün müşteriye yayımlanabilmesi için uygulanan tekniklerden biri de hiç kuşkusuz hem doğrulama hem de sağlama faaliyetleri içerisinde yer alan test sürecidir. Yazılım firmaları tarafından uygulanan test türleri, en geniş kapsamıyla, aşağıda detaylandırılmıştır:

- Kara-kutu Testi

Bu tür testlerde yazılımın yapısı, tasarımı veya kodlama tekniği hakkında herhangi bir bilgi olması gerekli değildir. Yazılımın, müşteri isteklerini karşılayıp karşılamadığı ve işlevselliği test edilir.

- Beyaz-kutu Testi

Bu tür testler, uygulama kodunun iç mantığı üzerindeki bilgiye bağlıdır. Yazılım kodundaki deyimler, akış denetimleri, koşullar vb elemanlar test edilir.

- Birim Testi

Mikro ölçekte yapılan bu testte, özel fonksiyonlar veya kod modülleri test edilir. Bu test, test uzmanlarınca değil programcılar tarafından yapılır ve program kodu ayrıntılarına ve içsel tasarım biçiminin bilinmesi gereklidir. Uygulama kodu çok iyi tasarlanmış bir mimaride değilse oldukça zor bir testtir.

- Evrimsel Entegrasyon Testi (Evolutionary Integration Testing)

Uygulamanın yeni işlevsel elemanları eklendikçe sürekli test edilmesidir. Bu testte uygulamanın tüm parçaları tamamlanmadan önce yeni eklenen parçanın işlevselliğinin, öncekilerden yeteri ölçüde bağımsız şekilde çalışıp çalışmadığı sınanmaktadır. Test uzmanları ve/veya programcılar tarafından yapılan bir testtir.

- Entegrasyon Testi (Integration Testing)

Bir uygulamanın farklı bileşenlerinin beraberce uyum içinde çalışıp çalışmadığını sınamak için yapılan bir testtir. Bileşenler, modüller, bağımsız uygulamalar,

istemci/sunucu uygulamaları biçiminde olabilirler. Bu tür testlere, özellikle istemci/sunucu uygulamaları ve dağıtık sistemlerin testinde başvurulmaktadır.

- İşlevsellik Testi (Functionality Testing)

Bir uygulamanın işlevsellik gereksinmeleri üzerine odaklandırılan kara-kutu testidir. Bu tür testler, test uzmanları tarafından yapılır, ancak bu uygulama yayınlanmadan önce kodların programcılar tarafından incelenmeyeceği anlamına gelmez. Testin herhangi bir aşamasında program kodlarının da incelenmesi gerekir.

- Sistem Testi (System Testing)

Uygulamanın, tanımlanan gereksinmelerin tümünü karşılayıp karşılamadığını sınamak için yapılan bir kara-kutu testidir.

- Regresyon Testi (Regression Testing)

Uygulama ve uygulama ortamlarında gerekli değişiklikler ve sabitlemeler yapıldıktan sonra yeniden yapılan testlere ilişki (regresyon) testi denilir. Böylece, önceki testlerde belirlenen sorunların giderildiğinden ve yeni hatalar oluşmadığından emin olunur. Uygulamanın kaç kez yeniden test edilmesi gerektiğini belirlemek güçtür ve bu nedenle, özellikle uygulama geliştirme döneminin sonlarına doğru yapılır.

- Kabul Testi (User Acceptance Testing)

Son kullanıcı veya müşteri veya isteklerine dayanan son test işlemidir. Ayrıca, son kullanıcıların belli bir süre kullanımından elde edilen sonuçlar üzerinde de yapılabilmektedir.

- Yük Testi (Load Testing)

Uygulamanın çok ağır yükler (veya işlem yoğunluğu) altında test edilmesidir. Örneğin, bir Web sitesi için sistem tepkisinin hangi noktada azaldığı veya yanıt veremez olduğunu belirlemek için yapılan testler gibi.

- Verim (Performans) Testi (Perfromance Testing)

Yukarıda da belirtildiği gibi, bu test 'zorlanım' ve 'yük' testi ile eş anlamlı olarak da kullanılabilmektedir. Ancak, yapılması gereken performans testinin ne olduğunun, gereksinmeler veya kalite güvencesi veya test planlarında açıklanmış olmalıdır.

- Kullanabilirlik Testi (Usability Testing)

Kişisel yargılara göre değişen bir test olup hedeflenen son kullanıcı veya müşteri kitlesine bağlı olarak değişir. Programcılar ve test uzmanları genellikle bu tür testler için uygun değildir, yani bu testlerin doğrudan son kullanıcılar üzerinde yapılması gerekir.

Bu test genellikle geliştirme sürecinin erken aşamalarında yapılır, böylelikle uygulamanın kullanıcı arayüzlerinde önemli değişiklikler yapılması mümkün olur.

- Güvenlik Testi (Security Testing)

Yazılımın, gerek iç ve gerekse dış kaynaklı yetkisiz erişimlere, kötü amaçlı kullanımlara karşı korunması ya da güvenliğini test etmek için yapılır. Çok karmaşık ve özel test tekniklerinin kullanıldığı bir test türüdür.

- Uyumluluk Testi (Compatibility Testing)

Yazılımın özel bir donanım, yazılım, işletim sistemi, ağ veya ağ protokolü vb. ortamda beklenen şekilde çalışıp çalışmadığını sınamak için yapılan testlerdir. Örneğin, Almanca bir uygulama sürümünün Fransızca bir Windows 2000 platformu üzerinde düzgün şekilde çalışıp çalışmadığı; bir iletişim yazılımının en yaygın kullanılan 100 modemle sorunsuz çalışıp çalışmadığı; bir uygulamanın değişik sürümlerinin yaygım kullanılan platformlarda (AIX, NT, Linux vs) sorunsuz çalışıp çalışmadığı, ya da bir video uygulamasının değişik kartlar üzerinde çalışıp çalışmadığı gibi testler birer uyumluluk testi örneğidirler. Bu test için, test firmasının geniş bir donanım ve yazılım parkı ve birikiminin olması tercih edilmektedir.

- Doyum Testi (Satisfaction Testing)

Yazılımın, son kullanıcı veya müşteri tarafından beğenilip beğenilmediğini, ya da ihtiyaçlarını karşılayıp karşılamadığını belirlemek için yapılır.

- Kurma/Kaldırma Testi (Install/Remove Testing)

Bu test, yazılımın kurulması ve kaldırılması ile ilgili tüm seçenekler ve özelliklerin düzgün şekilde çalışıp çalışmadığını sınamak için yapılır. Kurulumda, tüm gerekli dizinler ve bunlarda yer alacak dosyaların (dll, .cfg, .txt vb) oluşturulması gereklidir.

- Ağ Testi (Network Testing)

Çok kullanıcılı uygulamaların ağ ortamında gerçekten ağ üzerinde çalışabilme yeteneklerini ortaya koymak için yapılan bir testtir. İstenirse, farklı ağ işletim ortamları ve iletişim kuralları altında test yapılması tercih edilmelidir.

- Alfa Testi (Alfa Testing)

Bitirilme aşamasına yakınlaşmış olan bir uygulama için yapılan testtir. Bu test sonucunda ürün üzerinde küçük değişiklikler yapılabilir. Programcılar veya test uzmanlarınca değil, son kullanıcılar tarafından yapılır.

- Beta Testi (Beta Testing)

Uygulamanın tamamlanması ve zorunlu testleri yapıldıktan sonra, son sürümü çıkarmadan önce hatalar ve/veya sorunları saptamak üzere yapılan testlerdir. Programcılar veya test uzmanlarınca değil son kullanıcılar tarafından yapılır (Cebeci, 2001).

Birim test boyunca bulunan hata sayısı, gereksinim analizi safhası sonrasında değiştirilen gereksinimlerin sayısı, geliştirme safhası boyunca tespit edilen hata tipleri, öncelikleri ve süreç içerisinde hangi aktivite gerçekleştirilirken (gözden geçirme, denetim, test vb.) tespit edildiği bilgisi, gibi tüm bu gösterimler yazılım yaşam sürecinin davranışını anlamak için kullanılabilir.

### 3.1. İstatistiksel Süreç Kontrolü (Statistical Process Control) ve Tarihçesi

İstatistiksel Süreç Kontrolü (İSK), bir süreci sürekli denetleme ve süreçteki değişkenliği kontrol altına almada kullanılan bir kalite kontrol yöntemidir. Müşteri şartlarının yerine getirilip getirilmediğine ve sürecin kendi ürettiği değişkenlik sınırları içinde olup olmadığına karar vermede bir araç olarak kullanılmaktadır. İSK, sürecin kontrol altında olup olmadığını tespit eder, ancak sürecin kontrol dışı olmasına ait nedenleri ortaya koyamaz. Bu noktada İSK, bir uyarı sistemi olarak çalışmaktadır (Durman, 2005).

İstatistik bilim dalı içerisinde en çok uygulama bulan alanlardan birinin de İSK olduğu söylenebilir. İSK, bir ürünün en ekonomik ve yararlı bir şekilde üretilmesini sağlamak, önceden belirlenmiş kalite kriterlerine uygunluğunu ve standartlara bağımlılığı hedef almak, kusurlu ürün üretimini en aza indirmek amacıyla istatistik prensip ve tekniklerin üretimin bütün safhalarında kullanılmasıdır (Akın, 1996).

İSK, istatistiksel terimler altında süreç kontrolünü sağlamayı amaçlayan bir yöntemdir.

İSK' da istatistik, bir bütünün tamamını kontrol etmek yerine bütünden örnekler alarak sonuçlara göre bütün hakkında tahminde bulunmak için kullanılan araçları ifade eder. Süreç, ürünün belirlenen nitelikte elde edilebilmesi için belirlenen yöntem, standart, prosedür ve insan gücünün bütününü içerir. Kontrol, süreçteki verilerin ölçümünde ve analizinde istatistiksel tekniklerin uygulanması anlamını taşır.

İSK, muayene yolu ile %100 kalite güvencesinin sağlanamamasından dolayı ortaya çıkmıştır. Burada amaç ürünü kontrol etmekten ziyade, ürünü üreten sistemi yani süreci kontrol etmektir. Üretilen ürünün özellikleri onu üreten sürecin bir fonksiyonudur. Daha değişik bir ifade ile süreçle ürün arasında bir sebep-sonuç ilişkisi vardır. Eğer tüm süreç değişkenleri kontrol altına alınabilinirse ürünün özellikleri de kontrol altına alınabilir (Aksoy, 2005).

İSK sadece bir araç olmayıp, aynı zamanda kalite problemlerine sebep olan değişkenliklerin azaltılması için de bir stratejidir. İSK' nın iyileştirilebilmesi için sadece iyi bir istatistiksel süreç kontrolünün sağlanması yeterli değildir, bu durum gerçekleştikten sonra sürecin iyileştirilmesi de gerekmektedir. Japonların Kaizen adını verdikleri sürekli iyileştirme, bu düşünce altında oluşmuştur. Sürekli iyileştirme (Kaizen) iki aşamalı bir süreçtir.

Birinci aşama sürecin durgun (stabil) hale getirildiği durumdur. Bu aşamanın safhaları, "Standartlaştır - Uygula - Kontrol et - Önlem al" dır. Bu çevrim sağlanıp süreç kontrol altına alındıktan sonra süreçte iyileşmeyi ve yenilenmeyi sağlayacak olan döngü "Planla - Uygula - Kontrol et - Önlem al" yani ikinci aşama devreye girmektedir. Bu iki çevrim asla sonu olmayan bir döngüdür. (Gençyılmaz ve Zaim, 1999)

Yazılım sürecinde İSK ‘de kullanılan temel istatistiksel teknikler:

- Kalite Kontrol Kartları (Control Chart),
- Neden-Sonuç Diyagramı (Cause and Effect Diagram),
- Dağılım (Saçılım) Diyagramı (Scatter Diagram),
- Histogram,
- Pareto Diyagramı (Pareto Diagrams),
- Kontrol Tabloları (Control Sheet),
- Gruplandırma (Stratification).

biçimindedir. Yukarıda verilen teknikler İSK ‘nin sadece kontrol kartları ile sınırlı olmadığını diğer istatistik tekniklerini de kapsadığını göstermektedir. Genellikle kontrol kartlarıyla birlikte öteki İSK teknikler de kullanılır. Bununla birlikte kontrol kartları İSK’ nın temel taşıdır. Bu tekniklerin açıklamaları Ek 3’de verilmiştir.

İSK her zaman kontrol kartlarının kullanımını gerektirir (Gupta, 2002).

Kontrol kartları “genel nedenler” ve “özel nedenler” in sebep olduğu değişkenlikler arasındaki farklılık için bir istatistiksel yöntem sağlar.

Değişkenliğin genel nedeni, süreç bileşenleri (insan, prosedür, standart ve yöntemler) arasındaki normal ya da rasgelelikten kaynaklanan etkileşim sonucudur. Genel nedenlerden kaynaklanan süreç performansındaki sapma rasgeledir fakat tahmin edilebilir sınırlar içerisinde farklılık gösterir. Benzer bir durumda süreç performansı durgun ve zaman içerisinde sabit bir model ile nitelendirilir ve süreç “kontrol altında” ya da “durgun” ya da “tahmin edilebilir” olarak adlandırılır. Diğer taraftan, özel nedenler süreç bileşenlerinin bir ya da daha fazlasında ani ya da devamlı olarak normal olmayan değişiklikleri işaret eder. Bu değişiklikler sürecin doğasında var olan bir parçası ve sürece girdi olan çevre, kendi başlarına süreç adımları ya da süreç adımlarının çalıştırıldığı durumlar gibi normal olamayan değişikliklerden kaynaklanabilen sadece geçici bazı durumlarına özel parçası değildir. Belirgin şekilde

süreç performansını ve çıktılarını etkilerler ve süreç benzer durumda "kontrol dışı" ya da "durgun olmayan" süreç olarak adlandırılır.

Bir yazılım sürecinde yaygın olarak ölçülemeyen özelliklere ilişkin p, np, c, u kontrol kartları kullanılmaktadır. Ölçülebilen özelliklere ilişkin kontrol kartları ise $\overline{X}$, R, ve S' dir.

### 3.2. Veri Toplama Teknikleri

Süreç iyileştirme projesine başlamadan önce doğru ve gerekli verinin toplanması oldukça önemlidir. Bu konuda dikkat edilmesi gerekli konular şu başlıklar altında toplanabilir:

- Hedefler Net ve Açık Tanımlanmalıdır: Veri toplanmadan önce, bu verinin ne işe yarayacağı ve hangi amaçlar doğrultusunda kullanılacağı belirlenmelidir. Bunun için de kontrol edilecek süreçlerin önceden belirlenmiş olması şarttır. Kalitede veri toplamanın amaçları şunlar olabilir:
    - Üretim sürecinin gözlenmesi ve denetimi (Sürekli iyileştirmeye yönelik)
    - Uygunsuzluk analizi
- Ölçümlerin güvenilirliği sağlanmalıdır: Ölçüm yöntemleri, ölçüm sıklığı gibi faktörler göz önüne alınmalıdır. Ayrıca Otomatik Tanıma / Veri Toplama (OT-VT) tekniklerinin kullanılması, veri toplarken oluşacak hız kaybı ve kişiye bağlı hataların önüne geçecektir.
- Tüm bunlardan sonra toplanan verilerin, kullanılacak istatistik yöntemine uygun olarak kaydedilmesi ve sonraki işlemleri kolaylaştıracak şekilde bir araya getirilmesi gerekir. Verilerin alındığı tarih/saat, veriyi kaydeden kişi, üretimin yapıldığı program ve üreten kişi, üretilen ürün gibi kritik bilgiler, mutlaka veriyle birlikte işlenmelidir. Ayrıca verinin görsel olarak analizini çabuklaştıracak şekilde düzenlenmesi de (örneğin çetele tablosu tutulması) hataların daha çabuk tespitini sağlar (Altısigma, 2003a).

Veriler, yazılım firmasının tercihine göre günlük, haftalık, aylık ya da yazılım ürününün yayımlama aşamalarında alınabilir.

### 3.3. Standartlar ve İSK

Standardizasyon, yazılım mühendisliğinde ve özellikle yazılım ölçüm alanlarında kilit bir rol oynamaktadır. Standartlar firmalara aynı fikirde olmalarını ve iyi-tanımlanmış uygulamalar ve teknolojilerin kullanılmasını sağlar. Standardizasyon, terminolojiler, uygulamalar, iş geleneklerinde fikir birliği şartı ile hep beraber hareket etmeyi sağlamak için kullanılan bir araçtır (Garcı´aa et al., 2005).

Yazılım sektöründe kullanılan ve bir yöntem olarak benimsenen Yazılım Yaşam Döngüsü Süreçleri (Software life cycle processes - ISO/IEC 12207) standardı sürecin iyileştirilmesi için "Süreç İyileştirmesi" maddesi gereğince aşağıdaki görevleri vurgulamaktadır (ISO/IEC 12207:1995):

- Kullanılan süreçlerin güçlü ve zayıf noktalarının anlaşılmasını sağlamak amacıyla tarihsel, teknik ve değerlendirme verileri toplanmalı ve analiz edilmelidir. Bu analizler süreçlerin iyileştirilmeleri için geri bildirim olarak, projelerin (ya da alt projelerin) yönetimindeki değişiklikleri yönetmek ve teknolojik ilerleme ihtiyaçlarını tanımlamak için kullanılmalıdır.

- Kalite maliyet verisi toplanmalı, bakımı yapılmalı ve bir yönetim faaliyeti gibi firmanın süreçlerinin iyileştirilmesi için kullanılmalıdır. Bu veri hem problemlerin çözülmesi ve önlenmesi hem de yazılım ürünleri ve hizmetindeki uygunsuzlukların kurulum maliyet amacına hizmet etmektedir.

Yukarıdaki iki madde, veri toplanmasının ve analizinin önemini vurgulamaktadır.

Yazılım sektöründe kullanılan diğer bir standart Amerikan Ulusal Standart (American National Standard – ANSI' dir. Bu standart, kalitede kullanılan, veriyi analiz ederken üretim boyunca kaliteyi kontrol altında tutmak için kullanılan kontrol kartlarını anlatır. Ayrıca, kontrol kartın önemini vurgulayan ANSI / Kalite Kontrol için Amerikan Topluluğu (American Society for Quality Control - ASQC) Z1.1, Z1.2 ve Z1.3 1985 standartları da kontrol kartların önemini ve nasıl kullanılacağı belirten standartlardır.

ISO 9001:2000 Kalite Yönetim Sistemleri - Gereksinimleri (Quality management systems - Requirements) uluslararası standardının "Ölçüm, analiz ve iyileştirme" başlığı altında yer alan ve bir alt başlık olan "Veri Analizi" maddesinde, toplanan uygun verilerin analiz edilmesi belirtilmiştir. ISO 9001:2000 standardını yorumlayan

bilim adamları bu analiz tekniklerinin uygun istatistiksel teknikler kullanılarak gerçekleştirilmesi gerektiğini belirtmişlerdir. ISO/IEC 9003:2004 standardı da kalite güvence ve yönetiminde ölçümün önemini vurgulamaktadır.

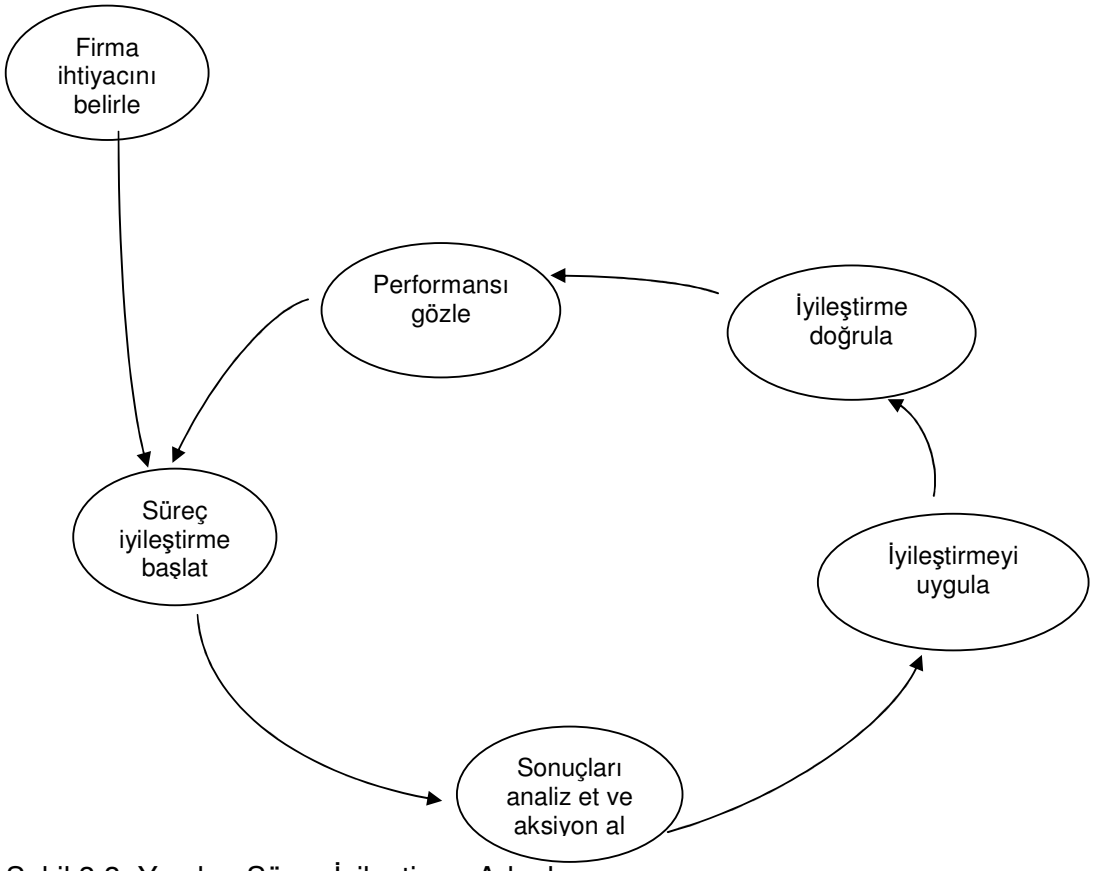
### 3.4. Yazılımın İyileştirilmesi

Yazılım Süreç İyileştirmesi, firmanın etkinliğini (iç ihtiyaçlar) ve teslim edilen ürünü iyileştirmek (dış baskılar) için arzu edilir:

Dolayısıyla süreç iyileştirme ile

1. Sürekli ve hızlı değişen teknolojilere cevap vermek,
2. Zaman içerisinde gelişen ve kayıt altına alınması ihtiyacı duyulan süreçleri tanımlamak ve dokümante etmek,
3. Üretilen yazılımın güvenilirliğini ve kalitesini arttırmak,
4. Maliyeti azaltmak,
5. Ölçümler için fırsat yaratmak,
6. Olgun olmayan süreçlere karşı olgun süreçler istemek,
7. İyileştirme için insana değil sürece odaklanma kazanımları mümkündür.

Yazılım süreç iyileştirmeyi gerçekleştirmek için izlenen adımlar Şekil 3.3. ile gösterilmektedir.




Şekil 3.3. Yazılım Süreç İyileştirme Adımları

### 3.4.1. Yazılım olgunluk ve yetenek modelleri

Yazılım üretimi diğer alanlara kıyasla oldukça farklı özellikler gösterir. Bu sektör farklı disiplinler tarafından mühendislik, sanat, zanaat, bilim dalları ile ilişkilendirilir. Yazılım sektöründe diğer sektörlere kıyasla tekrar oldukça az, farklı kişilerin ürüne etkileri daha fazladır. Tüm hataları oluşmadan gerçekleştirmek proje koşulları ve maliyetleri içerisinde olanaklı değildir. Yazılımda üretim teknikleri sabit olduğu için ürünün kalitesini, genel olarak onu üreten sürecin kalitesi belirler. Bu nedenle yazılım sektöründe ürün odaklı kalite yönetiminden çok süreç odaklı kalite yaklaşımı hâkimdir. Müşteriye sağlanan ürün ve hizmet, yönetilen süreçlerin çıktılarıdır. Süreç yönetimine odaklı düşünce, yazılım geliştirme faaliyetleri yürüten firmalarda yöntemlerin kullanımını avantaj ve zorunluluk olarak öne çıkartmaktadır (Altınordu, 2002).

Yazılımın bitiminde uygulanan inceleme (review), teftiş (inspection), gözlem (walkthrough), denetim (audit) diye tanımlanan değişik kalite çalışmaları yer almıştır. Günümüzde yazılım sektörü, yazılımın her aşamasında kalite olgusunu tam

anlamıyla gerçekleştirebilmek için değişik modelleri, yöntemleri, resmi standartları kullanmaya çalışmakta, bunların dünya çapında kullanılır olması için çaba vermektedir.

Yazılım projelerini zamanında, beklenen maliyetle ve söz verilen tüm fonksiyonlarıyla teslim etmekte yaşanan güçlükler yazılım firmalarını süreç iyileştirme çalışmalarına yöneltmiştir. Bu nedenle günümüzde pek çok firma, yazılıma yönelik geliştirme, bakım gibi aktivitelerini düzenlemek için Yetenek Olgunluk Modeli (Capability Maturity Model - CMM), Yetenek Olgunluk Modeli Entegrasyonu (Capability Maturity Model Integration - CMMI) ve Yazılım Süreç İyileştirme ve Yetenek Belirleme (Software Process Improvement and Capability dEtermination, SPICE - ISO/IEC 15504) gibi yazılım süreç iyileştirme modellerini kullanan çalışmalar gerçekleştirmektedir. Yazılım süreç iyileştirme çalışmaları en kısa tanımlama ile firmanın yazılım aktivitelerini gerçekleştirirken kullandıkları mevcut yöntemlerin değerlendirilmesini, daha etkin (maliyet, zaman ve güvenilirlik boyutlarında) gerçekleştirilmesi için iyileştirmeyi ve yeni yöntem ile elde edilen sonuçları ölçerek ne kadar iyileşme sağlandığının gözlemlenmesini içerir.

Yazılımın ölçülmesi, yazılım mühendisliğinde gittikçe artarak önemli bir rol oynamaktadır. Günümüzde yazılım ölçümleri, büyük veritabanı projeleri için yüksek kalitede tahmin sistemlerinin etkili olmasını, yazılım geliştirme ve bakım projelerinin iyileştirilmesini ve anlaşılmasını, kuşkulu alanları aydınlatarak sistem kalitesinin değerlendirilmesini ve sürdürülmesini, çalışmalarında araştırmacılara en iyi yolu belirlemelerine yardımcı olmayı sağlar. Yazılım ölçümleri, yazılım firmalarında yazılım süreç iyileştirmenin kurumsallaşmasına ve değerlendirilmesine yardım etmek için önemli araçlardır. Aslında yazılım ölçümü CMM, CMMI ve SPICE gibi öncü yazılım yetenek ve olgunluk modellerinin temel taşlarından bir kısmıdır.

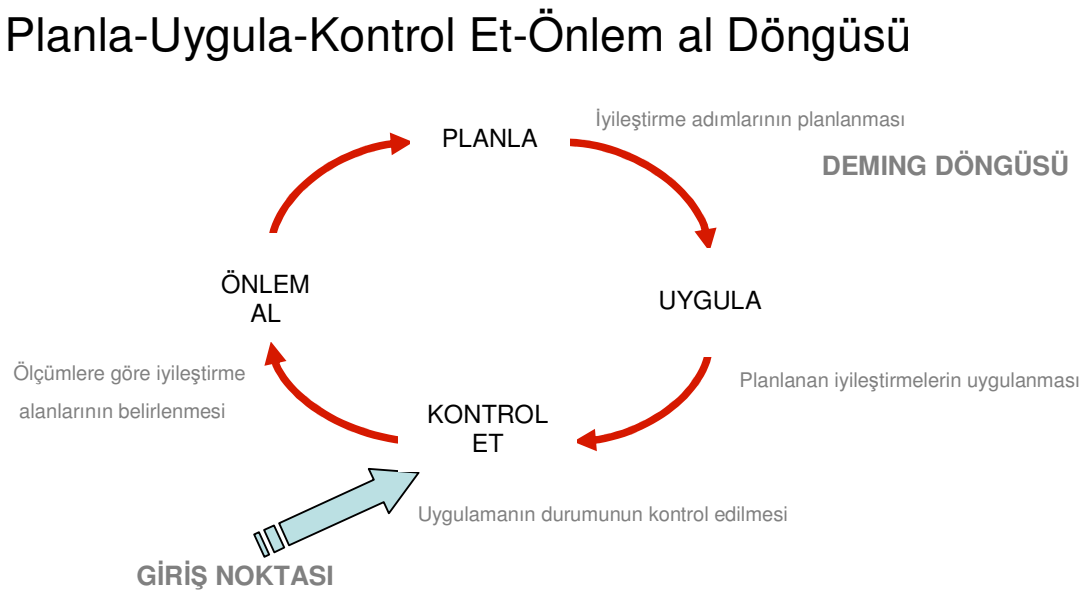
Yazılım ölçümü diğer model ve standartların bir parçasıdır. En çok sunulan örnekler olasılıkla CMM, CMMI ve SPICE gibi süreç değerlendirmesini ve olgunluk modellerini içermektedir.

### 3.4.2. İSK ve yetenek-olgunluk modelleri

1991 yılında Yazılım Mühendisliği Enstitüsü (Software Engineering Institute - SEI) tarafından yazılım geliştirme süreçlerinin iyileştirilmesinde adım adım bir yöntem

sağlamak için bir model kurulmuştur. Bu model firmanın nasıl kalite hedeflerine erişeceğini göstermeyi amaçlayan 5 olgunluk adımını tanımlamaktadır.

Süreç modelinin ana bileşenlerinden herhangi birinde girdiler ve çıktılar önemlidir. Standart süreç modelinde yer alan girdiler insan, prosedür, standart, yöntem ve çevre kategorileridir. Çıktılar ise ürünler, aksiyomlar (belitler) ya da daha az kavranılan şeyler olan deneyim ve bilgidir. İstatistiksel süreç kontrol teknikleri, girdi ve çıktıları ölçmek ve süreç iyileştirme aktivitelerine girdi sağlamak için kullanılır. Bu durum, Şekil 3.4.'de gösterilen Deming süreç iyileştirme döngüsünü ve istatistiksel kontrol tekniklerini gerekli kılmaktadır (Burr and Owen, 1996).




Şekil 3.4. Deming Döngüsü

Aksiyonun nedeni ne olursa olsun önemli olan sadece ürün ya da süreç iyileştirilmesi üzerine odaklanmak değil bunun tüm firma tarafından benimsenmesidir. Buradaki amaç süreçleri tanımlayarak firmanın olgunluğunu iyileştirmek ve süreçleri gittikçe artarak tekrar edilebilen yapmaktır. Burada firmanın olgunluğundan kastedilen firmanın süreçlerinin dokümante edilmesi ve belirlenen kalite standartları ile sistemin işlerliğinin sağlanmasıdır. Bundan sonraki adım sürecin her tekrar eden döngüsünün daha az sapmaya sahip olduğundan emin olmak için durumunu değiştirmektir. Şu anki sapmanın ne olduğunu bilmeden sapmayı azaltmak ve süreci kontrol altında tutmak için uygun duruma asla erişilemeyebilinir ve bundan dolayı da bir yarar elde edilemez. Bu durum iyileştirme fırsatlarının keşfedilmesine, kavranmasına ve

geliştirilmesine bir temel sağlar. Süreçlerin sınıflandırılabilmesi, süreç ölçüm dağılımlarının sınıflandırılmasının analizinden ve zaman içerisinde süreçler arasındaki ortak noktalar bulunabilmesinden dolayı da önemlidir. Bu durum, mevcut sistemde yer alan özel nedenler üzerinde çalışılmasını gerektirir.

Yetenek olgunluk modeli doğrultusunda gerçekleştirilen değerlendirme sonucunda, müşteriye daha az hatalı kodu içeren ürünün teslim edilmesi sağlanarak firmanın o an itibariyle nerede olduğu bilgisi elde edilebilir.

Yazılım sektöründe İSK tekniklerinin kullanılması, son zamanlarda firmaların CMM, CMMI ve SPICE gibi süreç iyileştirme modellerinin olgunluk seviyelerine yükselme arzularıyla artmaktadır. Bu modeller, daha yüksek olgunluk seviyelerine ulaşmaları için kritik bir adım olan İSK' yı uygulamaları için yazılım firmalarına yol göstermektedirler.

CMM, CMMI ve SPICE hem proje seviyesinde süreç kontrolü hem de organizasyonel seviyede süreç iyileştirilmesi amaçları için kontrol kartlarını önermektedirler (Sargut ve Demirörs, 2004).

### 3.4.3. Yetenek olgunluk modeli (Capability Maturity Model - CMM)

Olgunluk anketini (maturity questionnaire) kullanarak yazılım sürecini değerlendiren CMM, Amerikan Savunma Bakanlığı'nın 1970 - 1980'lerde yaşanan yazılım krizine çözüm bulması amacıyla Carnegie Mellon Üniversitesi'nden yardım istemesi üzerine, üniversite bünyesinde kurulan Yazılım Mühendisliği Enstitüsü (Software Engineering Institute-SEI) tarafından geliştirilmiştir. Genellikle büyük boyutlu firmalara hitap eden CMM' de, hem dışa karşı belgelendirme, hem de iç süreçlerin detaylı bir şekilde değerlendirilmesi söz konusudur.

CMM, 5 seviyeli bir modeldir (1–5 arası). Aşağıdaki tüm seviyeler tanımlanmıştır. Bu seviye tanımlarının sonunda yer alan parantezler içerisinde, her seviyeye ait anahtar süreç alanları verilmiştir. Bunlar:

1. Seviye: Başarının sadece bireylere bağlı olduğu "başlangıç" seviyesi (initial).

2. Seviye: Yazılı olmayan ve kısmen tutarlı süreçlerin olduğu “tekrarlanabilir” seviye (repeatable - gereksinim yönetimi, proje planlaması, proje takibi, taşeron yönetimi, kalite güvencesi, konfigürasyon yönetimi).

3. Seviye: Firma kültürünün yazılı hale geldiği “tanımlı” seviye (defined - süreç odaklaması, süreç tanımı, eğitim programı, bütünleşik yazılım yönetimi, yazılım ürün mühendisliği, gruplar arası koordinasyon, ayrıntılı değerlendirme)

4. Seviye: Tanımlı hale gelen süreçlerin artık ölçülebildiği, performans göstergelerinin değerlendirilebildiği “yönetilen” seviye (managed - nicel süreç yönetimi, yazılım kalite yönetimi)

5. Seviye: Kurumsallaşmanın gerçekleşip, geri beslemelerin sistematik bir şekilde değerlendirilmeye başlandığı “en iyileşen” seviye (optimizing - hata önleme teknoloji değişim yönetimi, süreç değişim yönetimi)

Yukarıda sadece isimleri geçen anahtar süreç alanları, ilgili seviyeye ulaşılması için gerçekleştirilmesi gereken süreçleri göstermektedir.

CMM uygulaması için hiyerarşik olarak, seviye belirleme, bir sonraki seviyeye geçmeden önce eksiklikleri belirleme, eksiklikleri hiyerarşik sıraya dizme, eksikliklerin giderilmesi için plan yapma, planı hayata geçirmek için kaynak ayırma, uygulama ve yöntem kapsamında, Deming döngüsü aşamaları uygulanır.

3.4.4. maddesinde tanımlanan ve 2001 senesinde CMM’ in belli seviye modellerinin entegrasyonu ile oluşan CMMI modelin kullanılmaya başlanması ile CMM güncelliğini yitirmiştir. Dolayısıyla burada CMM açısından İSK üzerinde durulmayacaktır.

### 3.4.4. Yetenek olgunluk modeli entegrasyonu (Capability Maturity Model Integration – CMMI)

Yetenek olgunluk modelleri, en genel anlamda firmaların insan kaynaklarını, süreçlerini ve teknolojilerini firmanın iş yapabilme performansını uzun vadeli geliştirecek şekilde olgunlaştırmasıdır.

Yetenek olgunluk modelleri fikri Amerikan Savunma Bakanlığı'nın (Department of Defence - DoD) isteği ve desteği üzerine, Yazılım Mühendisliği Enstitüsü (SEI) bünyesinde Carnegie Melon Üniversitesi tarafından yürütülmektedir. SEI 1991'den

günümüze kadar yazılım geliştirme, yazılım tedariki, sistem mühendisliği, ürün geliştirme ve insan konularında çeşitli olgunluk modelleri geliştirmiştir.

Sözü edilen modeller pek çok firma için faydalı olmakla birlikte, modellerin birbirlerinden farklı ve entegre olmamış olmaları uygulama aşamasında çeşitli sorunlar yaratmıştır.

Bu sorunu ortadan kaldırmak ve kaynak modelleri entegre etmek amacıyla SEI 1997 yılında Yetenek Olgunluk Modeli Entegrasyonu (Capability Maturity Model Integration - CMMI) adı altında bir girişim başlatarak; Yazılım Yetenek Olgunluk Modeli (SW-CMM), Elektronik Endüstrileri İşbirliği Ara Standardı (EIA/IS–731) ve Entegre Ürün Geliştirme Yetenek Olgunluk Modelini (IPD-CMM) birleştirerek, firma genelinde süreçlerin olgunlaştırılması için kullanılabilecek tek bir gelişim çatısı oluşturulmuştur (Atalay, 2003).

Yazılım sektöründe kullanılabilecek en temel yöntem olan CMMI, dünyadaki çeşitli firmalardan geniş kapsamlı katılım ile geliştirilmiştir. 1987 yılında ABD Savunma Bakanlığı'nın kurduğu SEI, bu alanda öncü kurumdur. SEI tarafından Aralık 2001'de CMMI 1.1 versiyonu basamaklı ve sürekli model olarak hazırlanarak yayınlanmıştır.

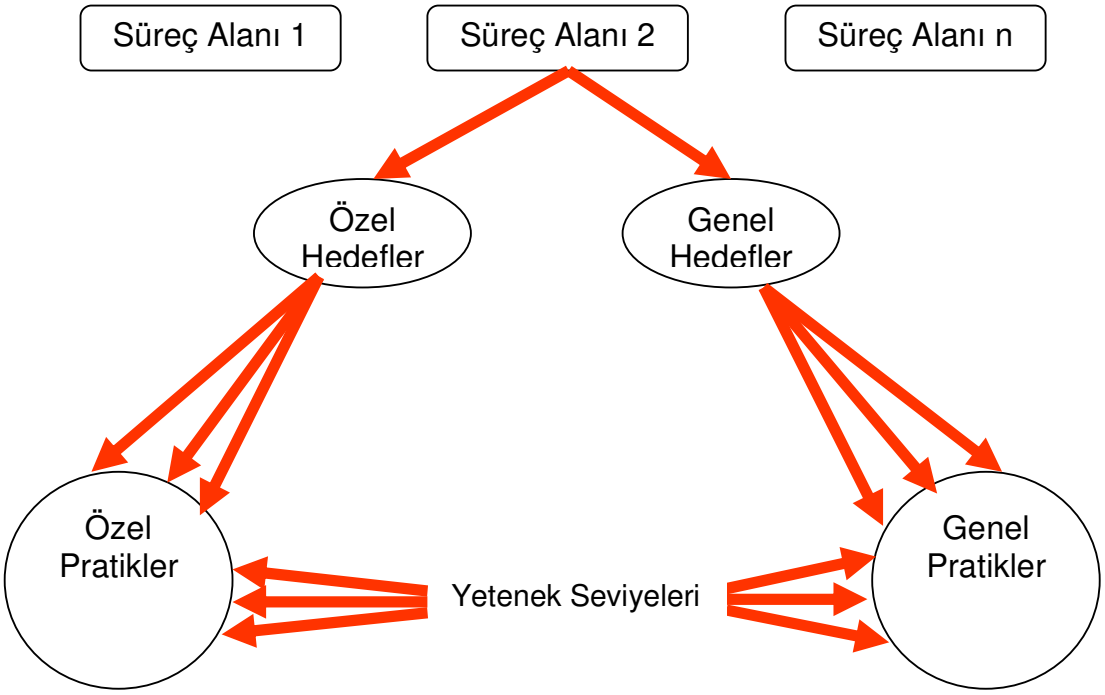
CMMI sürekli modeli, süreçler için altı yetenek seviyesi tanımlamaktadır (Çizelge 3.1.).

Çizelge 3.1. CMMI Yetenek Seviyeleri

| Seviye | Seviye Adı | Seviye Tanımı |
|---|---|---|
| 0 | Eksik | Süreç, ya yok, ya da öngörülen temel uygulamaları içermiyor. |
| 1 | Yapılan (var olan) | Süreç öngörülen temel uygulamaları büyük ölçüde içeriyor. |
| 2 | Yönetilen | Süreç, temel uygulamaların performansını ve ilgili iş ürünlerini düzenli bir şekilde yönetmeyi başarıyor. |
| 3 | Tanımlı (yerleşmiş) | Süreç, standart bir süreç tanımına göre ve uygun kaynakların atanması ile yürütülüyor. |
| 4 | Niceliksel olarak yönetilen (kestirilebilir) | Süreç, sayısal olarak ölçülüyor ve ölçüm verilerine göre kontrol ediliyor. |
| 5 | En iyileşen | Sürecin değiştirilme yöntemi tanımlanmış ve sürekli olarak iyileştiriliyor. |

Yoğunlukla, CMMI'ın 4. ve 5. yeterlilik seviyelerinde İSK kullanılır.

Birinci seviye dışında, her olgunluk seviyesi, süreç alanlarına bölünür. Süreç alanları 2'den 5'e seviyelerde gruplanır. Süreç alanları, hedeflere ulaşılmak (hedef yetenek seviyesi) için özel ve genel pratikleri içerir. Her bir genel ve özel pratik (generic and specific practice) belirli bir yetenek seviyesinde yer alır. Genel hedef ve genel pratikler tüm süreç alanlarında yer alırlar. Özel hedef ve pratikler ise sadece ilgili süreç alanında yer alırlar. Yetenek seviyesi 1 olan özel pratikler temel pratikler (Base Practices), 2 ve üstü olanlar ise ileri pratikler (Advanced Practices) olarak adlandırılır. (Şekil 3.5.) (http://www.sei.cmu.edu/cmmi/)




Şekil 3.5. CMMI Sürekli Model Gösterimi

CMMI süreç kategorileri Çizelge 3.2. ile verilmiştir. Her bir süreç kategorisi kendi içerisinde süreç alanlarından oluşmaktadır (Örn.; Destek süreç kategorisi içerisinde Konfigürasyon Yönetimi, Süreç ve Ürün Kalite Güvence, Ölçüm ve Analiz, Karar Analizi ve Çözümlemesi ve Neden Analizi ve Çözümlemesi süreç alanları yer almaktadır).

Çizelge 3.2. CMMI Süreç Kategorileri

| | |
|---|---|
| SÜREÇ YÖNETİMİ<br>(Process Management) | PROJE YÖNETİMİ<br>(Project Management) |
| MÜHENDİSLİK<br>(Engineering) | DESTEK<br>(Support) |

CMMI' daki hem sürekli hem de basamaklı model için geçerli olan ölçüm tabanlı genel pratikler ise Çizelge 3.3.'de yer almaktadır:

Çizelge 3.3. Ölçüm-Tabanlı Genel Pratikler (Goldenson et al., 2003)

| GENEL PRATİK | DURUM |
|---|---|
| 3. seviye: İyileştirme bilgilerini topla (GP 3.2) | Planlamadan ve sürecin gerçekleştirilmesinden elde edilen ürünleri, ölçümleri, ölçüm sonuçlarını ve iyileştirme bilgilerini, gelecekte kullanılmasını desteklemek ve firmanın süreçlerini ve süreç kaynaklarını iyileştirmek için topla. |
| 4. seviye: Süreç için ölçüm hedeflerini tanımla (GP 4.1) | Müşteri ihtiyaçları ve iş hedefleri üzerine dayalı süreç performansı ve kalite hakkında süreç için ölçülebilir hedefler tanımla ve kur. |
| 5. seviye: Sürekli süreç iyileştirmesinden emin ol (GP 5.1) | Firmanın iş hedefleriyle ilişkili olacak şekilde sürecin sürekli iyileştirildiğinden emin ol. |
| 5.seviye: Problemlerin kök nedeni düzelt (GP 5.2) | Hataların kök nedenlerini ve süreçteki diğer problemleri tanımla ve düzelt. |

Çizelge 3.3., yazılım projelerinde iyileştirme çalışmalarında İSK yöntemlerinin CMMI' daki zorunluluğunu göstermektedir.

CMMI' da, Organizasyonel Süreç Yönetimi ve Ölçülebilir Proje Yönetimi süreç alanları, projenin kurulmuş olan kalite ve süreç performans hedeflerine ulaşılmasını, projenin tanımlı sürecini niceliksel olarak yönetmeyi ve firmanın standart yazılım süreçleri için süreç performansının yönetilmesini sağlar. (Sargut ve Demirörs, 2005)

Ayrıca, CMMI' da "Ölçüm ve Analiz" adıyla bir kilit süreç alanı yer almaktadır. "Ölçüm ve Analiz" süreç alanı içerisinde yer alan belirli özelliklerin (specific practices) bir maddesinde uygun veri analiz yöntemlerinin ve araçlarının seçilmesi belirtilmiştir. Bunun için istatistiksel tekniklerin (dairesel çizim, histogramlar, çizgi grafikleri, dağılım

grafikleri, gruplandırma, neden-sonuç diyagramları, pareto diyagramı ya da kontrol tabloları) seçilmesi ve gerekliliği belirtilmiştir (Chrisis et al., 2004).

"Ölçülebilir Proje Yönetimi" süreç alanı içerisinde tanımlanan belirli özelliklerden birinde "ölçümlerin ve analitik tekniklerin" seçilmesini zorunlu kılmaktadır. Burada analitik tekniklerden kastedilen istatistiksel yöntemlerdir. Bir diğer maddesinde "her bir ölçüm özelliği için süreç performansının doğal sınırlarının hesaplanması" istenmektedir. Doğal sınırların hesaplaması için aşağıdaki yöntemler örnek olarak gösterilmektedir:

- Kontrol kartları,
- Güven aralıkları (dağılım parametreleri için),
- Tahmin aralıkları (gelecek çıktılar için).

Ayrıca, süreç sapmasına neden olabilecek özel nedenlerin analiz edilmesini ve normal olmayan oluşumların nedenlerinin tanımlanması gerekliliği CMMI'da belirtilmektedir. Bunun için kullanılacak tekniklere örnek olarak aşağıdakiler önerilmektedir:

- Neden – Sonuç Diyagramı,
- Tecrübeler,
- Kontrol kartları,
- Alt Gruplama.

"Organizasyonel Süreç Yönetimi" süreç alanı, süreç performansının ölçülmesinin nasıl kurulacağını tanımlamaktadır (Chrissis et al., 2004).

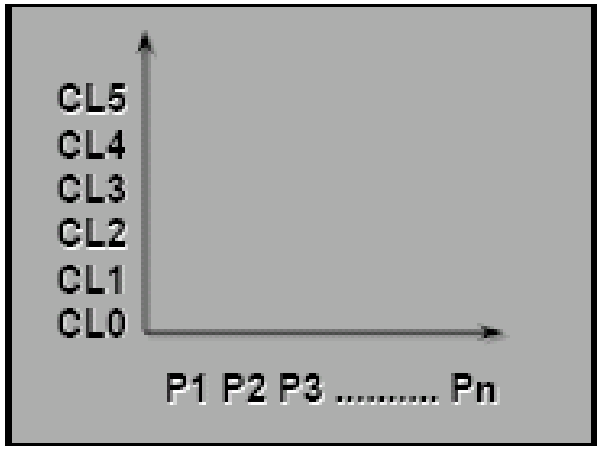
### 3.4.5. Yazılım süreç iyileştirme ve yetenek belirlemesi - ISO/IEC 15504 (Software Process Improvement and Capability dEtermination – SPICE)

SPICE' ın oluşumu, Uluslararası Standartlar Örgütü (International Organization for Standardization - ISO) ve Uluslararası Elektroteknik Komisyonu'nun (International Electrotechnical Commission - IEC), 1991'de ISO/IEC 15504 modeli için araştırma sürecini onaylamasıyla başlamıştır. ISO/IEC, 1993'te Yazılım Süreç İyileştirme ve

Yetenek Belirleme (Software Process Improvement and Capability dEtermination - SPICE) projesini başlatmıştır. ISO/IEC, 1998'de ISO/IEC TR 15504'ü yayımlamıştır. ISO/IEC 15504'ün, 2006 yılında ise bir uluslararası standart olarak basılması planlanmaktadır.

SPICE süreç değerlendirmesi, kendi kendine değerlendirme ile (firma içerisinden), grup tabanlı değerlendirme ile (firma içerisinden bir grup), sürekli değerlendirme ile (otomasyona girmiş bir veri toplama süreci ile) ya da bağımsız değerlendirme ile (firmadan bağımsız uzman tarafından) yapılabilir.

SPICE iki boyutlu bir modeldir (süreç kategorileri (P1, P2, … Pn) ve yetenek seviyeleri (CL0, … CL5) yer almaktadır). Şekil 3.6., iki boyutlu SPICE modelini göstermektedir.



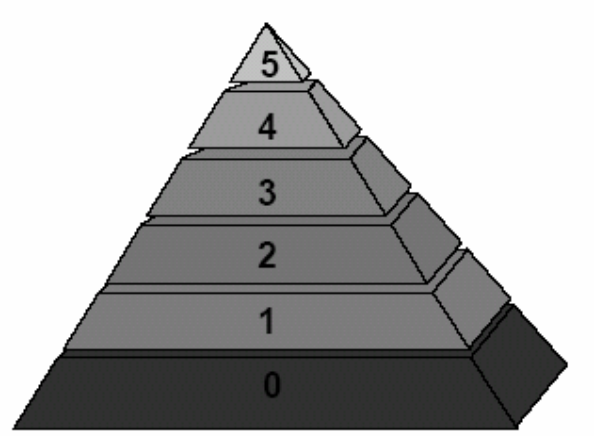

Şekil 3.6. SPICE İki Boyutlu Modeli

SPICE' a göre, birinci boyutu (süreç kategorileri) oluşturan süreç kategorileri 5 sınıfa ayrılır

1. Müşteri-satıcıya direkt etkisi olan süreçler (CUS)
2. Mühendislik süreçleri (ENG)
3. Projeyi oluşturan ve yöneten süreçler (yönetim) (MAN)
4. Destek süreçleri (SUP)
5. Organizasyon süreçleri (ORG)

Her bir süreç kategorisi, kendi içerisinde de alt süreçlerden oluşmaktadır. (P1, P2, P3, … Pn):

İkinci boyut olan yetenek boyutunda ise 6 yetenek seviyesi (0 - 5 arası) Şekil 3.7.'de gösterilmiştir:




Şekil 3.7. SPICE Yetenek Seviyeleri

0. Seviye: Başarının sadece bireylere bağlı olduğu "eksik" seviye (incomplete).

1. Seviye: Planlama yapılmadan, süreçlerin genel olarak yerine getirildiği "var olan" seviye (performed).

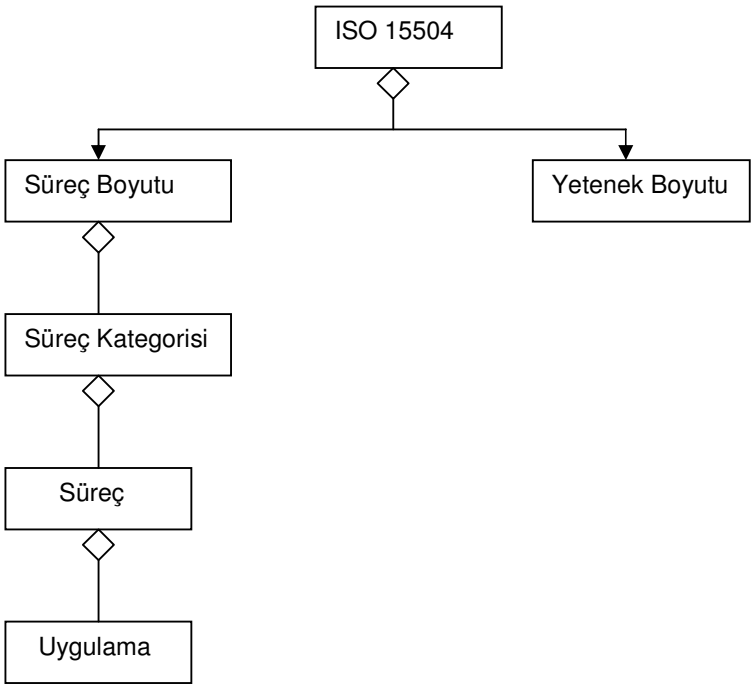
2. Seviye: Süreçlerin tanımlandığı, iş planının uygulandığı "yönetilen" seviye (managed).

3. Seviye: Organizasyonda belgelendirilmiş standart süreçler ve uygulamaların olduğu "yerleşmiş" seviye (established).

4. Seviye: Denetim altındaki süreçte detaylı performans ölçümlerinin toplanabildiği "kestirilebilir" seviye (predictable).

5. Seviye: Geri beslemelerle sürekli iyileştirilen "en iyilenen" seviye (optimizing).

SPICE' ın yüksek seviyede gösterimi Şekil 3.8.'de gösterilmiştir:




Şekil 3.8. SPICE' ın Yüksek Seviye Gösterimi

Yetenek seviyeleri, 2., 3., 4., ve 5. seviye için 2 tane, 1. seviye bir tane süreç özelliğinden (process attribute) oluşmaktadır. Her süreç özelliği, bir sürecin ölçülebilir karakteristiğini sunmaktadır. Seviyelere göre süreç özellikleri Çizelge 3.4'de verilmiştir.

Çizelge 3.4. Seviyelere göre süreç özellikleri

| Seviye | Süreç Özelliği |
|---|---|
| Seviye 1 | 1.1 Süreç Performansı |
| Seviye 2 | 2.1 Performans Yönetimi<br>2.2 Ürün Yönetimi |
| Seviye 3 | 3.1 Süreç Tanımı<br>3.2 Süreç Kaynağı (Olanağı) |
| Seviye 4 | 4.1 Süreç Ölçümü<br>4.2 Süreç Kontrol |
| Seviye 5 | 5.1 Süreç Değişikliği<br>5.2 Devamlı İyileştirme |

1. Seviyede incelenen süreç alanları için temel pratiklerin yerine getirilmesi ve ürünlerin özellikleri ile birlikte beklenen maddelerin sağlanması beklenir. 2. ve daha yüksek seviyelerde ise Çizelge 3.4.'de yer alan süreç özelliklerini sağlaması gerekmektedir.

SPICE modelinde ölçüm, 3. Seviyede, süreç performans verisinin toplanması ile başlamaktadır. Bu seviyede süreç performans verisi kullanılarak süreç davranışının anlaşılması hedeflenmektedir.

SPICE' da, ölçüm özelliği, bir yazılım firmasında süreç performansındaki eğilimleri analiz etmeye ve firmanın karşısında yer alan tanımlı limitler içerisinde süreç yeteneğini sürdürmeye ihtiyaç duymaktadır. Aynı şekilde, "Süreç Kontrol" olarak adlandırılan 4. Seviye, süreç özelliği, iyileştirmenin uygulanması ve kontrol altında tutulmasını sürdürmek için süreç performansının kontrol edilmesine ihtiyaç duymaktadır.

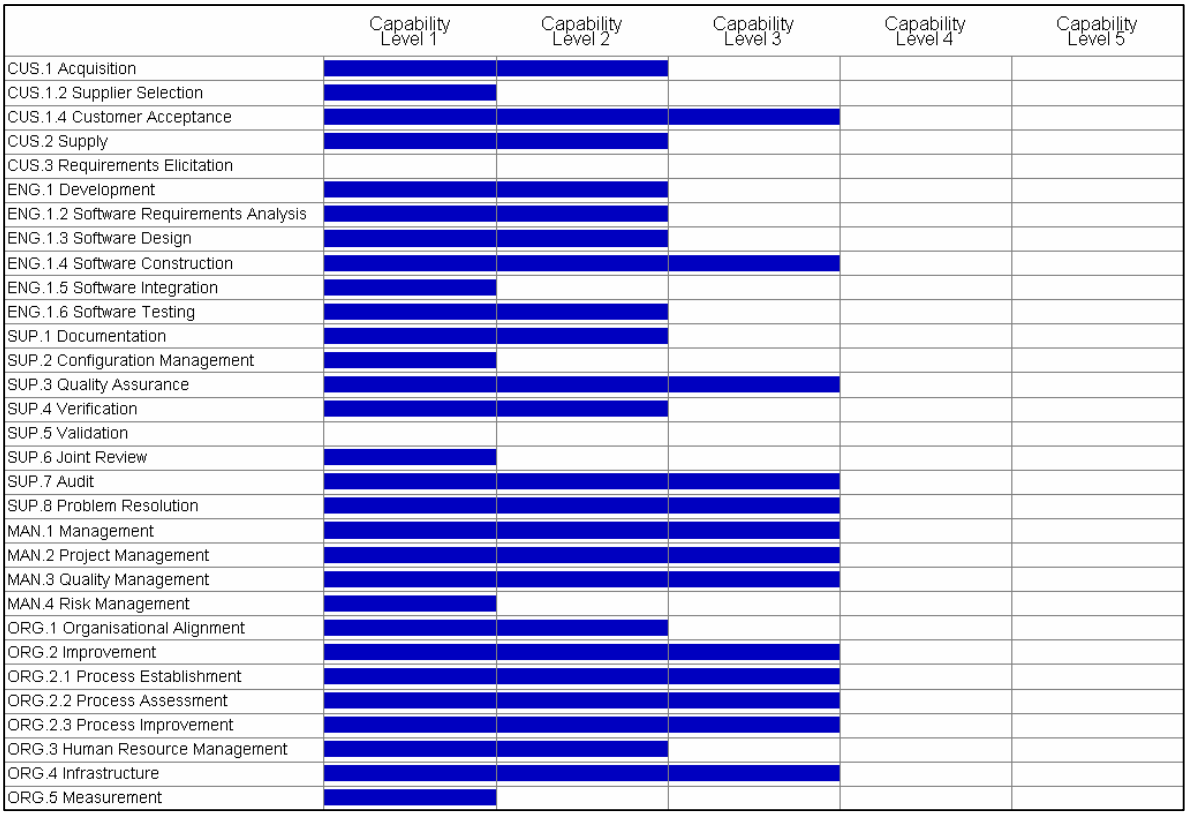
4. Seviyede, ilgili iş hedeflerine ulaşılmasını destekleyen ürün ve süreç hedefleri ve ölçümler tanımlanır ve belirli ürün ve süreç ölçümleri toplanır. Sürecin performans

eğilimleri analiz edilir. Analiz için gerekli uygun ölçüm teknikleri tanımlanır. Bu noktada istatistiksel analiz tekniklerinden pareto analizi, istatistiksel kontrol kartları gibi istatistiksel süreç kontrol yöntemlerinden uygun olanının seçilmesi ve kullanılması beklenmektedir. Analizi gerçekleştirmek için ölçümler toplanır ve süreç kontrol parametreleri tanımlanır. Süreç performansını kontrol etmek için kullanılan analiz ölçümleri, aksiyonları kontrol etmek ya da iyileştirmek için tanımlanır.

5. Seviye, sürecin ölçülebilir temeli üzerine dayanarak standart süreç tanımı için değişiklikleri tanımlar. Gerçek ve potansiyel problemlerin kaynağı analiz edilerek sürekli iyileşen süreç tanımlanır (ISO/IEC TR 15504: 1998).

Bu modeller yazılım firmalarını İSK tekniklerini uygulamaları için yönlendirmişlerdir. Yüksek olgunluk seviyesinde (4. seviye ve üstü) İSK' nın ilgili gereksinimlerinin varlığı ve İSK' nın 4. seviye'ye ulaşıldıktan sonra uygulanabileceği genel inancı, genellikle İSK tekniklerini daha önce uygulamaları için firmalara engel olmuştur. İSK uygulamasının başarılı olabilmesi için kritik nokta süreç durağanlığı, ölçüm yeteneği ve veri doğruluğudur. Diğer bir deyişle, eğer süreç sürekli olarak uygulanıyorsa, doğru ölçümler seçildiyse ve güvenilir veri toplama mekanizması kurulduysa İSK tekniklerinin uygulanmasından fayda elde edilebilir.

Firmanın nerede olduğunu görmesi ve bu doğrultuda iyileştirme çalışmaları başlatabilmesi açısından SPICE modeli kullanılabilir. Örnek olarak, SPICE modeli kullanılarak SPICE 1-2-1 paket programından elde edilen değerlendirme sonuçlarını içeren grafik Şekil 3.9'da gösterilmiştir. Aşağıdaki veriler firmanın gizlilik ilkesi benimsenerek kullanılmıştır.




Şekil 3.9. SPICE Değerlendirme Sonuç Tablosu

Yukarıdaki Şekil 3.9. incelendiğinde firmanın hedeflediği seviye için hangi süreçlerini iyileştirmesi gerektiği görülmektedir. Bu doğrultuda süreç iyileştirme çalışmaları başlatılabilir.

## 4. YAZILIMDA İSK'NİN KULLANILMASI

Yazılım geliştirme sektöründeki dil ve geliştirme yöntemlerindeki çeşitlilikten dolayı herhangi bir firma için konular hakkında genelleme yapmak zordur. Deming'in "herhangi bir süreçteki sapmayı anlamayla başlayabilirsek, gereksiz ya da özel sapma nedenlerini eleyerek süreci nasıl kontrol edeceğimizi anlamaya başlayabiliriz" ifadesi yazılımda İSK' nın kullanılmasında da geçerlidir. Bu ifade, yazılımda ürünün yayımlanması sonrasında hata sayısını minimuma indirmek için kullanılan bir mekanizma ile sürecin değişkenliğini azaltmak biçiminde yorumlanabilir. Geliştirme maliyetinin azaltılması, süreç tahmin maliyetinin iyileştirilmesi ve proje karlılığının arttırılması yazılımda önemli olup bunları gerçekleştirmek için İSK kullanılır.

Yazılım geliştirme süreci boyunca belli ölçümler, maliyet, zaman ve bulunan hata sayısı değişir. Bununla birlikte, bunlar süreci kontrol etmek için yeterli bilgiyi sağlamazlar, dolayısıyla amaç geliştirme döngüsünün her kısmında değişkenlik gösteren özellikleri bulmak olmalıdır.

Üst seviye yöneticiler istatistiksel yöntemleri kullanarak yazılım sürecini nicelendirmek ve kontrol altına almayı isterler. Bu durumda, İSK kaçınılmazdır ve rasgele sapmalar sonucu başlatılan düzeltici faaliyet sapmalarını izole eden bir araç sağlamak için yazılıma uygulanır (Eickelmann ve Anant, 2003).

Yazılım geliştirmede İSK' nın kullanılmasında pek çok araştırmacı çalışmıştır. Bunların bir kısmı süreç iyileştirme modelleri üzerine ve yüksek süreç olgunluğuna ulaşmak için İSK üzerine odaklanmışlardır (Burr, Owen, Florac, Carleton, Barnard, Jakolte, Saxena, Humphrey). Bu çalışmalardan biri, daha yüksek olgunluk seviyelerine erişmek için adımları ve bir yazılım firmasında süreçlerin iyileştirilmesi için basit bir rehber olan hareketleri belirleyen yazılım süreç yönetimi için bir iskelet (framework) tanımlayan Humphrey'e aittir (1989). Çalışmalarında, 4. seviye firmalar için veri analiz tekniğinin bir anlamı olarak İSK' yı vurgulamaktadır.

Florac ve Carleton (1999), yazılım süreç iyileştirilmesi için CMM kapsamında İSK' nın kullanımını açıklamışlardır. Florac ve Carleton, Shewhart'ın prensiplerine dayanarak, süreç performansının kontrolü ve değerlendirilmesi için prensipler ve yöntemler üzerine odaklanmışlardır. Hatta yazılım geliştirmede kontrol kartlarının

uygulanmasıyla ilişkili konuları ve üretim sektöründen kazanılan deneyimlerin yazılım süreçlerine dâhil edilmesini tartışmışlardır.

İSK, 2001 Yüksek Olgunluk seminerlerinde önemli bir konu olmuştur. Paulk ve Chrissis (2001), 35 yüksek olgunluk seviyesinde yazılım firmaları (4. ve 5. seviye) için tanımlanan CMM uygulamalarının anlaşılması için gerçekleştirilen seminerlerde yer almışlardır. Bu CMM-tabanlı çalışma, 3. seviye ölçümlerinin İSK' nın uygulanması için yeterli olmadığını göstermiştir.

Bazı kaynaklar, genel İSK prensiplerini, özellikle yazılımın geliştirilmesinde bir kılavuz olarak kullanmaktadırlar. Burr ve Owen (1996), gereksinim, tasarım, gerçekleştirim ve bakım aşamaları boyunca yazılımın kalitesini kontrol etmek ve yönetmek için mevcut uygulanmakta olan istatistiksel tekniklerin tanımlanmasında rehberlik etmişlerdir. Faydalı İSK araçlarından olan kontrol kartları üzerine odaklanarak yazılım alanında ölçüm, süreç iyileştirme ve süreç yönetimi için yol gösterici rol oynamışlardır.

Lantzy (1992), İSK' nın yazılım süreçlerine uygulanmasının tartışılması üzerine ilk çalışmalardan birini sunmuştur. Bir yazılım firmasında başarılı İSK uygulaması için bir kılavuzu özetlemektedir. Bunlar:

- Ölçümler, müşteri tarafından tanımlanan ürünün kalite karakteristikleriyle ilişkili olmalı,
- Ölçümler, anlaşılır maddelerin üretildiği aktiviteler için seçilmeli,
- İSK sadece kritik süreçler için uygulanmalı,

Kan (1985), yazılım geliştirmesinde ölçümlerin uygulanması ve teori hakkında kapsamlı bir çalışma sunmuştur. Yazılım alanında süreç olgunluğuna erişmenin zorluklarını vurgulamıştır. Yazılım firmalarına yardımcı olabilecek kontrol kartlarının kullanımı üzerinde durmuştur. Bununla birlikte, üretimdeki son ürün kalitesinin sunulması gibi bir kontrol sağlanması yerine genellikle süreç içerisindeki ölçümlerin kullanılması üzerinde çalışmıştır.

Son-ürün kalitesi, projenin sonunda ölçülebilir, dolayısıyla süreçler üzerinde zamanında kontrol zorlaşır. Kan (1985) aynı çalışmasında, yazılım geliştirmede süreç durgunluğuna erişmek için süreç olgunluğu gerekliliğinin altını çizer. Son olarak,

projenin süreç içerisindeki hedeflerini karşıladığında ve son-ürün kalite hedeflerine ulaştığında süreçlerin kontrol altında tutulabileceğini göstermiştir.

Radice (1998), İSK tekniklerinin yazılım alanında kısıtlandığını tanımlamış ve uygulamalı örneklerle kendi teorik bilgisini destekleyerek ayrıntılı bir makale vermiştir. Tüm İSK tekniklerinin yazılım süreçleri için uygulanamayabileceğini ve u-kart tekniğinin olası bir teknik olabileceğini belirtmiştir.

Tarihçesine bakıldığında, yazılım alanında İSK' nın uygulamalı deneyimlerini sunan çalışmalar da görülmüştür. Weller (2000), her bir incelemede hata yoğunluk verisi için u-kartı çizmiştir.

Florac, Carleton ve Barnard (2000), SEI ve Space Shuttle Onboard Projeleri arasında ortak bir çabadan elde edilen analizlerini sunarak ilk olarak anahtar süreçlerin seçilmesinin önemini, operasyonel tanımların sağlanmasını ve rasyonel alt gruplama sorunun adreslenmesini, doğru kontrol kartların kullanılmasını vurgulamıştır. Daha sonra bu faktörlerin her birini uygulayarak paket gözden geçirme verisi üzerine İSK' yı uygulamıştır. Son olarak yazılım süreçlerine uygulanan İSK' nın faydalarını özetlemiştir.

İSK' nın yazılım geliştirmede kullanılabilirliği "Can Statistical Process Control be Usefully Applied to Software" isimli Avrupa Yazılım Mühendisliği Süreç Grubu (European SEPG) konferansında 1999 panelinde de tartışılmıştır. Panelde, Keller (1999), üst yönetim kararlarının alınmasında ve tahmin edilmesi süreçlerinde İSK' nın önemi üzerinde durmuştur. Belirli bir İSK uygulaması için gözlemlerini ve sonuçlarını da belirtmiştir. Bernard ve Carleton (1999), "Space Shuttle Onboard Project" üzerindeki deneyimlerine başvurarak karmaşık süreç davranışlarını vurgulamışlardır. Hirsh (1999), İSK kartlarının, arzu edilen faydaları toplamak için kullanılması gerektiğini ve üst yöneticilerin İSK' nın kullanılmasını sağlamak için eğitilmeleri gerektiğini vurgulamıştır. Meade (1999), Lockheed Martin Company'de 4. seviye uygulamasının parçası olan İSK kullanımından bir özet sunmuştur. Verinin anlaşılmasının önemini belirterek tüm yazılım ölçümlerine İSK uygulamasının mümkün olmayacağını açığa çıkartmıştır (Sargut ve Demirörs, 2005).

### 4.1. Yazılım Geliştirmedeki İSK' nın Zorlukları ve Faydaları

Basit kavramlar, istatistiksel süreç kontrolün temelini oluşturmasına rağmen, İSK' nın uygulanması özellikle yazılım sektöründe zordur. Üretim sektöründe çok sayıda aynı ürünün imal edilmesi, ölçülen özellikler için çok sayıda örneklem elde edilmesini sağlar. Üretimde, ürün belli ve dolayısıyla ölçülebilen özellikler ve değişkenler kolaylıkla tanımlanmaktadır. Diğer taraftan, yazılım ürününü tanımlamak zordur. Yazılım sektöründe, İSK' nın uygulanmasını zorlaştıran problemler ise aşağıda yer almaktadır:

- Tanımlanmamış süreç: Özellikle istatistiksel kontrol uygulanacak süreçler iyi hazırlanmış süreç tanımlarına sahip olmalıdır. Eğer yazılım süreci dokümante edilmez ve belirlenen kurallara (standartlara) uygun olarak geliştirilmezse ölçülebilir bir performans sonucu elde etmeyi beklemek yanlış olur.

- Tekrarlanabilir bir üretimin gerçekleştirilmemesi: Her ürün birbirinden ayrı ve farklı özelliklere sahiptir. Bu doğrultuda analiz gerçekleştirilmelidir.

- Ölçüm seçiminin eksikliği: Tüm süreçlerin istatistiksel süreç kontrole ihtiyaç duymadığı unutulmamalıdır. Firmanın hedefleri ve İSK sonucunda beklentileri doğrultusunda ölçümler seçilmelidir. Detaylı bir grup ölçümü tanımlamak doğru değildir. Bu durumda, süreç ya da ürünün performansına ışık tutacak bir ya da iki ölçümün seçilmesi yerinde olacaktır. Ancak, bu ölçümlerin seçilmesinde maliyet de düşünülmelidir.

- Bireysel ya da küçük olaylara odaklanma. Kontrol kart ya da dağılımlar, bireysel uygunsuzlukları tanımlamak yerine süreç eğilimlerinin tespit edilmesine yardımcı olacak şekilde oluşturulmalıdır.

- Sorunun araştırmasında yapılan hatalar. İSK sadece bir problemin var olduğu bilgisini verir. Denetim gibi detaylı bir araştırma ile problem incelenmezse İSK' nın kullanımından bir fayda elde edilemez.

- Eğitimin eksikliği. Pek çok problem etkili eğitimler ile düzeltilebilir. Buradaki eğitim, projedeki diğer disiplinler üzerine değil yazılım mühendisliği üzerine dayanmalıdır.

Yazılım sektöründe, İSK' nın uygulanmasının avantajları ise: İlk olarak, hatalar daha erken tespit edilebilir ya da önlenebilir. İSK ile yazılım geliştirme süreci izlenerek hataların nedenine ilave hatalar yaratılmadan önce tespit edilebilir. İkinci olarak, İSK tekniklerinin kullanılması masrafsızdır çünkü sürecin tüm çıktıları üzerinde detaylı kontrolleri yapmak için ihtiyaç duyulandan daha az çaba ile sürecin düzenli olarak işletildiğinden emin olunur. Dolayısıyla, daha yüksek kaliteye daha düşük geliştirme maliyetiyle erişilebilir. Son olarak, İSK tekniklerinin kullanılması, sürecin ve problemlerin sayısal olarak ölçülmesini sağlar. Böylece daha az sapma ile tahmin (kestirim) gerçekleştirilir.

Sonuçta, İSK' nın kullanılması pratik ve yazılım geliştirmesine uygun olmayabilir fakat yazılım süreçlerinin anlaşılırlığını ve uygulanabilirliğini arttırır (Gupta, 2002).

# 5. UYGULAMA

Uygulama, yurt içi ve yurt dışı orta ve büyük ölçekli yazılım ve sistem entegrasyon projeleri geliştiren, anahtar teslimi yani proje bazlı gereksinimleri belirleyip ürünü tamamlayıp tespit eden bir firmadan alınan veriler ile gerçekleştirilmiştir.

Yazılım ölçümü 3 kategoriye ayrılmaktadır,

- Yazılım için harcanan çaba ve planlamanın ölçülmesi (Software Effort and Schedule Measurement). Çalışanın harcadığı sürenin hesaplanması ve planlama bilgisinin raporlanması
- Yazılım kalite ölçümleri (Software Quality Measurement). Problemlerin ve hataların sayılması
- Yazılım büyüklük ölçümü (Software Size Measurement). Kaynak durumunun sayılması

Birinci ve üçüncü maddelere ilişkin veri, firma tarafından ölçülmediğinden, uygulama, yazılım kalite ölçümleri üzerine gerçekleştirilmiştir.

Veriler, firma tarafından yazılımı gerçekleştirilmiş bir otomasyon aracından alınmıştır. Bu otomasyon aracı, firmanın belirlediği kriterler altında, projeler hakkında bilgilerin tutulduğu bir yazılımdır. Veriler, belli periyotlar dâhilinde toplamda son 1,5 yılı içerecek şekilde 30 proje üzerinden alınmıştır. Bu projelerde tespit edilen hata sayıları, hata tipleri, bu hataların öncelikli olarak düzeltilme sıraları ve testten dönen hata sayıları bilgileri kullanılarak İSK tekniklerinden gruplandırma, histogram, pareto diyagramı ve u kontrol kartı uygulanmıştır.

Firmanın kullandığı otomasyon aracı içerisinde projelerde üretilen dokümanlara ilişkin bilgi tutulmadığından sadece kodlama, test ve bakım aşamalarını içerecek şekilde toplu bilgiler alınmıştır. Tespit edilen hataların giderilmesi, gözden geçirme aktiviteleri ve denetim için harcanan süre bilgileri tutulmadığından performansa yönelik bir iyileştirme çalışması için uygulama gerçekleştirilememiştir.

Uygulamada, projelere ilişkin hata tipleri 3 grupta incelenmiştir:

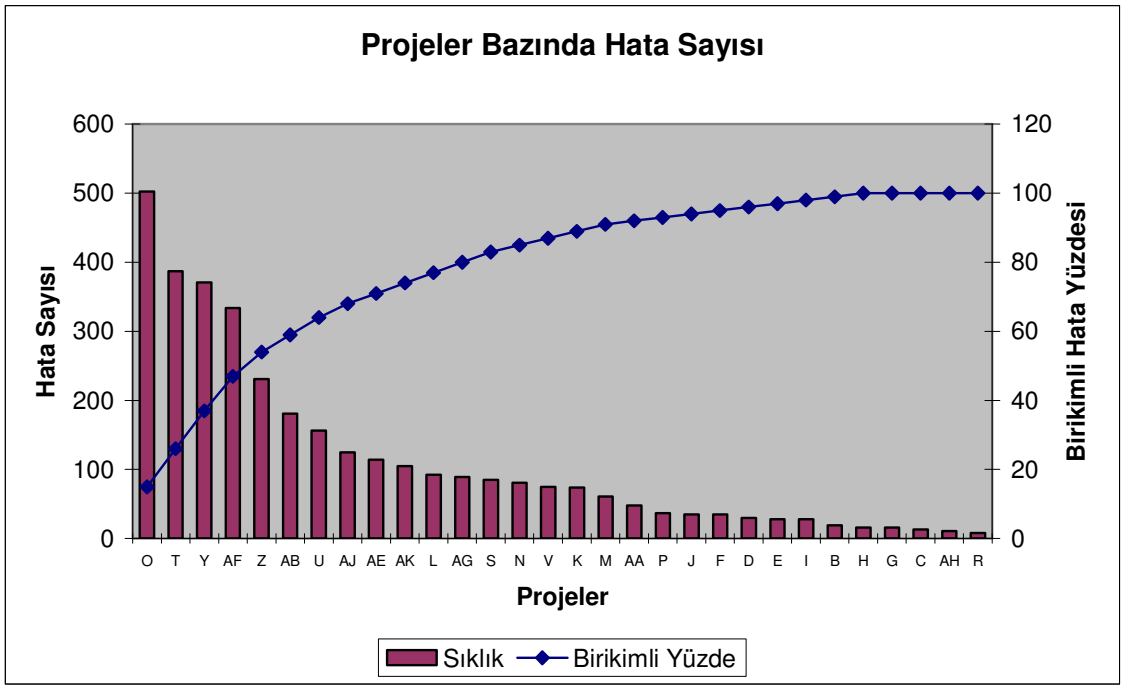
1. Yazılım geliştirici tarafından tespit edilen, kodlama ve veritabanı yapısıyla ilgili hataların bildirimi için kullanılan hata tipi (I nolu hata),

2. Kullanıcı tarafından tespit edilen, kodlama ve veritabanı yapısıyla ilgili hataların bildirimi için kullanılan hata tipi (II nolu hata),

3. Kullanıcı tarafından tespit edilen, tek seferlik oluşan, daha çok verileri ilgilendiren özel hataların bildirimi için kullanılan hata tipi (III nolu hata),

Uygulamada, İSK teknikleri kullanılmadan önce, projeler A harfinden başlayarak B, C, D, ... şeklinde devam eden harfler ile adlandırılmıştır.

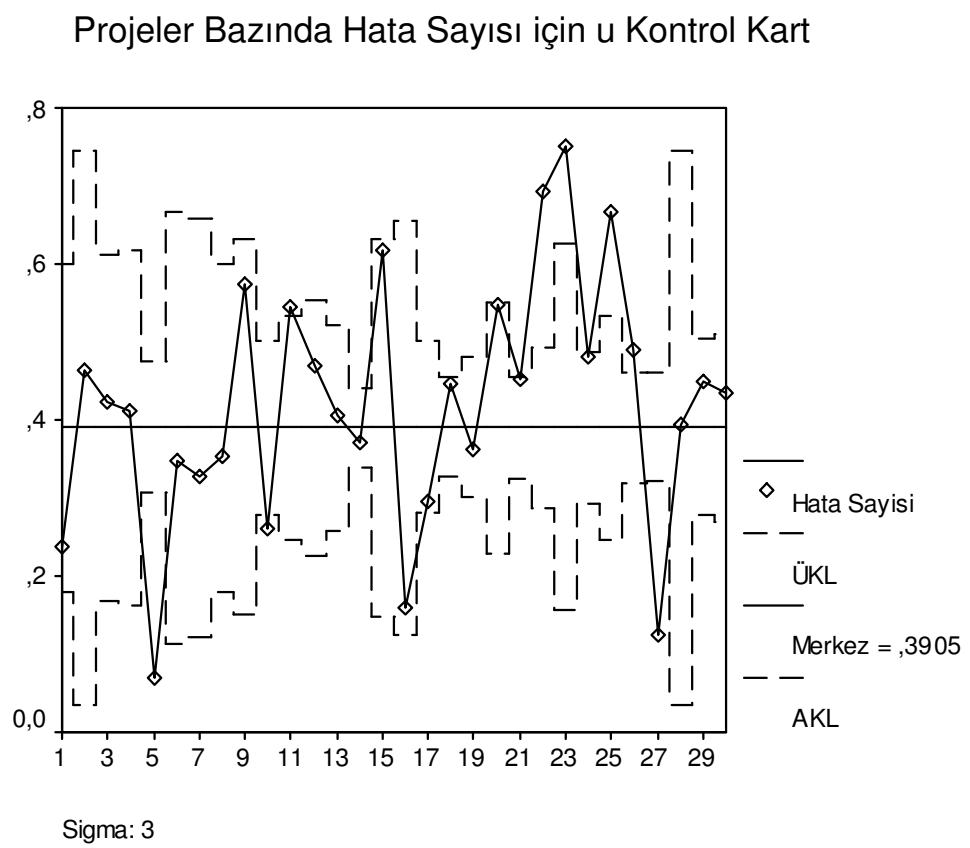
Firma tarafından gerçekleştirilen projelerde tespit edilen hata sayıları için pareto diyagramı çizilerek projeler arasındaki hata sayıları Şekil 5.1.'de incelenmiştir.




Şekil 5.1 Tüm projeler bazında hata sayısı pareto diyagramı

Pareto diyagramı incelendiğinde O, T, Y, AF, Z, AB, U, AJ, AE, AK, L ve AG projelerindeki hata sayısı oranlarının, firmadaki hata oranı sayısının yüzde seksenini oluşturduğu görülmektedir. Bu projelerdeki hata sayılarının neden bu kadar yüksek çıktığı bilgisi detaylı olarak firma tarafından irdelenerek hata sayısının azaltılması yoluna gidilmelidir.

Doğal olarak, projelerin büyüklükleri de çok önemlidir. Dolayısıyla hata sayısı yüksek çıkan projeleri sapma olarak değerlendirmek yanlış olur. Bu durumda, hata sayısını incelemek ve sapmaları belirlemek amacıyla u kontrol kart çizilmiştir. Kontrol kart çiziminde ±3σ kontrol limitleri kullanılmıştır. Projelerin büyüklükleri ise adam/saat olarak alınmıştır.

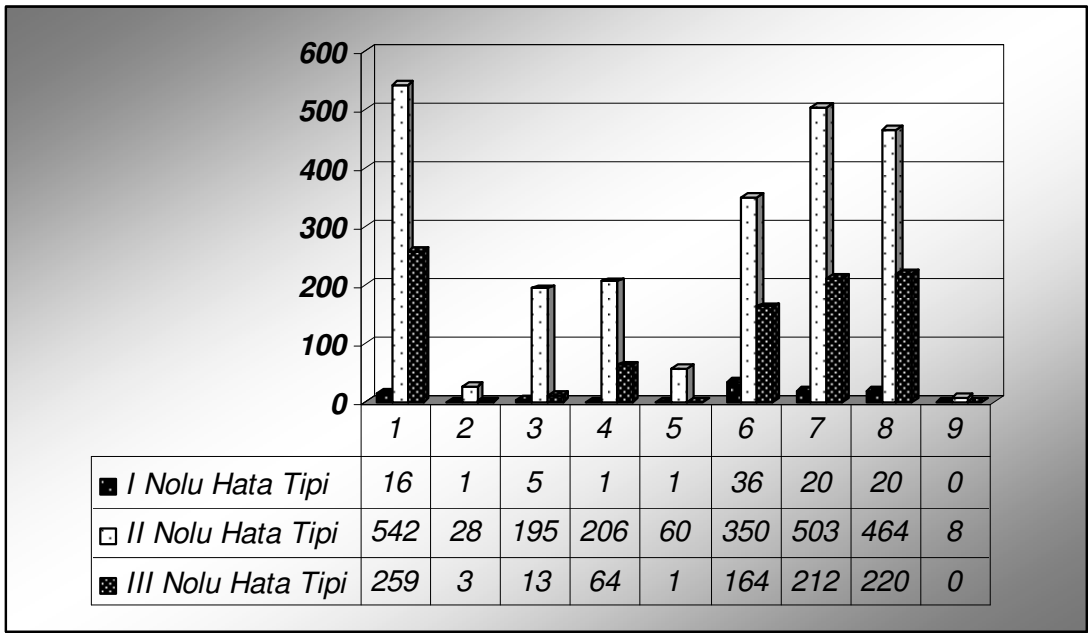



Şekil 5.2 Projelerdeki hata sayısı u kontrol kart grafiği

Kontrol kart incelendiğinde, 5., 10., 11., 22., 23., 25. 26. ve 27. yani F, K, L, Z, AA, AE, AF ve AG projelerinin kontrol limitleri dışında olduğu gözlenmiştir. Bu projelerde tespit edilen hata sayılarının süreçte sapmaya neden olduklarını ve istatistiksel olarak kontrol altında olmadıklarını söyleyebiliriz. Bu durumda, firma, süreçlerini bu projeler doğrultusunda tekrar gözden geçirmeli ve iyileştirme çalışması başlatmalıdır. Süreç iyileştirme çalışmaları sonucunda tekrar ölçümler alınarak başarıya ulaşılıp ulaşılmadığı takip edilmelidir.

Firma tarafından, uygunsuzlukların sayısı dışında bu uygunsuzlukların giderilmesi için önceliklerini belirleyen sınıflandırma da toplanmıştır. Bu öncelikler 1'den

başlayarak önem sırasına göre 2, 3, 4, … şeklinde sıralandığında, firma genelinde tespit edilen hata sayılarının öncelikleri için Şekil 5.3. histogramı elde edilmiştir.

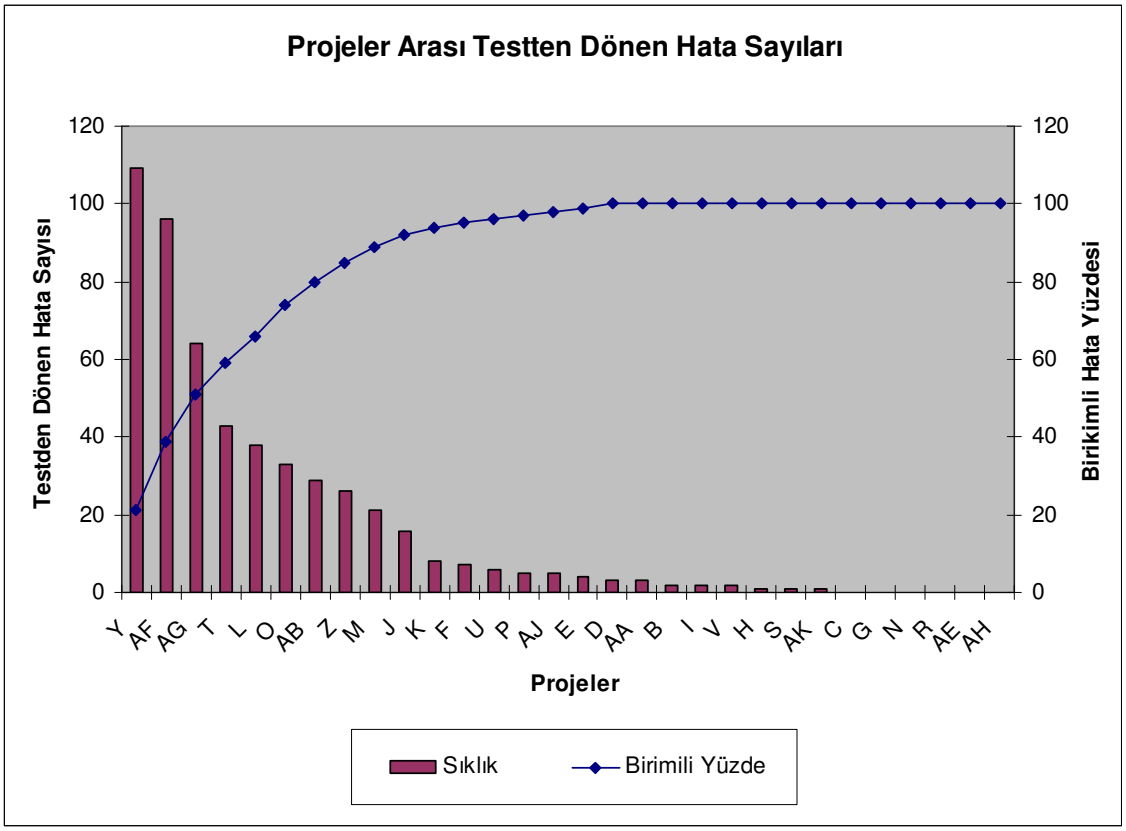



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| I Nolu Hata Tipi | 16 | 1 | 5 | 1 | 1 | 36 | 20 | 20 | 0 |
| II Nolu Hata Tipi | 542 | 28 | 195 | 206 | 60 | 350 | 503 | 464 | 8 |
| III Nolu Hata Tipi | 259 | 3 | 13 | 64 | 1 | 164 | 212 | 220 | 0 |

Şekil 5.3. Firma genelinde tespit edilen hata sayılarının giderilme önceliklerini gösteren histogram

Bu histogram, hataların giderilme önceliğinin "düşük" (6, 7 ve 8. önem sıraları) olduğunu gösterse de grafiğin en solunda yer alan 1 nolu öncelik değerinin diğerlerine göre en yüksek olduğu görülmektedir. Bu da, projenin takviminde sapmaya neden olabilecek kritik hataların sayısının oldukça yüksek olduğu anlamına gelmektedir. Bu değerin bu kadar yüksek çıkmasında II nolu hata tipinin etkinliği de göz önüne alınırsa, kullanıcı tarafından yüksek öncelikli hataların tespit edilmesi, firmanın başarısızlığını göstermektedir. Bu durumda, tespit edilen yüksek öncelikli hatalar incelenmeli, kaynağı araştırılmalı ve bu noktalarda iyileştirme çalışmaları başlatılmalıdır.

Firmadan alınan bir diğer çeşit veri tipi ise projelerde, testten dönen hata sayılarıdır. Bu veri kullanılarak projeler arası testten dönen hata sayılarını incelemek için pareto diyagramı çizilirse Şekil 5.4. elde edilir.




Şekil 5.4. Projeler arası testten dönen hata sayısı pareto diyagramı

Şekil 5.4'de Y, AF, AG ve T projelerinde test sırasında tespit edilip yazılımcıya gönderilen hatalı sayısının yüksek olduğu görülmektedir. Firma, testten dönen hata sayılarının yüksek çıktığı projeleri incelemelidir. Bu projelerde uygulanan test tiplerini ve en çok hangi aşamada uygunsuzlukların döndüğünü tespit etmeli ve bu doğrultuda iyileştirme çalışmaları başlatmalıdır. Eğer test türlerine ilişkin bilgiye erişemiyorsa tüm test sürecini gözden geçirmesi ve iyileştirmesi önerilmektedir.

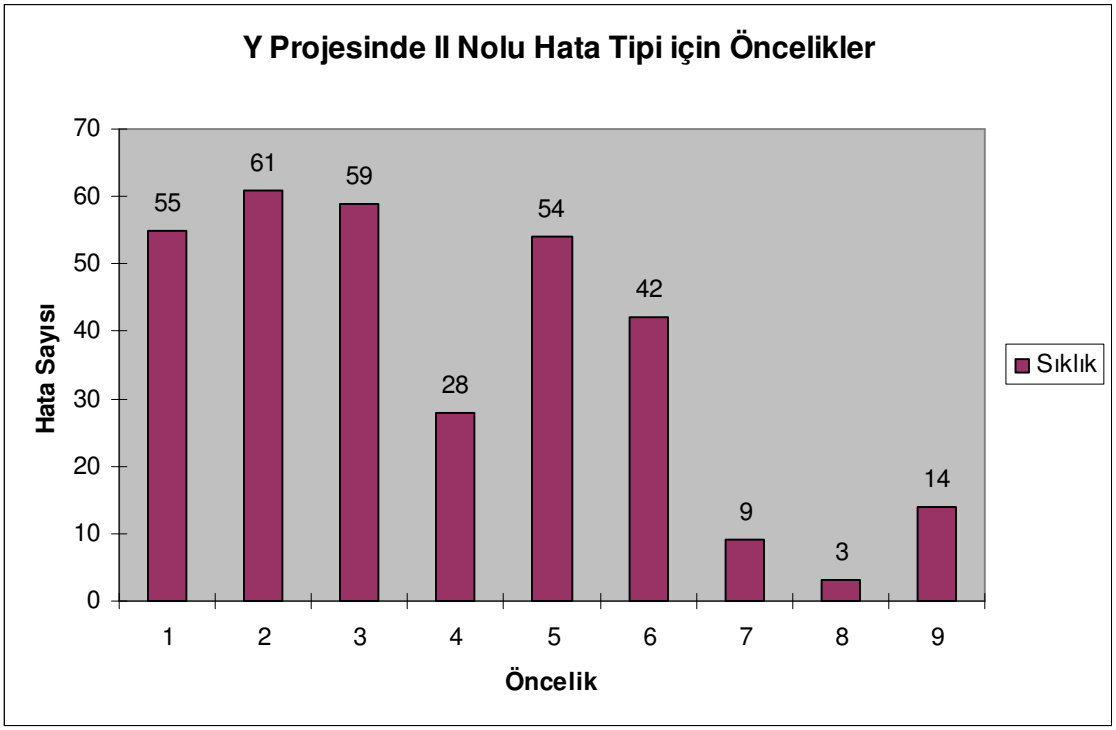
En yüksek hata sayısına ve testten dönen hata sayısına sahip Y projesine ilişkin tespit edilen hataların tipleri incelendiğinde ise aşağıdaki tablo ile karşılaşılmıştır:

Çizelge 5.1 Y projesine ilişkin hata tipi dağılımı

| Hata Tipi | Hata Sayısı |
|---|---|
| I nolu hata tipi | 22 |
| II nolu hata tipi | 325 |
| III nolu hata tipi | 24 |

Firmanın Y projesi için II numaralı hata tipi sayısının diğer hata tiplerine göre çok fazla olduğu görülmüştür. Bu durumda, firma, neden bu projede II nolu hata tipinin sayısının bu kadar yüksek olduğunu araştırmalı ve bu doğrultuda süreç iyileştirme çalışmaları başlatmalıdır.

Y projesindeki yüksek sayıda uygunsuzluğun tespit edildiği II nolu hata tipindeki uygunsuzlukların öncelikleri incelendiğinde ise Şekil 5.5. elde edilmiştir.




Şekil 5.5. Y projesine ilişkin hata tiplerinin önceliklerinin dağılımı

Yukarıdaki Şekil 5.5., Y projesinde tespit edilen II nolu hata tipindeki uygunsuzlukların büyük bir kısmının yüksek önceliğe sahip olduğunu göstermektedir.

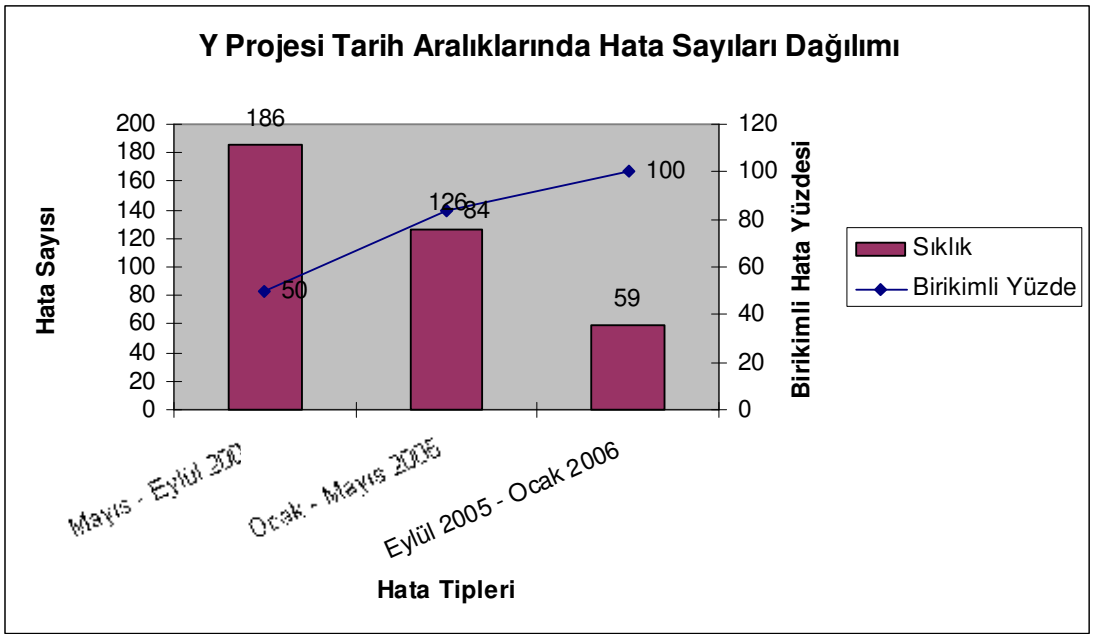
Bu durum Y projesine ilişkin proje takviminin sapmasına neden olabilir. Yüksek öncelikli hata sayılarının nedenleri incelenmeli ve bu doğrultuda iyileştirme çalışması başlatılmalıdır.

Hata tipleri altı aylık periyotlarda incelendiğinde aşağıdaki çizelge elde edilmiştir:

Çizelge 5.2 Y projesinin belli periyotlarda tespit edilen hata tipleri dağılımı

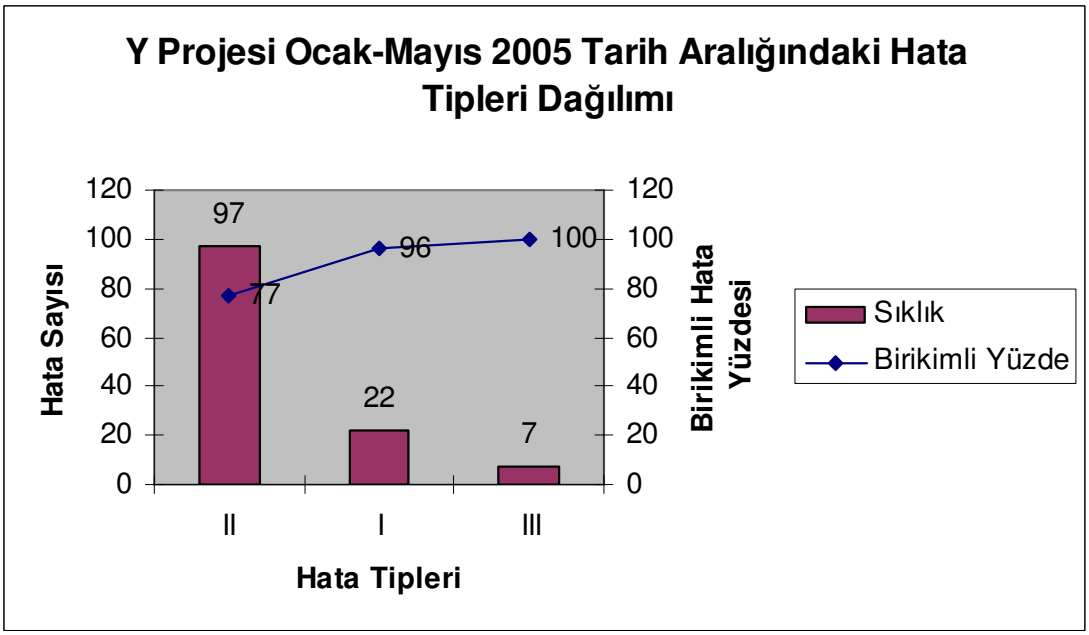
| | Ocak-Mayıs 2005 | Mayıs-Eylül 2005 | Eylül 2005-Ocak 2006 |
|---|---|---|---|
| I nolu hata tipi | 22 | 0 | 0 |
| II nolu hata tipi | 97 | 173 | 55 |
| III nolu hata tipi | 7 | 13 | 4 |

Y projesine ilişkin Çizelge 5.2, projenin Mayıs-Eylül 2005 tarihleri arasında en yüksek II nolu hata tipini tespit ettiğini göstermektedir. Firma, geriye dönük olarak Y projesinde, bu tarihler arasında hata sayısının neden diğer tarih aralıklarına göre yüksek çıktığını araştırmalıdır. Yukarıdaki verilere dayanarak çizilen pareto diyagramı Şekil 5.6'da gösterilmektedir.

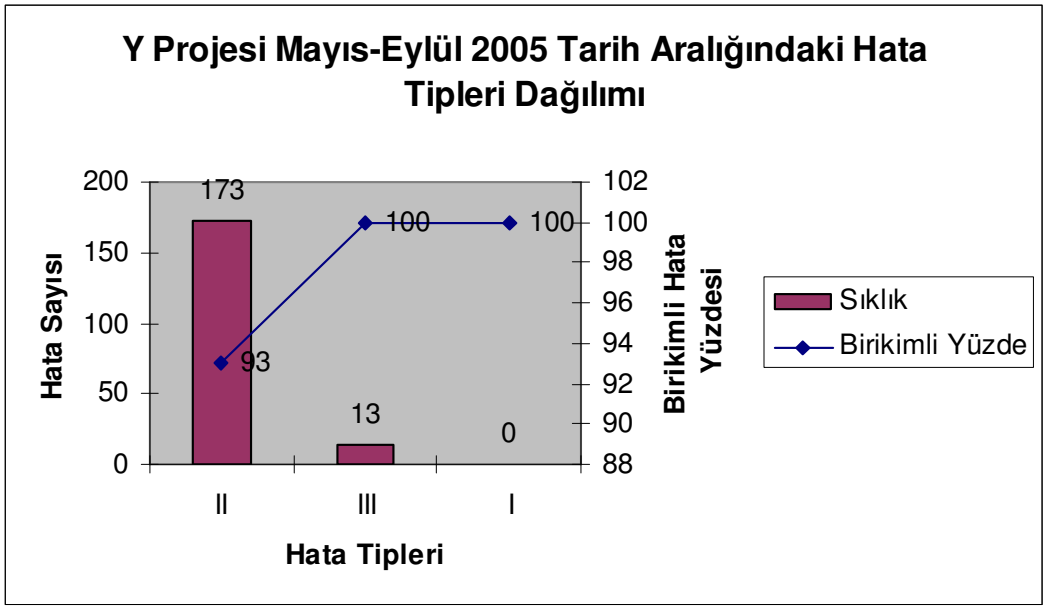



Şekil 5.6. Tarih aralıklarına göre, Y projesi için hata sayıları pareto diyagramı

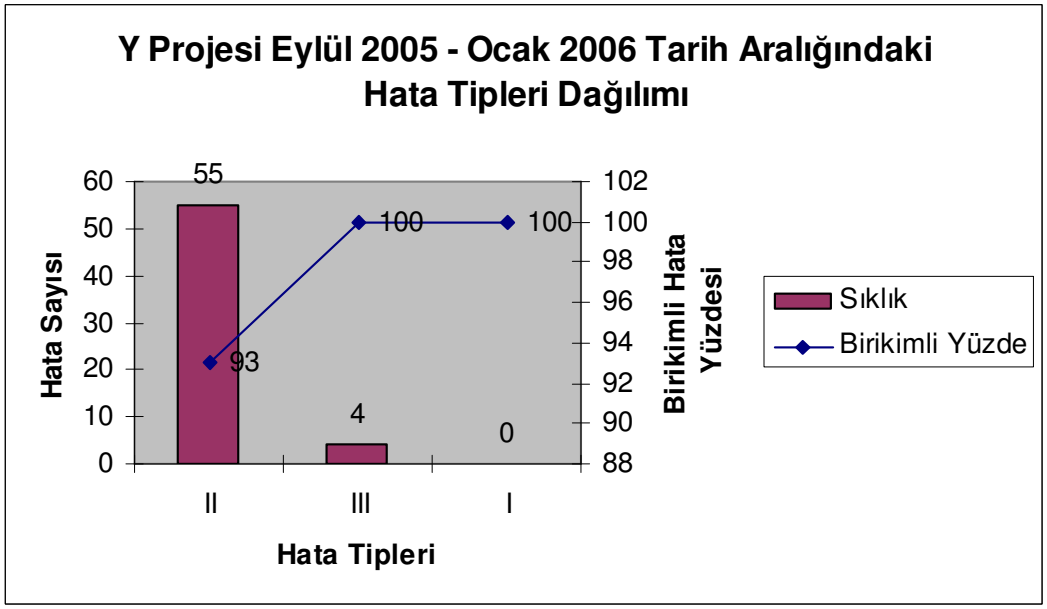
Y projesine ait tarih aralıklarındaki hata tipleri incelenirse:




Şekil 5.7. Ocak-Mayıs 2005, Y projesi için hata tipi pareto diyagramı




Şekil 5.8. Mayıs - Eylül 2005, Y projesi için hata tipi pareto diyagramı




Şekil 5.9. Eylül 2005 - Ocak 2006, Y projesi için hata tipi pareto diyagramı

Şekil 5.7., 5.8. ve 5.9.'da II nolu hata tipinin yani kullanıcı tarafından tespit edilen hata sayısının diğer hata tiplerine oranla çok yüksek olduğu görülmüştür. Y projesinde doğrulama aktivitesinin yapılmadığı ya da doğru şekilde gerçekleştirilmediği sonucuna varılmaktadır. Firmanın müşteri karşısında başarılı olabilmesi için en az hatayla ürünü teslim etmesi gerekliliği unutulmamalıdır. Bu doğrultuda doğrulama aktivitelerine gerekli önemin verilmesi önerilmektedir.

# 6. SONUÇ

Mevcut literatür incelenerek bir yazılım firmasında iyileştirme programına başlamadan önce İSK tekniklerinin iyice anlaşılması gerekir. İSK tekniklerini uygulamaya geçmeden önce aşağıdaki maddelerin üzerinden giderek uygulama gerçekleştirilmelidir:

- İSK tüm yazılım süreçlerine uygulanamaz,
- İSK yazılım firmasında sadece kritik süreçlere uygulanabilir,
- Tüm İSK teknikleri yazılım süreçleri için uygun değildir,
- Süreçler iyi tanımlanmalı, ancak bu durumda İSK teknikleri başarıyla uygulanabilir.

Bir yazılım firmasında süreç iyileştirilmesi için ön koşul, yazılım firmasının belli bir olgunluk seviyesine ulaşmasıdır. Bununla kastedilen, süreçlerin işlerliğinin sağlanması, dokümante edilmesi, belli bir standart doğrultusunda firma çalışanlarının aynı dili konuşmasının sağlanmasıdır. Bu aşamadan sonra yazılımın ölçülerek iyileştirilmesine başlanmalıdır. Olmayan bir şeyin iyileştirilemeyeceği gerçeğinden yola çıkarak, firmaların nereye gelmeyi istediklerini bilmeleri ve müşteri ihtiyaçlarını doğru bir şekilde analiz etmeleri gerekmektedir. Sorulması gereken soru: “Yazılımın geliştirilme sürecinde, neyi ölçersem yazılım ürünümü iyileştirebilirim?”dir. Bunun için öncelikle iyi tanımlanmış bir ölçüm sürecine, bunu toplayacak tarafsız bir kalite grubuna hatta otomasyona ihtiyaç vardır.

Bir süreci tam olarak anlamak için zaman içerisinde süreç değişikliklerinin nasıl hesaplanacağını bilmek gerekir. Bunu yapmak için, hata verisi (toplam hata sayısı, belirli tip hata sayısı gibi) belirlenen bir periyot zamanında (gün, hafta, ürünün yayımlanma zamanları gibi) toplanmalı, analiz edilmeli ve yorumlanmalıdır. Daha detaylı araştırma, yazılım geliştirme aktivitelerinde nelerin düzeltilmesi gerektiğini işaret eder.

Çalışma sonunda gerçekleştirilen uygulamada, firmanın gerekli ölçümleri toplayamamasından dolayı yeterli analiz teknikleri uygulanamamıştır. Yazılım firmaları için hatanın tespit edildiği aktivite (gözden geçirme, test vb.) ve yazılım

geliştirme süreci safhası (analiz, tasarım, geliştirme, test veya bakım) firmanın nerelerde sorun olduğunu görmesi ve bu noktaları iyileştirme çalışmaları başlatması açısından önemlidir. Örneğin, eğer gereksinimler üzerinde çok fazla değişiklik yapılıyorsa, bu, gereksinim analizi safhasına gerekli önemin verilmediği veya gözden geçirmelerin doğru bir şekilde, doğru kişiler tarafından yapılmadığı sonucuna varılarak bu noktaların iyileştirilmesi için çalışmalar başlatılmalıdır.

## KAYNAKLAR


Altınordu, B., 2002, CMMI Modeli ve Yazılımda Kalite, http://www.btnet.com.tr/koseyazi.phtml?kategori=14&yazi_id=400000153

Akın, B., Öztürk, E., 2005, İstatistik Proses Kontrol Tekniklerinin Bilgisayar Ortamında Uygulanması, VII. Ulusal Ekonometri ve İstatistik Sempozyumu, İstanbul Üniversitesi, İstanbul, 26-27 Mayıs 2005, 3s.

Aksoy, H. H., 2005, Yönetim Ve İşletmede Kalite Kavramı, Ankara

AltıSigma, 2003a, İstatistiksel Proses Kontrol, 12 Şubat, http://www.altisigma.com/modules.php?name=News&file=article&sid=15

AltıSigma, 2003b, Proses Kontrol Çizelgeleri, 14 Mart, http://www.altisigma.com/modules.php?name=News&file=article&sid=32

ANSI Z1.1-1985 (ASQC Standard B1-1985) 9. madde, 2p

Atalay, İ., 2003, Yetenek Olgunluk Modelleri (CMM). http://www.ilkeratalay.com/articles/cmmi.php

Bircan, H., Gedik, H., 2003, Tekstil Sektöründe İstatistiksel Proses Kontrol Teknikleri Uygulaması Üzerine Bir Deneme, Cumhuriyet Üniversitesi İktisadi ve İdari Bilimler Dergisi, Cilt 4, Sayı 2, 71s,

Burr, A., Owen, M., September 1996, Statistical Methods for Software Quality, Coriolis Group, International Thomson Computer Press, London, ISBN: 1-85032-171-X, xvi p.

Canbolat, Y. B., 2000, Azerbaycan'da İlk Kalite Çemberleri Uygulaması, Journal of Qafqaz University, Volume III Number I, 117s,

Card, D., 1994, Statistical Process Control for Software, IEEE Software, May 1994, 97p.

Cebeci, Z., 2001, Çukurova Üniversitesi, http://cebeciz.cukurova.edu.tr/content/bilisim/yazilimtestleri.asp

Chrisis, M. B., Konrad, M., Shrum, S., 2004, CMMI Guidelines for process: Integration and Product Improvement, Addison Wesley, 255p.

Çetin, C., Akın, B., Erol, V., 2001, Toplam Kalite Yönetimi ve Kalite Güvence Sistemi, 2. Baskı, İstanbul: Beta Yayım, 418s,

Durman, B. M., Pakdil, F., 2005, İstatistiki Proses Kontrol Uygulamaları İçin Bir Sistem Tasarımı, VII. Uluslararası Ekonometri ve İstatistik Sempozyumu, İstanbul Üniversitesi, İstanbul, 26-27 Mayıs 2005, 2s.

Eickelmann, N., Anant, A., 2003, Statistical Process Control: What You Don't Measure Can Hurt You!, IEEE Computer Society, March/April 2003, Vol. 20, No. 2, 49p.

Fenton, N., Krause, P., Neil, M., 2002, Software Measurement: Uncertainty and Causal Modeling, IEEE SOFTWARE, July / August 2002, 116p,

Garcı´aa, F., Bertoab, M. F., Caleroa, C., Vallecillob, A., Ruı´za, F., Piattinia, M., Generoa, M., 2005, Towards a Consistent terminology for software measurement, Information and Software Technology, June 2005, 1p.

Gençyılmaz, G., Zaim, S., 1999, Eğitimde Toplam Kalite Yönetimi, İ.Ü. İşletme Fakültesi Dergisi, Cilt: 28, Sayı: 2/, 4s.

Goldenson, D. R., Jarzombek, J., Rout, T., 2003, Measurement and Analysis in Capability Maturity Model Integration Models and Software Process Improvement, The Journal of Defense Software Engineering, Jul 2003, 21p.

Gupta, R., 2002, Use of Statistical Process Control : An Experiential Learning at Satyam, The SEPG Conference on Tour in Asia Pac 2002, India, February 25 – 28, 2002, Asia Pac., pp. 5-6

ISO/IEC 12207:1995 Information technology - Software life cycle processes First edition, 1995-08-01, 46p.

ISO/IEC TR 15504-1:1998 (E)

Jacob, A. L., Pillai, S.K., 2003, Statistical Process Control to Improve Coding and Code Review, IEEE SOFTWARE Published by the IEEE Computer Society, May/June 2003 (Vol. 20, No. 3), ISSN: 0740-7459, 5p.0

Kalite Teknikleri, 2002, Vakıf ve Şirketleri Aylık Bülten, Sayı 4, 01-31 Ekim 2002, http://www.euvakif.org/gundem/gundem04.htm

Küçükkongar, A., 2002, İstatistiksel proses kontrol tekniklerinin karşılaştırılması, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara, 27s.

O'Regan, G., 2002, A practical Approach to Software Quality, Springer, ISBN: 0387953213, 1p.

Özkan, M., 2001, Yazılımda Kalite, Computer Life, Sayı: 53 http://www.bilgiyonetimi.org/cm/pages/mkl_gos.php?nt=55

Parekh, N., 2005, Software Verification & Validation Model – An Introduction, http://www.buzzle.com/editorials/4-5-2005-68117.asp

Sargut, K. U., Demirörs, O., 2004, Utilization Of Defect Density Metric For İSK Analysis, No:1 Winter 2004, Software Quality, 20p.

Sargut, K. U., Demirörs, O., 2005, Utilization of İSK in Emergent Software Organizations: Pifalls and Suggestions, 1p.

Sarıdoğan, M. E., 2004, Yazılım Mühendisliği, Papatya Yayıncılık, ISBN: 9756797576, İstanbul, s. 325-329

TR Software, 2004, Yazılımda Kalite, http://www.tr-software.com/tr-software/nshow.aspx?rid=39&lng=2&pge=4

Wallace, D. R., Ippolito, L. M., Cuthill, B., 1996, Reference Information for the Software Verification and Validation Process, NIST (National Institute of Standards and Technology) Special Publication 500-234, March 29 1996, 73p,

## EKLER

### Ek 1 Kontrol Kart Tipleri

Ölçülemeyen özellikler için kontrol kartları;

| Kart Türü | Kullanıldığı Yer | Standartlar bilindiğinde | | Standartlar bilinmediğinde | |
|---|---|---|---|---|---|
| | | Merkez Çizgi | Kontrol Limitleri | Merkez Çizgi | Kontrol Limitleri |
| p | Hatalı oranı | p | $p \pm 3\sqrt{p(1-p)/n}$ | $\bar{p}$ | $\bar{p} \pm 3\sqrt{\bar{p}(1-\bar{p})/n}$ |
| np | Hatalı sayısı | np | $np \pm 3\sqrt{np(1-p)}$ | $n\bar{p}$ | $n\bar{p} \pm 3\sqrt{n\bar{p}(1-\bar{p})}$ |
| c | Hata sayısı | c | $c \pm 3\sqrt{c}$ | $\bar{c}$ | $\bar{c} \pm 3\sqrt{\bar{c}}$ |
| u | Birim başına hata sayısı | u | $u \pm 3\sqrt{\frac{u}{n}}$ | $\bar{u}$ | $\bar{u} \pm 3\sqrt{\frac{\bar{u}}{n}}$ |

Ölçülebilen özellikler için kontrol kartları;

| Kart Türü | Standartlar bilindiğinde | | Standartlar bilinmediğinde | |
|---|---|---|---|---|
| | Merkez Çizgi | Kont. Limitleri | Merkez Çizgi | Kont. Limitleri |
| $\bar{X}$ (Ortalama) | $\mu_x$ | $\mu_x \pm A\sigma_x$ | $\bar{\bar{X}}$ | $\bar{\bar{X}} \pm A_1\bar{S}$<br>$\bar{\bar{X}} \pm A_2\bar{R}$ |
| R (Dağ.Gen.) | $d_2\,\sigma_x$ | $D_2\sigma_x$<br>$D_1\sigma_x$ | $\bar{R}$ | $D_4\bar{R}$<br>$D_3\bar{R}$ |
| S (Std. Sapma) | $c_2\,\sigma_x$ | $B_2\sigma_x$<br>$B_1\sigma_x$ | $\bar{S}$ | $B_4\bar{S}$<br>$B_3\bar{S}$ |

**Ek 2 Sürecin Kontrol Dışı Olduğu Durumlar**

Bazen tüm örneklem noktaları kontrol sınırları içinde rasgele bir davranış göstermez. Örneğin, aşağıdaki maddeler ile karşılaşılması durumunda süreç için istatistiksel olarak kontrol dışındadır yorumu yapılır. Çizelgeye dayalı kontrolde aşağıdaki şartlar bir araya geldiğinde süreç kontrol dışına çıkmış sayılır:

- Bir ya da daha fazla noktanın sınır dışına çıkması
- Art arda 7 noktanın hepsinin, ortalamanın altında ya da üstünde kalması
- Art arda her 11 noktadan 10'unun, her 14 noktadan 12'sinin ya da her 20 noktadan 16'sının ortalamanın altında ya da üstünde kalması
- Art arda 7 noktanın artan ya da azalan bir eğilim göstermesi
- 3σ çizgisi yakınındaki her üç noktadan ikisinin, 2σ çizgisi dışına taşması
- Noktaların periyodik değişim göstermesi
- Noktaların büyük çoğunluğunun 1.5σ aralığı içinde kalması. Bu durum istenen bir özellik gibi görünse de, dağılımın normal olmadığına işaret eder. Ya limitler yanlış seçilmiştir; ya da örnek grupları hatalı oluşturulmuştur.

Ortalama çizgisinin bir tarafındaki ardışık noktalar (8'i bir sırada ya da 12 taneden 11 gibi) süreç ortalamasında değişikliği işaret eder. Ortalama çizgisinin yakınında veri noktasının olmadığı yukarı ya da aşağı eğimlerin olduğu modeller fazla düzenlemeyi ya da iki sürecin varlığını işaret eder. Veri noktalarının artan ya da azalan dizi ile sabit ya da eğimli olması süreçteki kademeli değişikliği gösterir. Tüm veri noktalarının ortalama çizgisinin yakınlarında yer alması güvenilir olmayan veriyi işaret edebilir. Döngü ya da tekrar eden veri nokta serileri tekrar eden süreci işaret eder.

## Ek 3 Yazılımda Kullanılan Temel İSK Teknikleri

### Kontrol kartları

Süreç sapmasını değerlendirmekte kullanılan öncelikli istatistiksel teknik, kontrol karttır. Kontrol kart tüm süreç ortalaması ve kontrol limitleri ile ilişkili ardışık süreç ölçümlerini gösterir.

Kontrol kart kavramı, ilk olarak 1924 yılında Bell Telephone Laboratories elemanlarından W.A. Shewhart tarafından getirilmiştir. Kontrol kartın amacı, genel değişkenlik faktörlerini özel değişkenlik etmenlerinden ayırarak süreçteki normal olmayan değişimlerin önüne geçmektir (Altısigma, 2003b).

Kontrol kart, orta çizginin her iki tarafında yer alan kontrol limitleri ve bir merkez çizgiye sahip hareketli bir karttır (run chart). Kontrol limitleri Üst Kontrol Limit (ÜKL – upper control limit) ve Alt Kontrol Limit (AKL – lower control limit) olarak adlandırılır. Orta çizgi ve her iki kontrol limiti, süreç faaliyeti esnasında bir seri gözlem sonucunda toplanıp hesaplanan tahminleri sunar. Orta çizgi genellikle gözlemlenen süreç ortalamasıdır. Geleneksel (Shewhart) kontrol limitleri merkez çizgiden ±3 sigmadır (±3σ). Sigma, kart üzerinde çizilen verinin tahmin edilen standart sapmasıdır. İki kontrol limit arasındaki fark, sürecin normal çalıştırılması için sınırı ve sürecin sunduğu genel nedenlerden kaynaklanan etkileri işaret eder. 3σ-kontrol çizelgelerinde, genel değişkenliğe bağlı olarak kontrol dışına çıkma olasılığı % 0.027'dir. ±3σ genişliği aksiyon limitleri (action limits) olarak adlandırılırken ±2σ genişliği uyarıcı limitler (warning limits) olarak adlandırılır. Bu limitlerin kullanılması için kurallar pratikte farklılık göstermektedir. Genellikle, uyarıcı limitler kontrol limitleriyle aynı anlamda kullanılırlar. Bunun yanı sıra, uyarıcı limitlerin en önemli amacı erken uyarı ya da dikkati sağlamaktır.

Süreç devam ettikçe elde edilen değerler çizelgeye işlenir. Bu şekilde sürecin istatistiksel özellikleri görsel biçimde sunulmuş olur. Bu işlem süreç esnasında firmanın denetimine kolaylık sağladığı gibi, aynı zamanda çizelge değerlerinin analiziyle sürekli bir süreç iyileştirmesine gidilebilir. Süreç değerlerinin kontrol limitleri dışına çıkması, süreçte özel değişkenliğin mevcut olduğunu gösterir. Bu durumda, kontrol kart dışında belli yöntemler kullanılarak sebeplerin araştırılması ve düzeltici/önleyici faaliyetlerin başlatılması yoluna gidilir.

**Ek 3'ün Devamı**

Kontrol kartın esas ve en önemli amacı, üretim sektöründe kaliteyi kontrol etmek için tanımlı bir prosedür sağlamaktır. Bununla birlikte, gelecekteki projelerin kontrolü için bir sistem kurmadan önce ürünün kalitesinin geçmişinin bir resmini elde etmek önemlidir. Bu bilgi sayesinde kartın üzerinde ilk olarak yer alacak geçici kontrol limitlerinin ne olacağına karar verilir. Dolayısıyla, pratikte, hangi kontrol kartın kullanılacağına karar vermek için ilk amaç genellikle geçmiş kalite kayıtların analiz edilmesidir. (ANSI Z1.1–1985).

Kontrol kart uygulamalarında, örnekler gruplar halinde alınır. Grubun kendi içinde hesaplanan ortalaması, çizelgeye örnek değeri olarak işlenir. Pratikte grup örnek adedi (n) olarak 3–5 gibi değerlerin seçilmesi uygun olur. Örneklerin gruplar halinde alınması, çizelgeyi herhangi bir değişkenliğe daha duyarlı hale getirir. Bu şekilde yapılan örneklemede her bir grubun standart sapması, tek tek örneklerin standart sapmasına oranla daha düşüktür.

Nitel veriler (özellik) ve nicel veriler (değişken) için hazırlanması olası pek çok farklı tip kontrol kart ve bunlara ait merkez ve kontrol limitleri EK 1'de verilmiştir. Sürecin istatistiksel olarak kontrolde olduğu duruma ilişkin bilgi Ek 2'de verilmiştir.

Kontrol kartları hemen hemen ölçülebilen tüm faaliyet alanları için uygulanabilir. Yazılım için bazı örnekler verirsek: hata sayısı, hata önceliği ve tipi, eğitim için harcanan emek, uygulama zamanı ve belirli bir zaman periyodu başına problem raporu sayısı vb.

**Histogram**

Histogramlar, ölçüm değerlerinin dağılımını gösteren ve bu dağılımın standart limitlerine göre durumunu belirten bir çubuk diyagram kartlarıdır.

**Pareto diyagramı**

Süreçte problemi oluşturan nedenin bulunmasına ilişkin en çok kullanılan temel tekniklerden biri de veriye dayalı olan, Vilfredo Pareto tarafından geliştirilmiş Pareto diyagramıdır. Pareto'ya göre problemlerin kaynaklarının %80'i, tüm problemlerin %20'sini oluşturan basit nedenleri ortadan kaldırmakla çözümlenebilir. 80_20 kuralı

**Ek 3'ün Devamı**

olarak da adlandırılan Pareto diyagramı, az sayıdaki önemli sorunu, çok sayıdaki önemsiz sorundan ayırma tekniği biçiminde de tanımlanabilir.

Pareto diyagramını oluşturan adımlar şu şekilde özetlenebilir:

1. Hangi, sorunların araştırılacağına ve verilerin nasıl toplanacağına karar verilir.
2. Analiz için ölçü birimi seçilir (saat, gün, adet gibi).
3. Analiz edilecek veriler için zaman aralığı seçilir. Bu bir kaç saat sürebileceği gibi günler, aylar bile gerekebilir.
4. Pareto diyagramı için veri çizelgesi hazırlanır.
5. Dikey eksenlerden soldaki eksen etkenlerin sıklıklarını, sağdaki ise birikimli sıklıkları gösterir.
6. Yatay eksen üzerinde etkenler sıklıklarına göre azalan bir biçimde sıralanır. Sıklığı çok az olan birkaç etken birleştirilerek "diğerleri" olarak adlandırılan bir etken oluşturulabilir. Bu etken yatay eksen üzerinde en son sıraya yerleştirilir.

**Dağılım diyagramı (Saçılım)**

Dağılım diyagramı, üretilen ürünün kalitesini etkileyen herhangi iki özellik arasında ilişkinin var olup olmadığını belirler. İki değişken arasındaki ilişkinin miktarı korelasyon katsayısı ile belirlenir. Bu istatistiksel yöntemle, ilişkinin derecesi ve yönü bir katsayı ile ortaya konulur.

**Neden-Sonuç diyagramı (Cause and Effect Diagrams)**

Neden-sonuç diyagramı, Prof. Dr.Kaoru Ishakawa tarafından verildiğinden Ishakawa Diyagramı ya da Balık Kılçığı Diyagramı olarak da bilinir. Bu yöntem, kalite çemberleri, kalite sorunlarının nedenlerini saptamakta ve izlemekte kullanılır.

Ele alınan probleme ilişikin her ana etken ve bu etkenlerden kaynaklanan olası alt etkenlerin saptanması ve incelemesi sağlanır.

**Ek 3'ün Devamı**

**Kontrol tabloları**

Kontrol tablosu, örnek gözlemlerden veri toplayarak model çıkarma ya da problem çözme gibi işlemlerin akıllı bir bağlama noktasıdır (TKY 1992: 12). Kontrol tablosu, üretim sırasında hangi olayların ne sıklıkta meydana geldiğini gösteren, kullanım ve anlaşılması basit bir formdur. Kontrol tabloları sürecin özelliğine göre iki farklı grupta düzenlenir (Bircan ve Gedik, 2003):

**Gruplandırma**

Belli kategorilere ve özelliklere göre bilgilerin gruplandırılma sürecidir. Gruplandırma, sorunları çözmede yardımcı bir yöntemdir. Çözüm sürecine yardım etmekte fakat kendi başına sorunları çözememektedir. Ancak, çözüme yaklaşımda uygulayıcılara önemli bilgiler sağlamaktadır (Küçükkongar, 2002).

## ÖZGEÇMİŞ

Adı Soyadı : Özden Gür Çivlik

Doğum Yeri : Ankara

Doğum Yılı : 1976

Medeni Hali : Evli

Eğitim ve Akademik Durumu:

Lise: 1990-1993 Kocatepe Mimar Kemal Lisesi

Lisans: 1993-1997 Hacettepe Üniversitesi Fen Fakültesi İstatistik Bölümü

Yüksek Lisans: 1997-20.. Hacettepe Üniversitesi Fen Fakültesi İstatistik Bölümü

Yabancı Dil: İngilizce

İş Tecrübesi:

Kasım 2001 - Temmuz 2005 Cybersoft Ltd. Şti.

- Firma içi kalite yönetimi; süreçlerin takibi ve sistemin işlerliğini sağlanmak,
- Yazılım Kalite Güvence faaliyetlerini gerçekleştirmek ve YKGP hazırlamak,
- Yazılım Konfigürasyon Yönetimi faaliyetlerini gerçekleştirmek ve YKYP hazırlamak,
- SPICE referansı ile projeleri kendi içerisinde değerlendirmek,
- UML başlangıç düzeyde eğitim vermek,
- UML diyagramlarını gözden geçirmek,
- Firma ve proje bazında iç denetimleri planlamak, gerçekleştirmek ve raporlamak,
- Sistemin, ISO 9001 standardına uygunluğunu sağlamak.

Nisan 2000 - Kasım 2001  MilSoft Yazılım Teknolojileri A.Ş.

- Firma içi kalite yönetimi; süreçlerin takibi ve sistemin işlerliğinin sağlanmak,
- Yazılım Kalite Güvence faaliyetlerini gerçekleştirmek ve YKGP hazırlamak,
- Yazılım Konfigürasyon Yönetimi faaliyetlerini gerçekleştirmek ve YKYP hazırlamak,
- Firma içi Dokümantasyon Kontrolü ve Takibi,
- İç Denetim (Auditing) yapmak,
- Geliştirilen uygulamalarda veri tabanı oluşturulmak ve düzenli veri girişinin sağlanmak,
- İstatistiksel analizler sonucunda Pareto Analizini yapmak ve Histogramları oluşturmak,
- Sürekli geliştirilmekte olan yazılım testlerine katılmak ve denetlemek,
- Kalite standartlarının takibini sağlamak ve standartları yorumlamak.