HACETTEPE UNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Ad : **Serdar**

Soyad : **GÜL**

Numara : **20421689**

Dersin Adı : **445 Yazılım Mühendisligi Kavramları**

Konu : **Okuma Ödevi (Ödev 6)**

Ders Sorumlusu : **Ayça Tarhan**

Teslim Tarihi : **26.12.2008**

Yazılım Projesi Yönetimi Uygulamaları (Başarısızlık ve Başarı)

1995 ile 2004 arasındaki yazılan yaklaşık 250 adet projenin analizi ilginç bir şablonu gözler önüne seriyor. Başarılı ve tahmin edilen süreç içerisinde gerçekleştirilen büyük çaplı projeleri ,zamanında teslim edilemeyen , bütçe aşımına uğrayan veya tamamlanamadan iptal edilen büyük çaplı projelerle karşılaştırırken 6 yaygın problemin gözlendiği tespit ediliyor. Bunlar zayıf proje planlama , zayıf maliyet kestirimi , zayıf çözümlemeler , dönüm noktalarını takip etmedeki zayıflıklar , zayıf değişiklik kontrolleri ve zayıf kalite yönetimidir. Buna karşın başarılı projelerin bu durumlardaki başarısının ortalamanın genelde üzerinde olduğunu görebiliriz. Belki en ilginç nokta , bu sorunların teknik ekipten çok proje yönetimi ile bağlantısının olmasıdır. 2 çalışma hipotezi ortaya çıkarıldı.

- Zayıf kalite kontrolü maliyet ve zaman aşımında en büyük katkı sahibi olan problemdir.
- Zayıf proje yönetimi , yetersiz kalite yönetimine en çok sebep olan şeydir.

Bu makale sahibinin şirketi tarafından 1995 ile 2004 arasında yaklaşık 10000 fonksiyon noktası içeren , ki bu da C programlama dilinde yaklaşık 1 milyon ifadeye eşdeğerdir , 250ye yakın projenin analizinden türetilmiştir.

Ortalamanın altında veya üstünde olan , projelerdeki başarılı veya başarısız metotları belirlemek zordur. Buna rağmen kutupsal tersleri hesaplanıp , son derece ilginç sonuçlar ortaya çıkabilir.Kutupsal ters deyimi projelerdeki maliyet , zamanlama ve kalite hedeflerindeki tayflarda olası ters bitişleri belirtir. Projelerde %35den fazla zaman , maliyet aşımı ve olduğu zaman veya proje tesliminden sonra tecrübeli kişiler ciddi problemleri bu değerler olmadan ortaya çıkardığı zaman , bazı ilginç şablonlar görülebiliyor.

Analiz edilen 250 projede , 25i zaman , maliyet ve kalite bakımından başarılı olarak bulunurken , yaklaşık 50 tanesinde zaman aşımı veya maliyette %35in üzerinde fazla tespit edildi.175 tanesinde ise büyük maliyet ve zaman aşımı , ya da tamamlanamadan iptal durumları görüldü. Bu projeler ; system yazılımları , bilgi sistemleri , özelleştirilmiş projeler , ve savunma uygulamaları içermektedir. Bu sonuçlar da göstermektedir ki büyük system geliştirme olayı son derece rizikolu bir girişimdir. Gerçekten sahibi tarafından gözden geçirilen bazı başarısız projelerde uzman tanık olarak çalışırken kontrat ihlalinden dolayı davalar açıldığını gözlemledik.

Bu büyük uygulamalar hem system yazılımlarını hem de bilgi sistemlerini içerir. Hem şirketler hem de hükümet ajansları içindedir. Geliştirme metotlarına bakınca hem şelale hem de spiral geliştirmenşn kullanıldığı görülür. Çevik metotlar kullanılmaz çünkü 100 fonksiyonu geçen büyük projelerde bu tip fonksiyonlar nadirdir. Makaledeki Tablo 1de başarılı ve başarısız projelerde sıklıkla görülen durumların karşılaştırılması yapılmıştır.

Yazar ve arkadaşları , müşteriler tarafından yazılım geliştirme pratiği , kullanılan araçlar , kalite ve verimlilik sonuçlarını çeşitli projeler için değerlendirmek amaçlı kullanılırlar. Ama bu makale gözden geçirilen konular hakkında önyargılı olabilir. Biz zayıf eğitim , tecrübesiz görevliler veya zayıf personel pratiği gibi konularda görüş beyan etmiyoruz. Elbette bu 6 olayın dışında önemli olan pek çok husus bulunmaktadır. Ama bu 6 olay çok sık bir şekilde gözlenmekte olduğundan anlatılmıştır.

Başarılı ve başarısız projelerdeki durumları karşılaştırmadan önce başarı ve başarısızlık için kullanılan anahtar faktörler olan proje planlama ve proje kestirimi arasındaki fakrları gözden geçirmek daha iyi olur.

Proje yönetim araçları deyimi , öncelikli amacı 100lerce hatta 1000lerce üstüste binen veya birbirine bağlı görevi içeren projelerin sofistike zamanlamasını yapan bir çok araçtan oluşan aileye verilen isimdir. Bu araçlar son derece detaylı görevlere kadar düşebilir ve bireysel çalışanların zamanlamasının hakkından gelebilir. Proje yönteim sınıfı içinde Artemis Views , Microsoft Project , Primavera ve

Project Manager Workbench gibi ürünleri bünyesinde barındırabilir.

Buna rağmen proje yönetim araçları ailesi genel amaçlı , doğal ve özelliştirilmiş yazılım boyutlama ve yazılım maliyet kestirimi araçları gibi kabiliyet kestirimi içermeyen araçlardır. Genel proje yönetim araçları kusur yok etme etkinliği gibi kalite sorunlarıyla başa çıkamaz. Proje yönetim araçları kullanışlıdır ama yazılımın tamamen yönetim kontrolünde olması için bazı ek bileşenlerin de olması gerekir.

Yazılım maliyet kestirimi sanayisi ve proje yönetimi araçları sanayisi ayrı olaylardır Proje yönetim araçları yazılım kestirimi araçlarından bir 10 yıl önce , yani 1960lı yıllarda bulunmuştur. İkisi de ayrı işler olmasına rağmen , şu anda teknik olarak birbirlerine katılmaya başlamışlardır.

Özelleştirilmiş yazılım maliyet kestirimi araçları örneğin Before You Leap, CHECKPOINT, Constructive Cost Model (COCOMO) II, CostXpert , KnowledgePlan , Parametric Review of Information for Costing and Evaluation – Software (PRICE-S), Software Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM) ve Software Life Cycle Management (SLIM) gibi araçları içerebilir.

Yazılım yönetim araçları büyük ve karmaşık silah sistemlerinin yönetimi için donanma tarafından geliştirilen çok sayıda tekniğin otomatikleştirilmiş halidir. Mesela PERT 1950li yıllarda savaş gemisi inşası gibi karmaşık askeri projeleri ele almak amacıyla yapılmıştır. Proje yönetim araçları kritik yol analizi , kaynak düzeyleme , Gannt üretimi , zamanlama çizelgeleri gibi bir çok diğer özelliği de barındırır.

Proje yönetim araçları(PYA) ticari yazılım maliyet kestirim (YMKA)araçları gibi yapılandırmasında dikkate alınacak uzmanlık arar. Mesela , SmallTalk gibi nesneye yönelik bir programlama dilinde maliyet ve kalitenin etkisini araştırıyorsak , standart bir PYA doğru bir tercih değildir.

Buna mukabil çoğu YMKA programlama dillerine gore yapılandırılmıştır ve bir uygulamadaki hesaplamayı istenen programlama diline gore yapabilir.

YMKA , PYA'dan 10 sene sonra üretilmeye başlandığından beri YMKA üreticileri nadiren detaylı PERT diyagramları inşası ve kritik yol analizi gibi proje yönetim fonksiyonlarını

kopyalamayı denediler. Bunun yerine YMKA PYA için veri üretimi yapıyor. Böylece ticari tahmin piyasasında PYA ve YMKA arasındaki arayüzler standart bir hale geldi.

Şimdi büyük yazılım uygulamaları için proje planlama ve proje tahmin araçlarına geri dönebiliriz.

Başarılı ve Başarısız Proje Planlama

Proje planlama deyimi işle ilgili ince hesap yapıları , ve çalışan üyelerine görev taksimatı gibi durumları kapsar. Proje planlama çeşitli zamanlama üretimi , Gannt ,PERT çizelgeleri gibi kritik yolları üretmeyi içerir.

Büyük şirketlerdeki büyük projeler için etkili proje planlama uzman planlaması ve otomatik planlama araçlarını içerir. 2004 civarında geniş bir yazılım projesi için başarılı bir planlama aşağıdakileri içerir.

- Artemis Views ve Microsoft Project gibi otomatik planlama araçlarını kullanmak
- Bütün işle ilgili ince hesap yapılarını geliştirmek
- Proje geliştirme faaliyetlerindeki kritik yol analizini idare etmek
- Proje boyunca görevli kiralamasını ve devri daimini göz önünde tutmak
- Taşeronları ve uluslararası takılmları göz önünde tutmak
- Gereksinim toplama ve analizini zamanında yapmak
- Gereksinimlerdeki değişiklikleri zamanında yapmak
- Kalite control faaliyetlerini bir takım olarak zamanında yapmak
- Eğer gereksinimlerin büyümesi dikkate değer ise çoklu piyasaya sürem işleminin yapılması

Başarılı projeler gerçekten bu planlamayı güzel yapmaktadır. Buna rağmen başarısız veya iptal edilen projelerde ise büyük planlama hataları görülmektedir. En yaygın planlama hataları sırasıyla değişen gereksinimleri etkili olarak ele almamak , proje boyunca görevli

kiralaması ve devri daimini önceden tahmin etmemek , detaylı gereksinim analizi için zaman ayırmamak , araştırma , test , hata tamiri için yeterli zaman ayırmamaktır.

Başarılı proje planlama büyük oranda otomatik bir şekilde yapılandırılmaktadır. Piyasada en az 50 adet ticari proje planlama aracı bulunmakta ve başarılı projeler bunlardan en az birini kullanmaktadır. Sadece başta planlama otomatik olarak yapılmamakta , ayrıca yeni varsayımlarla uyuşması açısından gereksinim sahasındaki değişmeler veya harici olaylar planı güncellemeyi tetiklemektedir. Elle yapılan metotlarla güncellemeler kolaylıkla yapılamayacağından proje planlama araçları büyük yazılım projeleri için bir gereklilik haline gelmiştir.

Başarılı ve Başarısız Proje Maliyet Kestirimi

Yazılım maliyet kestirimi (YMK) büyük yazılım projeleri için elle yapılamayacak kadar karmaşık bir işlemdir. Bu gözlem 75 adet , içinde COCOMO II, CostXpert, Knowledge-Plan, PRICE-S, SEER-SEM, SLIM gibi yaygın olarak kullanılan YMKAnın varlığına dayanmaktadır. Başarılı projeler bunlardan en az birini kullanmakta , 2 veya daha çok aracın kullanılması ise garip kaçmamaktadır. Kestirim uzmanları tarafından yapılan kestirimler de başarısız projelerde ele alınmamasında karşılık başarılı projelerde not olarak bulunmaktadır. Büyük sistemler için başarılı bir maliyet kestirimi şunları içermelidir.

- Yazılım kestirim araçları
- Fonksiyon noktalarına dayalı büyük teslimlerin biçimsel olarak boyutlandırılması
- Eğitimli kestirim uzmanlarının veya proje yöneticilerinin varlığı
- Kestirimde yeni ve değişen gereksinimlerin hesaba katılması

- Kalite kestiriminin maliyet ve zaman kestirimi kadar iyi hesaba katılması

Tezat olarak büyük başarısız projeler bu tip YMKAları kullanmamaktadır. Buna rağmen elle yapılan kestirimler 10000 fonksiyon noktası sınırındaki projeler için asla yeterli olmamaktadır.
Başarısız projeler noksan boyutlama yaklaşımları yüzünden çalışmanın boyutunu olduğundan daha eksik gösterme eğilimindedir. Başarısız projeler ayrıca kalite kestirimini de ihmal etmekte , bu durum da aşırı hata düzeylerinde test kısmını yavaşlatıp durmaya sebebiyet verdiğinden büyük bir ihmal olarak sayılmaktadır. Üretkenlin oranının kestirimindeki aşım veya büyük bir system için üretkenlik , zaman ve maliyet aşımı gibi diğer yaygın sebeplere başlı olan küçük projelerdeki üretkenliğe eşit olmalıdır. 10000 fonksiyon nokta sınır büyüklüğündeki projelerdeki kestirimde esas problem aşırı optimizasyon olayında yapılan hatalardır.
Proje planlama araçları ve proje kestirim araçları fonksiyonellik bakımından üstüste binmekte , ve piyasaya ayrı olarak sürülmektedir. Normalde proje planlama ve proje kestirimi araçları birbirlerine ileri ve geri yönde bilgi geçişi sunmaktadır. YMKA kapsamlı proje boyutlama , kaynak kestirimi ve kalite kestiriminde kullanılmaktadır. Proje planlama araçları ise kritik yol analizi , detaylı zamanlama ve işle ilgili ince hesap yapılarında kullanılmaktadır.

Başarılı ve Başarısız Proje Ölçümleri

Başarılı büyük projeler genelde üretkenliği ve kalitelli tarihi verileri elinde bulunduran yazılım geliştirme programlarına sahip firmalarda bulunur. Böylece yeni bir proje ona benzer eski projelerle zaman , maliyet , kalite ve diğer önemli faktörlerin geçerliliği için yargılanmak amacıyla karşılaştırılabilir. 10000 fonksiyon noktalı projelerde kullanılabilecek en kullanışlı ölçümler şunlardır

- Toplam çaba
- Toplam maliyet
- Geliştirme üretkenliği
- Gereksinimlerdeki değişmelerin hacmi ve oranı
- Kaynağa bağlı hatalar
- Hata yok etme etkinliği

Eforun ölçümü iş çöküş yapısını desteklemek için yeterince taneli yapıda olmalıdır. Maliyet ölçümleri tam , geliştirme maliyetlerini , daraltma maliyetlerini , satılan ve kiraya verilen paketlerle ilgili maliyetleri içerecek şekilde olmalıdır. Büyük şirketler ve başarılı projeler için tek bir alanda belirsizlik söz konusudur. Şirketlerin çeşitliliğinin genişliğine gore kurulan yük ve genel giderler. Bu çeşitlemeler şirketler , sanayiler ve ülkeler için karşılaştırmaları çarpıtır ve seviye işaretlemesini zorlaştırır. Elbette tek bir şirket için bu sorun değildir.

Fonksiyon noktaları Amerika ve Avrupadaki yazılım projelerinde kullanılan en yaygın ölçüdür ve kullanımı bütün dünyada giderek artmaktadır. Geliştirme üretkenliği ölçümleri fonksiyon noktalarını 2 şekilde kullanmaktadır , aylık görevli başına fonksiyon noktası ve / veya fonksiyon noktası başına iş saati.
Federal hükümet , bazı askeri projeler ve savunma sanayi hala eski kod satırı ölçüsünü kullanmaktadır. Bu risklidir çünkü gereksinimler , tasarım , dökümantasyon , proje yönetimi , kalite güvencesi gibi çok önemli faaliyetlerin ölçülmesinde kullanılamaz. Visual Basic gibi kod satırını saymada etkili yöntemleri olmayan programlama dilleri de vardır. Yaklaşık bir 3 proje yaygın kullanılan programlama dillerinden birini içerse de diğer bir uygulama da böyle olmayan 12 farklı programlama dilinden birini içerebilir.

Kalite ölçümleri yüksek sıralı yazılımcılar için bir gösterge ve başarılı projeler için evrenseldir. Çökmeye yakın ya da çöken projeler asla kalite ölçümü yapmayanlardır. Kalite ölçümleri kaynaktaki hata hacmi , güçlük seviyesi , güçlük seviyesi hataları , ve hata yok etme oranlarını içerir.

Gerçekten sofistike şirketler ve projeleri hata tok etme etkinliğini ölçerler. Bu geliştirme sırasında ve ürünü müşteriye verdikten sonra belli bir sure içerisinde karşılaşılan hataların toplamı demektir. Mesela geliştirici projeyi geliştiriken 900 hata bulursa ve kullanıcı ürünü aldıktan sonra 3 ay içinde 100 hata bulursa bu proje %90 hata yok etme oranı ile çalışıyor demektir. Bu yüzden kullanıcının bulduğu her hata hata yok etme oranını düşürür.
İlginçtir ki başarılı projelerde hata yok etme oranı Amerika ortalaması olan %85den 10 birim fazla olarak %95 olarak istatistiklere yansımaktadır.
Iptal edilen projelerde müşteri kullanımı gibi bir durum söz konusu olmadığından hata yok etme oranının ölçümü mümkün değildir. Buna rağmen müşteriye satılacak olan projeler , geç olarak , Amerika ortalamasının 5 puan , başarılı projelerin 15 puan altında olarak %80 oranında hata yok etme oranına sahiptir. Bu tanım da evrense hale gelen sadece bir düzine geciken veya bütçe aşımına uğramış etkili kalite yönetimi olmayan projeye dayanmaktadır.
Zaman ve maliyet aşımındaki çoğunluk test kısmında oluşmaya meyillidir ve bu da aşırı hata hacmi yüzünden olur. Bu da büyük sistemlerde etkili bir kalite control sistemi eksikliğinin zaman ve maliyet aşımına büyük bir katkıda bulunduğu hipotezinin ortaya konmasına neden olarak gösterilebilir.

Başarılı ve Başarısız Dönüm Noktası Takibi

Dönüm noktası takibi deyimi yazılım dünyasında belirsizliğini korumaktadır. Bazen bir faaliyetin başlangıcına , bazen bir faaliyetin bitişine , bazen de bir takvim tarihine işaret eder. Bu makalede bu deyim bir anahtar faaliyetin veya anahtar başarılacak şeyin bitişi olarak belirlenmiştir. Normalde bir bitme dönüm noktası bitirilecek şeyin gözden geçirme ve araştırmasının direk sonucudur. Dönüm noktası bir takvim tarihi değildir.

Proje yönetimi dönüm noktalarını inşa etmekten , bitişini gözlemlemekten , ve dönüm noktalarının başarılı bir şekilde bitişini veya karşılaşılan problemleri dürüstçe raporlamaktan sorumludur. Ciddi bir problemle karşılaşıldığı zaman dönüm noktasının bitirildiğini rapor etmeden önce o problemi çözüme ulaştırmak gerekir. Ortalama 10000 fonksiyon noktalı başarılı bir yazılım uygulamasında olması gereken dönüm noktaları şunlardır.

- Gereksinimleri gözden geçirme
- Proje planını gözden geçirme
- Maliyet ve kalite kestirimini gözden geçirme
- Harici tasarımı gözden geçirme
- Dahili tasarımı gözden geçirme
- Kalite planını ve test planını gözden geçirme
- Dökümantasyon planını gözden geçirme
- Yayılma planını gözden geçirme
- Eğitim planını gözden geçirme
- Kod araştırması
- Her geliştirme test aşaması
- Müşteri Kabul testi

Başarısız veya geciken projelerde ciddi dönüm noktası takibi eksikliği vardır. İş devam ederken faaliyetlerin bittiği rapor edilir. Dönüm noktaları işlerin bitmesi veya aktüel bitişler değil alelade takvim tarihleridir. Bazı tür gözden geçirmeler etkisizliğe yol açacak kadar yetersizdir.
Diğer yandan başarılı projelerde dönüm noktası takibi dikkate alınır ve en iyi şekilde yapılmaya çalışılır. Kayıp dönüm noktalarını makul göstermek veya bitmemiş bir işi bitmiş göstermek yoktur. Dönüm noktası bulunan başarılı projelerde , tamamlanmamış dökümanı veya kod segmentini göndermek , hata barındırmak , aşağı doğru genişlemeye desteklememe gibi durumlar olamaz.
Başarılı projelerde dönüm noktası takibinin bir diğer görünüşü problemlerin rapor edildiği ve ya gecikme olduğunda ne yapılacağıdır. Reaksiyon güçlü ve acildir. Çözüme yönelik hareketler planlanır , görev güçleri atanır , ve hataları düzeltmek mümkün olan en hızlı şekilde yapılır. Diğer yandan geri kalmış projelerde problem raporları ile

ilgilenilmez , ve doğrulama hareketleri çok nadir olarak yapılır.

Başarılı ve Başarısız Değişiklik Yönetimi

Ortalama 10000 fonksiyon noktalı uygulamalarda analiz ve tasarım fazları olduğunda ay başına %1 ile %3 arasında bir çalışma gerekir. Bu gerçek gereksinim aşamasındaki fonksiyon noktalarının toplanması ve tasarımdan sonraki fonksiyon noktalarının toplamı ile karşılaştırmaktan bulunmuştur. Eğer en başta 10000 nokta varsa , tasarımdan sonra bu 12000e çıktıysa gelişme oranı %20dir. Analiz ve tasarım için 10 aya harcandıysa aylık gelişme oranı da %2 olarak bulunur.
Gereksinimlerdeki Fonksiyon noktaları toplamı fazı yayılma ile karşılaştırıldığında toplam gereksinim değişimi ilk gereksinimlere oranla en fazla %50 artış gösterir. O yüzden ortalama 10000 fonksiyon noktalı başarılı yazılım projeleri değişmelerin control dışına çıkmadığından emin olmak için modern metotları ve araçları kullanmalıdır.
10000 fonksiyon noktalı uygulamalar için başarılı değişim kontrolü yaparken aşağıdakiler göz önünde tutulmalıdır.

- Ek müşteri/gelişim değişme kontrol yönetim kurulu veya tahsis edilmiş alanında uzmanlar
- Çokme değişmelerini en aza indirmek için ek uygulama tasarımı (JAD) kullanmak
- Çökme değişmelerini en aza indirmek için biçimsel prototipler kullanmak
- Değişmeleri sağlamak açısından planlı iterative geliştirmeyi kullanmak
- Bütün değişim isteklerinin biçimsel olarak gözden geçirilmesi
- 10 fonksiyon noktasından fazlası için bütün değişmeler için değişen maliyet ve zaman kestirimi
- İş etkisi bağlamında değişim isteklerini önceliklendirmek

- Özel sunma ve pazarlama için değişim isteklerinin biçimsel olarak atanması
- Kaynak referanslı otomatik değişim control araçları kullanmak

Biçimsel JAD oturumları kullanımında gözlenen ya ürünlerden biri çökme gereksinimleri değişikliklerindeki azalmadır. Planlanmamış gereksinim yüzeyindeki aylık %1 ile %3 arasındaki oran yerine JAD ve IBM ve diğer şirketlerin çalışmaları JAD teknolojisinin etkisine gore bu oranı planlanmamış gereksinim değişiklikleri için %1in altına indirmektedir.

Prototipler de çökme gereksinim değişikliklerinin azaltılmasında yararlı olmaktadır. Normal olarak anahtar ekranlar , girdiler ve çıktılar prototip haline getirilerek kullanıcıların başkasından aldığı tecrübe ile tamamlanacak uygulamanın neye benzeyeceği hakkında bir fakir sahibi olması sağlanır.

Mamafih değişmeler büyük sistemler için her zaman olacaktır. Her hangi bir gerçek zaman uygulaması için gereksinimleri dondurmak mümkün değildir , ve bunun olabileceğini düşünmek tecrübesizliktir. Buna rağmen yönetim şirketleri değişikliklerle başa çıkmaya hazır ve o kabiliyete sahiptir işlemi engellemeye izin vermezler. Bu yüzden iterative geliştirmenin bazı çeşitleri mantıki bir gerekliliktir.

Başarılı ve Başarısız Kalite Kontrolü

Etkili yazılım kalite kontrolü başarılı projeleri başarısız veya gecikmişlerden ayıran en önemli tek başına olan faktördür. Bunun sebebi de büyük sistemler için hataları bulma ve tamir etmenin yapılabilecek en maliyetli iş olması ve diğer her hangi bir faaliyete gore çok daha fazla zaman istemesidir.

Başarılı kalite control hata önleme ve hata yok etmeyi beraber içerir. Hata engelleme deyimi hata oluşturma ihtimalini en aza indiren veya hatanın başta olmasını

sağlayan eylemlerin tümüne verilen addır. Hata engelleme faaliyetlerine örnek olarak gereksinim toplamak için JAD , biçimsel tasarım metotları kullanımı , yapılandırılmış kodlama teknikleri kullanımı , ispatşanmış tekrar kullanılabilir material içeren kütüphane kullanımı verilebilir. Hata yok etme deyimi hataları bulan ve o hataları yok edip çalışabilir bir hale sokan eylemlerin tümüne verilen addır. Buna da örnek olarak gereksinim araştırmaları , tasarım araştırmaları , döküman araştırmaları , kod araştırmaları ve her tür test işlemleri verilebilir.

Bazı faaliyetlet aynı anada hem hata engelleme hem de hata yok etme işlemleri işini aynı anda yapacak yararı sağlar. Mesela tasarımda ortaklık ve kod araştırması hem hata engellemede etkili olup , aynı zamanda hata yok etmede de işe yaramaktadır. Hata engellemeye yararlı olmasının sebebi araştırma ortaklığının araştırmanın tespit edeceği hataları engelleme hususunda yardımcı olmasından dolayıdır. 10000 fonksiyon noktalı projelerde başarılı kalite control faaliyetleri aşağıdakileri içermelidir.

Hata Engelleme

- Gereksinim toplamak için JAD
- Biçimsel tasarım metotları
- Yapılandırılmış kodlama metotları
- Biçimsel test planları
- Biçimsel test durum yapılandırması

Hata Yok Etme

- Gereksinim araştırmaları
- Tasarım araştırmaları
- Döküman araştırmaları
- Kod araştırmaları
- Test plan ve durum araştırmaları
- Hata yok etme araştırmaları
- Yazılım kalitesi hakkında güvence vermek için gözden geçirme
- Birim test
- Bileşen testi

- Yeni fonksiyon testi
- Gerileme testi
- Performans testi
- System testi
- Kabul testi

Hata engelleme ve hata yok etme yöntemlerinin bir arada bulunduğu hali başarılı ve başarısız projelerde görülen hata sayılarının karşılaştırılmasında gözle görülür derecede farklar oluşmasına yol açar. 10000 fonksiyon noktalı projelerde ; başarılılarında toplam geliştirmede fonksiyon noktası başına 4 adet hata tespit edilir ve bunların %95i müşteriye teslimden önce yok edilir. Bir diğer deyişle teslim edile üründeki hata fonksiyon noktası başına 0.2 olup 2000 adet gecikmiş hatadır. Bunlarda %10 veya 200ü ciddi hatalardır. Geri kalan da küçük boyutlu hatalardır.

Buna ters olarak da başarısız projelerde fonksiyon noktası başına 7 hata bulunur ve bunların %80i müşteriye teslimden önce yok edilir. Bu durumda müşteriye teslim edilen üründeki hata sayısı fonksiyon noktası başına 1.4 ve toplamda 14000 olarak bulunur. Bu hataların %20si yani 280 adedi gayet ciddi hatalardır. Bu büyük sayı kullanıcılar için ciddi problemler doğurabilir.

Başarılı projelerin başarısızlara gore daha çok hata bulma ve yok etmesindeki sebeplerden biri de tasarım ve kod araştırmasıdır. Biçimsel tasarım ve kod araştırması hata bulma konusunda yaklaşık %65 oranında etkili olur. Ayrıca test durumları için daha iyi kaynak materyal sağlayacağı için testing etkisini artırıcı yönde etki eder.

Başarısız projeler tasarım ve kod araştırmasını göz önünde bulundurmaz ve sadece teste dayalıdır. Bu araştırmaların göz önünde bulundurulmaması 3 ciddi probleme yol açar. Bunlar test olayı projeyi yavaşlattığından ve durma noktasına getirdiğinden büyük sayıda hatanın hala proje içinde kalması , araştırma olmadan kötü hata(bad fix) tamiri oranının projeler için korku verecek derecede büyük olması ve sadece teste dayalı kapsamlı hata yok etme

işleminin hata yok etme oranını %80in üzerine çıkaracak kadar yeterli olamaması olarak belirlenebilir.

Bad fix deyimi yama veya hata yok etmenin kendisinden kaza ile kaynaklanan 2. dereceden hatalara verilen addır. Sanayideki ortalaması %7 civarında olup bu ortalama başarısız projelerde %20ye kadar çıkabilir(yani her 5 hata yok etme işlemi yeni bir hataya sebebiyet verebilir) , lakin başarılı projelerde bu oran %2 veya daha azıdır.

Sonuç

Büyük yazılım sistemlerinin başarısız olması için pek çok yol vardır. Sadece bir kaç yol onların başarılı olmasını sağlar. İlginç olan şudur ki proje yönetimi projeleri yolun başarısız veya başarılı olması için iten faktörlerin başında gelir.
Kalite kontrolünde beceriksiz ve proje yönetim görevlerinde yetersiz olan büyük yazılım projeleri genelde başarısızlık veya devasa zaman , maliyet aşımı gibi kötü sonlara doğru yelken açar.
Başarıya ulaşmayı sağlayacak en önemli yazılım geliştirme uygulamaları arasında proje başlangıcından öne planlama ve kestirim , proje sırasında değişen gereksinimleri içine çekmek , ve hataları en aza indirmek bulunur.
Başarılı projeler daima şu konularda üstündür.
Planlama , kestirim , değişim kontrolü ve kalite kontrolü.
Bunun tersi olarak da başarısız veya geciken projeler iyimser planları olan , kusurlu , tahmin edilmeyen değişimleri hesaba katmayan (kestirimde bulunmayan) , değişimleri iyi ele alamayan ve kalite kontrolünde başarısız olan projelerdir.

Yazar Hakkında

Capers Jones Yazılım Üretkenliği Araştırmaları şirketinin kurucusu ve baş bilim adamıdır. Yazılım yönetimi başlığında uluslararası danışman , konuşmacı , seminer başkanı ve yazardır. Jones daha önce ITT Programlama Teknolojileri Merkezi , Stratford Conn.da Programlama Teknolojileri konusunda müdür yardımcısı olarak görev yapıyordu. ITT'ye katılmadan önce 12 yıllık bir süreç boyunca IBM'de araştırma ve yönetici pozisyonlarında çalıştı. Yazılım kalite ve üretkenliği geliştirme metotları hakkındaki çalışmalarıyla IBM Genel Ürün Depatmanı üstün hizmet ödülünü kazandı. Jones'in yazılım proje yönetimi üzerine yazdığı 12 adet kitap ve 200ü aşkın makalesi bulunmaktadır. 20den fazla ülkede 150yi aşkın şirket , askeri servisler , hükümet ajanslarına yazılım proje yönetimi hakkında seminer vermiştir.