Kritik Finansal Sistemlerde Yazılım Değişiklik ve Takip Yönetim Sistemi

Mehmet Vacit BAYDARMAN[1], Ökkeş Emin BALÇİÇEK[2]

[1] Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Konya

[2] Kuveyt Türk Katılım Bankası, Ar-Ge Merkezi, Konya

mehmet.baydarman@kuveytturk.com.tr, emin.balcicek@kuveytturk.com.tr

Özet: Bu bildiride, kritik finansal sistemlere sahip olan kurumlarda, yazılım geliştirme yaşam döngüsünde kullanılan; Geliştirme (Development), Test, Canlı Öncesi (Pre-Production), Canlı (Production) ortamları arasında, yazılıma ait parçaların taşınmasını sağlamak ve taşınmış olan yazılım parçalarının geçmişini izlemek amacıyla önerilen sistem anlatılmıştır. Bununla beraber bu önerilen sistemi uygulayan bir uygulama dizaynı ve fonksiyonları detaylıca izah edilmiştir. Kritik finansal sistemlere sahip olan kurum ve kuruluşlar bu önerilen sistemi ve örnek uygulamayı baz alaraktan kendine ait sistemi oldukça kolay bir şekilde geliştirebileceklerdir. Bu uygulama ile yazılım parçalarından oluşturulan paketler, bir takım roller ile kurulmuş olan onay mekanizmasının kontrolünde canlı ortama kadar taşınıp, belirli yetkilere sahip bir servis aracılığı ile çalıştırılırlar. Paketlerin içerikleri, izledikleri yollar, ortamlar arası geçiş saatleri, paketler üzerinde işlem yapan kullanıcılar listesi vb. raporlar alınabilir. Böylece çok karmaşık ve hata riski oldukça yüksek olan ortamlar arası uygulama geçiş işleminin kusursuz bir şekilde işlemesi sağlanabilmektedir.

Anahtar Sözcükler: Yazılım Geliştirme Yaşam Döngüsü, Yazılım Geliştirme Ortamları, Yazılım Dağıtım Yönetimi, YGYD, Yazılım Değişim Yönetimi

Software Change And Tracking Management System On Critical Financial System

Abstract: In this paper, in corporations with critical financial system, the environment used in the Software Development Life Cycle; in between Development, Test, Pre-Production, Production environments, the system which is proposed in order to enable transportation components of software and monitor the history of software components that were moved. In addition, it is explained in detail that such an application which implements the proposed system design and function. Corporation with critical financial system can develop easily system belonging to themselves by using both this recomended system and sample application. With this application, packages consisted of pieces of software are transferred to production environment by approval mechanism set up with a set of rules and process is completed by a service with spesific authorization. Some report can be received such as the content of the package, the way followed and users processing. Thus, processes which are so complicated and even quite risky are perfectly realized to be processed.

Keywords: Software Development Life Cycle, SDLC, Software Development Environments, Software Deployment Management, Software Change Management

1.Giriş

Büyük ölçekli şirketlerde iş gereksinimleri ve süreçlerin karmaşıklığı dolayısıyla, var olan işlemleri optimum insan kaynağı ve bütçe ile yönetebilmek için yazılım geliştirmek en temel ihtiyaçlardan biridir. Özellikle bankacılık gibi servis odaklı şirketlerde teknoloji kullanımının oldukça fazla olması nedeni ile yazılım ürünlerinin ihtiyacı daha da fazla olmaktadır. Bununla beraber bu ölçekte yazılım geliştirme işinin başarımı ve yönetimi

göründüğünden daha zor bir mühendislik problemidir. Söz konusu yazılım geliştirme sürecinin kurumsallaşması ve bir modele uyarlanması ve bunun bir otomasyon yazılımı ile takip edilmesi oldukça gerekli ve önemlidir. Bu sürecin ismi Yazılım Geliştirme Yaşam Döngüsü'dür (YGYD). Bu sürecin işletilebilmesi için yazılım mühendisliği disiplinin tespit ettiği yazılım geliştirme metodolojilerinden bir tanesini seçmek gerekmektedir. En bilinen yöntemlerden bazıları "Waterfall", "Spiral", "Agile Software Development", "Rapid", "Prototyping", "Incremental" ve "Synchronize and Stabilize" gibidir.

En yaygın kullanılan metodolojilerden birisi olarak şelale yöntemi (waterfall) bildiride ki söz konusu YGYD içerisinde kullanılacaktır. YGYD' nin en temel adımları, analiz, dizayn, yazılım, test, son kullanıcı testi ve dağıtım adımlarıdır. Bu süreçleri işletebilmek için birden fazla rol ve sorumluluklara ihtiyaç duyulmaktadır. Proje yöneticileri, sistem analistleri, iş birimlerinden gelen iş analistleri veya müşteriler, yazılım mühendisleri ve daha bir çok pozisyon sayılabilir.

Yazılım geliştirilme süresinde bu adımların rahat işletilebilmesi için her bir adımda gerekli olan uygulamanın çalıştığı ortamlar vardır. Bu ortamların sayısı yazılımın büyüklüğüne, yazılan sistemin önem derecesine, yazılım organizasyonunda yer alan kişilere göre çeşitlilik gösterir. Bu ortamlar en genel olarak ; Geliştirme (Development) ,Test ,Canlı Öncesi(Pre-Production) ve Canlı (Production) olarak oluşturulur ve kullanılır. Geliştirme ortamı yazılımın ilk geliştirme yeridir. Yazılım mühendisi geliştirmeleri, yazılım mühendisi testleri, sistem analisti testleri bu ortamda yapılır. Test ortamı, kalite kontrol veya test mühendisi rollerinin geliştirilen sistemi test ettiği ve bulguların tespit edilip, düzeltildiği ortamdır. Son kullanıcı ise canlı öncesi ortamda gerçeğe oldukça yakın verilerle sistemi test eder ve onayını verir. Bu onaydan sonra uygulama canlı ortama alınabilir durumdadır.

İşte yazılımın parçalarının bu ortamlar arasında, ortam kullanıcılarını kesintiye uğratmaksızın, taşınması, oturmuş ve uygulanabilir bir yazılım değişiklik yönetim ve takip sistemi ile mümkündür. Bu sistem sayesinde bu işlemler yönetilebilir ve takip edilebilir bir sıfat kazanır ve riski oldukça azaltır. Bu bildiride, bu yazılım parçalarının ortamlar arasında hatasız ve sıfır risk ile geçişini sağlayan, olası bir hata durumunda geçmişe dönük geçişlerin ve değişikliklerin raporlanabildiği bir sistem anlatılacaktır. Bu sistemi birebir uygulayan 300 şubeli bir katılım bankasındaki uygulama örneği detaylı bir şekilde tarif edilecektir.



Şekil 1. Yazılım Geliştirme Yaşam Döngüsü

Bildirinin geri kalanında ilk olarak YGYD için gerekli ortamların detayları, daha sonra önerilen yazılım geçiş ve değişiklik yönetiminin nasıl olması gerektiği anlatılacaktır. En son olarak ise önerilen yöntemi kullananan örnek bir uygulamanın ekran ve fonksiyonları ifade edilecektir.

2. YGYD Ortamları

Kritik finansal sistemlerde yazılım süreçlerinin en temel araçlarından birisi olan Geliştirme (Development), Test, Canlı Öncesi (Pre-Production) ve Canlı (Production) ortamlarının sağlıklı bir yazılım geliştirme yaşam döngüsü için oluşturulması hayati önem taşımaktadır. Bu ortamların her birisi, istemci uygulamasını, uygulama sunucusu uygulamasını, veritabanı uygulamalarını ve ortama ait uygulama kodlarını içermelidir.

Geliştirme ortamı, yazılımın geliştirilmeye başlandığı ortamdır. Sisteme eklenecek yeni yapılar ve sistemde gerçekleştirilecek değişiklikler yazılım geliştiriciler tarafından

bu ortamda gerçekleştirilirler. Yazılım mühendisleri ve sistem analistleri uygulamaları bu ortamda test eder, ilgili kapsama uygun olana kadar bulguları giderirler.

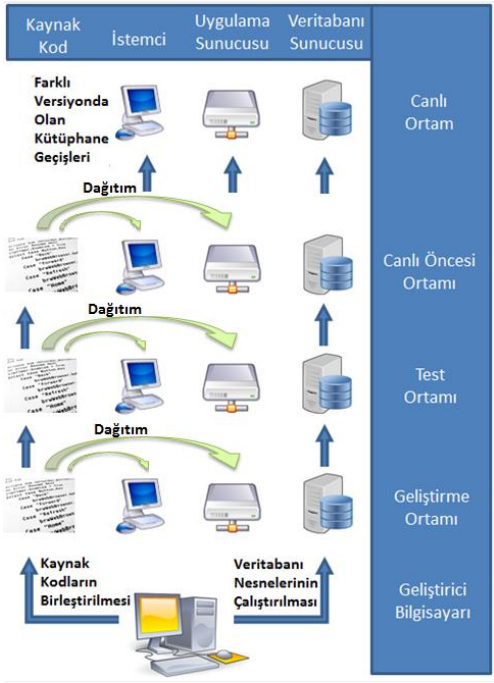
Geliştirme ortamında sistem analisti ve yazılım mühendisinden onay alan uygulama modülleri, test edilmeye hazır oldukları için geliştirme ortamından test ortamına aktarılırlar. Test mühendisleri ve analistler bu ortamda testlerini gerçekleştirirler. Sistemde tespit edilen yalnışlık veya eksiklikler, yazılım geliştiricilere bildirilir. Bulgu düzeltme işlemi geliştirme ortamında yapılmalı, gerekli testlerden sorna test ortamına aktarılmalıdır. Test ortamı kodlarına direk müdahele edilerek hiçbir değişiklik gerçekleştirilmemelidir.

Herhangi bir kontrolden geçirilmeden test ortamına aktarılan yazılım parçaları, canlı öncesi ortamına aktarılmadan önce gerekli görülen bir onay mekanizmasına tabi tutulurlar. Canlı ortam öncesinde, son kullanıcı tarafından, gerçek veriler ile neredeyse birebir aynı olan veriler kullanılarak son kontroller gerçekleştirilir.

Son kullanıcı testinden başarılı bir şekilde geçen yazılım parçaları, canlı ortama yani yazılımların kullanılmaya başlandığı ortama aktarılırlar. Bu ortamda meydana gelebilecek hatalar, sistemsel kesintilere sebep olma riski taşıdığı için büyük önem arz etmektedir.

Dağıtım, geliştirilen yazılım parçalarının ortamlara taşınması işlemidir. Bir şirketin yazılım geliştirme ortamları Şekil 2' de gösterilmiştir. Ortamlar arası geçişleri kod geçişi olarak yönetmek gerekmektedir. Geliştirme ortamında test edilen uygulamaların kodları, test ortamı kaynak kodu içerisine taşınmalıdır. Test ortamına taşınan kod derleme ve dağıtım sonrasında istemci ve uygulama sunucusuna gönderilmelidir. Uygulama kodları ile birlikte veritabanı kodları da geliştirme ortamından test ortamına benzer şekilde aktarılmalı ve veritabanı sisteminde hazır halde bulunan tablo, saklı yordam (stored procedure) ve fonksiyonlara dönüştürülmelidir. Burada yapılan işlemin aynısı canlı öncesi sisteme de geçiş esnasında da uygulanmalıdır.

Canlı ortama geçilirken uygulanacak yöntem ise veritabanı nesne ve fonksiyonları adına aynı, uygulama tarafında farklılaşmaktadır. Canlı ortam geçişlerinde diğer ortamlardan farklı olarak canlı öncesinde çalışan sınıf kütüphaneleri canlı ortama değişiklik ve versiyon kontrolleri ile aktarılmalıdır.



Şekil 2. Yazılım geliştirme ortamları

Sektörde dağıtım çözümleri var olan uygulamalar kullanılabileceği gibi, bu işlemi manuel olarak da gerçekleştirmek mümkündür. Ancak büyük sistemlere sahip olan kurum ve kuruluşlar bu denli karmaşık bir geçişi bir uygulama aracılığı ile gerçekleştirmek ve takip etmek zorundadırlar. Bu bildiride, bu süreci yönetebilecek ve riski minimize edebilecek bir sistem önerisi yapılacak ve örnek bir uygulaması aktarılacaktır.

3. Önerilen Yazılım Değişiklik ve Takip Yönetimi Sistemi (YDTYS)

Bir yazılımın geliştirilme sürecindeki en önemli olay, bu sürecin safhalarının bir düzen içerisinde ilerlemesini sağlamaktır. Geliştirme safhası tamamlandıktan sonra, geliştirilen yazılıma ait parçaların ortamlar arası geçişinin gerçekleştirilmesi gerekmektedir. Bu geçiş, yazılım parçalarının ortamlar arası taşınmasından ibarettir. Bu sistemin karmaşıklığından ötürü, sistemin takip

edilmesi için bir uygulamanın kullanılması kaçınılmaz bir ihtiyaçtır. YDTYS sistemin tam da bu noktasında devreye girmekte ve bundan sonraki sürecin kusursuz bir şekilde ilerlemesini sağlamaktadır. Bu sistem, temelde yazılım parçalarının YGYD ortamları arasında taşınması işlemini yerine getirmektedir.

Geliştirmeler belli talepler doğrultusunda gerçekleştirilmektedir. Dolayısıyla, gelişimini tamamlamış yazılım nesneleri, belli bir değişiklik talebi ile ilişkilendirilmiş olan bir pakete dönüştürülebilmelidir. Oluşturulan bu paketin ortamlar arası geçişi, gerekli uzman ve yöneticilerin de içinde bulunduğu bir onay mekanizması ile kontrol altına alınabilmelidir. Geçmişe dönük bu onaylar ve paket içerikleri raporlanabilmeli, gerektiğinde müdahale edilebilmelidir. Canlı ortama uygulama geçişi yetkilendirmelere tabi olması nedeni ile akışı düzgün bir şekilde tamamlayan paketler sadece bir uygulama servisi aracılığıyla gerekli görülen canlı ortama aktarılabilmelidir. Bu süreçte, paketin izlemiş olduğu akış diyagramı, ortamlar arası geçişin zamanlanması işlemleri de YDTYS' nin sahip olması gereken önemli özellikler arasında yer almaktadır.

3.1 Paket

Ortak bir yazılımın geliştirilmesi için en önemli araçlardan birisi Kaynak Kod Havuzu (KKH) uygulamasıdır. Bu uygulama açık kaynak kodlu veya profesyonel ürün ailelerinden seçilebilir. Olması gereken özellikler şu şekilde sıralanmalıdır; kurum içerisindeki bütün yazılım mühendisleri, sistem analistleri ve diğer gerekli rollere kod paylaşımı yapabilmelidir. Kod değişiklik geçmişini tutabilmelidir. Kişi ve dosya bazlı kilit sistemi en temel ihtiyaçlardandır. Ortam kodlarını tutabilmek için ilişkili ortam kodu (branch) oluşturma ve ortamlar arası geçiş için kod birleştirme (merge) işlemi yapılabilmelidir. Bu araç yardımı ile yazılımın kaynak kodları tek bir yerde tutulabilmekte ve yazılım geliştiricilerinin ortak dosyalar üzerinde geliştirmelerini yapmaları sağlanabilmektedir. Bu kaynak kodlarının her bir parçası uygulama nesnesi olarak adlandırılmaktadır. Bu nesneler paketimizin bir parçasını oluşturmakta ve kaynak kodlarının ortamlar arası taşınması sağlamaktadır. KKH'nın sunduğu ortak alanda bulunan kaynak kodlarına yapılan her müdahale bir değişim kümesi oluşmasına sebep olmaktadır. Her bir küme, yazılıma ait bir kod parçasının değişimlerini paket halde tutmaktadır. Gerekli görüldüğünde istenilen kod parçasının son halinin veya istenilen özel bir versiyonunun ortamlar arası taşınması da YDTYS ile sağlanabilmektedir. Yazılımın kod parçalarının değişim kümeleri YDTYS arayüzünde listelenmekte ve istenilen değişim kümeleri pakete eklenebilmektedir.

Saklı yordam, tablo, fonksiyon gibi veritabanı nesnelerinin pakete eklenebilmesi, kullanımı kolay bir arayüz ile sağlanabilmektedir. Herhangi bir veritabanı nesnesi grubuna girmeyen veritabanı sorguları da, güncelleme nesnesi olarak pakete eklenebilmektedir. Bu nesneler pakete eklendikten sonra, veritabanında gerçekleştirilen değişikliklerden (nesneler güncellenmediği sürece) hiçbir şekilde etkilenmezler.

Onaya gönderilen paketlerin onay anı ile değişiklik anı çakışmasını önlemek amacıyla paketlerde kilit mekanizması kullanılmaktadır. Bu mekanizma YDTYS üzerinde takip edildiğinden, veritabanından bağımsız çalışmaktadır. Yani pakete eklenen bir nesneye veritabanı üzerinden müdahale edilmesi durumunda, yapılan değişiklikler paket içerisindeki nesneye yansımamaktadır. Bir pakette bulunan nesnenin bir başka pakette kullanılmamak üzere kilitlenmesinin en temel sebebi, iki pakette de bulunabilecek aynı nesnenin ortamlar arasında taşınmaya çalışılması durumunda meydana gelebilecek problemlere engel olmaktır. Bir paket canlı ortama taşınana kadar belirli bir zamana ihtiyaç duyduğu için paket sahibinin ihtiyaç olduğu anda yerinde olmaması durumunun doğurabileceği sıkıntılar göz önüne alınarak, paketlere paylaşma özelliği eklenmiştir. Yani bir paketin birden fazla kullanıcı tarafından yönetilebilmesi sağlanmıştır. Tabi ki pakete aynı anda müdahale edilmesi durumuna karşılık gerekli önlemlerde alınmıştır.

Sistemde değişiklik taleplerinin tutulduğu otomasyonlar bu sistemden farklı düşünülmelidir. Bu sistem sadece yazılım değişiminin içeriğini barındırmaktadır. Değişiklik talebi sistemine ait takip numarasının, ilgili değişiklik paketi ile ilişkilendirilmesi gerekmektedir. Sistemler

normal şartlar altında çalışmak üzere dizayn edilirler. Süreçlerin elimizi kolumuzu bağlamaması için acil durumlarda, istisna yönetebilmek için bazı özellikler eklenmelidir. Bu istisna zamanlarda kullanabilmek üzere otomasyon sistemine acil geçiş özelliği eklenmek sureti ile bazı paketler ortamlar arası geçiş yapmaksızın en temel ihtiyaç olan noktalara uğrayarak canlı ortama geçebilmelidir. Bu özellik, özel durumlarda paketin ortamlar arası geçiş sürecini hızlandırmak amacıyla eklenmiştir. Bu durumda paketin sadece bir ortam için gerekli onayları alması yeterlidir.

Veritabanı nesnelerinin oluşturulmasının belli bir düzende olması gerekmektedir. Önerilen YDTYS sistemi bu düzene uyulup uyulmadığını otomatik olarak denetmeyebilmelidir. Örneğin, bir tablo yapısı oluşturulmadan o tablo ile alakalı bir fonksiyonun oluşturulmaya çalışılması hata ile sonuçlanacaktır. Bu yüzden YDTYS paketlerinin yapısında çalışma sırası özelliği kullanılmakta, veritabanı nesnelerinin çalışma sırası paket hazırlanırken belirlenebilmelidir.

3.2 Roller

Paket geçişlerinin ortamlar arasında geçişinin yönetilmesi için olması gereken belli başlı roller: Yazılım Mühendisi, Servis Yöneticisi, Konfigurasyon&Sürüm Uzmanı ve Veritabanı Uzmanı'dır.

Yazılım mühendisi, yazılım değişiklik paketinin oluşturulması, ortamlar arasında geçişin başlatılması, geçişin canlı ortama kadar takip edilmesi, değişiklik olması durumunda süreci tekrar baştan başlatabilmek işlemlerini gerçekleştirebilmelidir. Paketin geçişleri esnasındaki onay ve kontroller ise ilgili yetkili kullanıcılar tarafından gerçekleştirilmelidir.

Kullanıcılar, sahip oldukları yetkilere göre bir role atanmalıdırlar. Yetkilendirmeler ise projenin ihtiyaçlarına ve sistemin ilerleyişindeki sapmalara göre belirlenmelidir. Örneğin, bir YDTYS paketinin içerisindeki uygulama ve veritabanı katmanına ait nesneler bir yöneticinin kontrolünden geçmelidir. Bu gereklilik doğrultusunda uygulama nesnesi onaylama yetkisi ve veritabanı nesnesi onaylama yetkisi olmak üzere iki temel yetki belirlenmelidir. Bu yetki ve onaylama işlemleri aşağıdaki başlık içerisinde detaylandırılacaktır.

3.3 Değişiklik Paketi Ortamlararası Geçiş Süreci

Ortamlar arasında geçiş yapacak olan kod parçaları değişiklik paketi içerisine alınarak, tanımlanan roller aracılığıyla ortamlar arasında bir akış doğrultusunda taşınmalıdır. Bu akışın şeması, Şekil 3' de gösterilmiştir. Yazılımı değişiklik paketi geliştirme ortamında bulunan veritabanı ve uygulama kod değişimlerinin yazılım mühendisi tarafından seçilmesi sonucunda oluşturulmalı, değişiklik talep numarası ile ilişkilendirilmeli ve kaydedilmelidir. Paketin her durumunun tanımlayıcı bir isminin olması nedeniyle, oluşturulan bu paketin ilk durumu "Başlangıç" olarak isimlendirilmelidir. Oluşturulan paket ilk olarak test ortamına gönderilmeli, bu süreçte herhangi bir onay söz konusu olmamalıdır. Çünkü, test ortamı için onay istenmemesi test mühendisi ile yazılım mühendisi arasındaki etkileşimi oldukça hızlandıracak, prosedürlerden uzaklaştıracaktır. Paketin test ortamına gönderilmesi başarısız olursa, paketin akış durumunda bir değişiklik olmamalı, başarılı olması durumunda da paketin son durumu "Test" olarak güncellenmelidir. Test ortamındaki kontrollerin yapılması aşamasında, gerekli görüldüğü yerlerde paket içerisindeki nesnelere müdahale edilebilmelidir. Test ortamında iken yapılacak değişikliklerin test ortamına yansıyabilmesi için, paketin durumu tekrar başlangıç olarak güncellenmeli ve yeniden test ortamına gönderilmelidir.

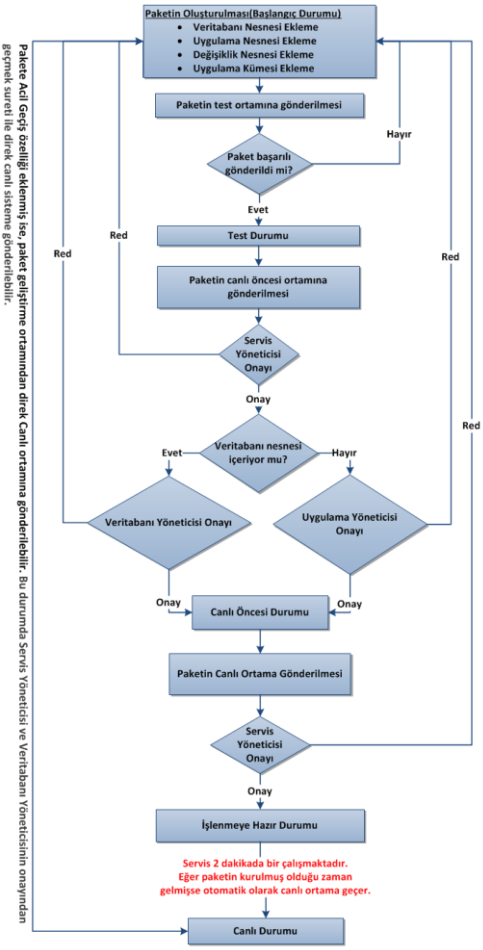
Test ortamındaki testleri başarılı bir şekilde tamamlanmış bir değişiklik paketi bu adımdan sonra canlı öncesi ortamına gönderilmeli, paketin durumunun da "Canlı Öncesi Ortamına Gönderildi" olarak güncellenmelidir. Bu durumda paket servis yöneticisinin onayını beklemeli, onaylandıktan sonra içerisindeki nesnelerin yapısına göre yeni durumu belirlenmelidir.

Eğer paket veritabanı nesnesi içeriyorsa, ilk önce veritabanı yöneticisinin onayından geçmelidir. Bu durumda paketin yeni durumu da "Canlı Öncesi için Veritabanı Yöneticisine Gönderildi" olarak güncellenmelidir. Paketin

içerisindeki veritabanı nesneleri, veritabanı yöneticisi tarafından incelenmeli, standartlara aykırı bir durum var ise bu durum belirtilmeli ve paket reddedilmelidir. Eğer paket uygulama nesnesi içeriyorsa uygulama yöneticisi tarafından onaylanmak üzere, paketin yeni durumu "Canlı Öncesi için Uygulama Yöneticisine Gönderildi" olarak güncellenmelidir. Uygulama yöneticisi, paketin içerisinde bulunan uygulama nesnelerine birleştirme işlemi uygulayarak canlı öncesi ortamındaki nesneler ile güncellemelidir. Böylelikle gerekli bütün onayları almış olan değişiklik paketinin yeni durumu "Canlı Öncesi" olmalıdır. Bu süreçte herhangi bir yönetici paketi reddederse, paket başlangıç durumuna geri dönmelidir.

Canlı öncesi durumundaki bir paket yazılım mühendisi tarafından son ve gerçek ortam olan canlı ortama gönderilmeli, paketin durumu da "Canlı Ortama Gönderildi" olarak güncellenmelidir. Veritabanı nesnesi içeren paketler, test ortamından canlı öncesi ortamına geçişte veritabanı yöneticisinin onayından geçtiği için, burda tekrar onaya tabi tutulmamalıdır. Servis yöneticisinin ve eğer pakette uygulama nesnesi var ise bir uygulama yöneticisinin onayından da geçen paketlerin durumu "İşlenmeye Hazır" olarak güncellenmeli ve canla ortama geçmeye hazır hale gelmiş olmalıdırlar. Canlı ortam gerçek ve sürekli çalışmakta olan bir sistemi barındırmaktadır. Çalışan bir sisteme müdahale edilemeyeceğinden dolayı, bu ortama geçişler belli zaman aralıklarıyla gerçekleştirilmelidirler. Bu yüzden paketlere "Zamanlama" özelliği adı altında dağıtım olma zamanı tanımlanmalıdır. Gerçek ortamda çalışmakta olan bir uygulama servisi, paketleri iki dakikada bir kontrol etmeli ve zamanı gelen paketlerin gerçek ortama aktarılmasını sağlamalıdır.

Paket acil geçiş özelliğine sahip ise, daha önce de açıklandığı üzere test ve canlı öncesi ortamlardan geçmesine gerek kalmaksızın canlı ortamına direk gönderilebilir. Bu durumda servis yöneticisi ve veritabanı uzmanının onayından geçmek sureti ile direk canlı sisteme gönderilebilir. Veritabanı uzmanı onayına gönderilen bu paketler "Acil Geçiş Nesnesi Olarak Veritabanı Uzmanına Gönderildi" olarak güncellenmelidir. Veritabanı uzmanının paketi onaylaması ve fiziksel olarak canlı ortamda paket nesnelerini çalıştırması sonucunda bu süreç tamamlanmış olur.



Şekil 3. Bir YDTYS paketinin ortamlar arası geçişini gösteren akış şeması

Bu süreçler içerisinde gerçekleşen her türlü onay ve red işlemlerinde kullanıcılar, e-posta yolu ile bilgilendirilmelidir.

3.4 Servis

Bilgi teknolojileri sektöründe gerçekleştirilen her projede veri güvenliği üst seviyede tutulmalıdır. Kullanıcılar tarafından yapılacak işlemlerde, bir yetkilendirme mekanizması ile kullanıcıların yetkileri dışındaki verilerle işlem yapmalarının kısıtlanması çok önemli bir unsurdur. Bu yüzden tam erişim yetkisine sahip bir servis uygulaması aracılığı ile, yapılacak işlemler kontrol edilebilmeli,

kullanıcıların yetkileri dışındaki işlemler iptal edilmeli ve konu ile ilgili birimler e-posta yolu ile bilgilendirilebilmelidir. Ayrıca bu servis uygulaması yapılan her türlü işlemi kayıt altına alabilmeli ve gerektiğinde kullanıcı bazlı bir raporlayabilmelidir. Canlı ortamına geçiş işlemlerinde bir güvenlik duvarı görevi gören bu servis uygulaması, onay mekanizmasını başarılı bir şekilde tamamlayan YDTYS paketlerinin canlı ortama aktarılmalarını da kontrol altına almalıdır.

3.5 Yönetim

Sistemin işleyebilmesi için kullanıcıların tanımlanması, rollerin oluşturulması, ilgili kullanıcılara belirlenen rollerin atanması gibi bir takım tanımlamaların yapılması gerekmektedir. Bu tanımlamalar sadece yetkili olan kullanıcılar tarafından gerçekleştirilebilmelidir.

Genel anlamda bahsedilen 4 farklı ortam uygulama bazlı değişebilmekte, dolayısı ile uygulama sunucusu ve veritabanı sunucusu tanım tabanlı olarak yönetilmesi gerekmektedir. Bu yüzden istenilen ortama, veritabanı lokasyonu ve uygulama lokasyonu olmak üzere iki tip lokasyon tanımı eklenebilmelidir.

Sistemi kullanacak olan kullanıcılar sisteme eklenmeli ve gerekli roller ile ilişkilendirilmelidirler. Rol bazlı yetkilendirmeler ile kullanıcıların işlemlere yetki verilebilmesi veya alınabilmesi gereklidir.

Bütün bu tanımlama ve yetkilendirme işlemlerinin yapılabilmesi için konfigurasyon&sürüm uzmanlarına "Yönetim" rolü de verilmelidir.

3.6 Raporlar

Olası bir hata veya denetim durumunda geçiş yapılan yazılım değişikliklerinin incelenmesi için gerekli raporlar sistemde geliştirilmelidir. Gerek duyulacak belli başlı raporlar: Paket Geçmişi Raporu, Veritabanı Nesnesi Geçiş Geçmişi, Uygulama Nesnesi Geçiş Geçmişi, Paketler İçerisinde Detaylı Metin Araması olarak listelenebilir.

Paket Geçmiş Raporu, belli bir tarih aralığı ve/veya yazılım mühendisi kullanıcı adı filtreleri verilmek sureti ile geçilen paketlerin listelenmesini sağlayan rapordur. Rapor, liste içerisinden seçilen bir paketin uygulama parçaları listesini gösterebilmeli ve paketin hangi ortamlar arasında, kimlerin onayı ile geçiş yaptığını listeleyebilmelidir.

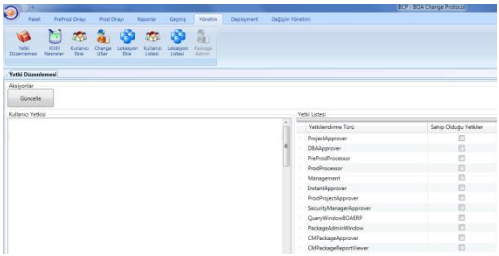
Veritabanı Nesnesi Geçmiş Raporu, bir veritabanı nesnesinin hangi paket veya paketler içerisinde olduğunu tespit edebilmek amacıyla, nesnenin ismi ve/veya tarih aralığı filtrelerini baz alarak nesnelerin listelenmesini sağlayan rapordur. Bu raporda, aranan nesnenin ait olduğu paketin sahibi, paketin oluşturulma tarihi, nesnenin ait olduğu veritabanı şeması, nesnenin tipi (tetikleyici, saklı yordam, tablo vs.) gibi bilgiler listelenebilmeli, seçilen nesneye ait içerik gösterilebilmelidir.

Uygulama Nesnesi Geçmiş Raporu ile uygulama nesnesi içeren paketler, nesnenin ismi, uygulamanın bulunduğu dosya yolu, nesnenin ait olduğu yazılım değişiklik talebine ait takip numarası ve/veya tarih aralığı kriterleri kullanılarak listelenebilmektedir. Sonuç listesinde paketin sahibi, oluşturulma tarihi, nesnenin ismi, uygulamanın bulunduğu dizin gibi bilgiler listelenebilmelidir.

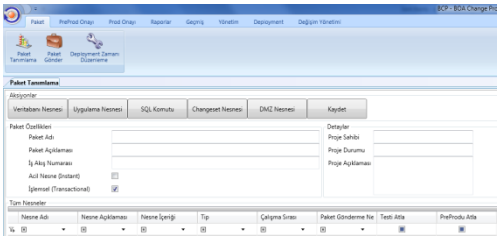
Paketler İçerisindeki Detaylı Metin Arama Raporu, tarih aralığı, paketin sahibi, nesnenin adı, takip numarası kriterleri ile birlikte, pakete ait herhangi bir bilginin (paketin ismi, nesnenin içeriği, paketin sahibi vs.) içerisinde geçen bir metin ile de arama yapılabilmektedir. Verilen kriterlere uygun bütün paketler listelenebilmeli, pakete ait bütün nesneler yine bir liste şeklinde gösterilebilmeli ve seçilen herhangi bir nesnenin içeriği incelenebilmelidir.

4. Sonuç

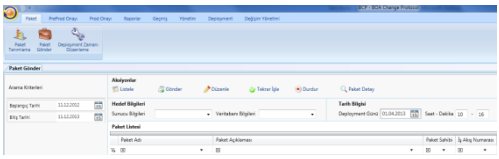
Önerilen sistemin uygulanması ile oluşan uygulama 300 şubeli bir Katılım Bankasında kullanılmış ve sıfır riske yakın, yönetilebilir, takip edilebilir sistem geçişleri otomatize edilebilmiştir. Uygulamanın örnek resimleri aşağıdaki gibidir.



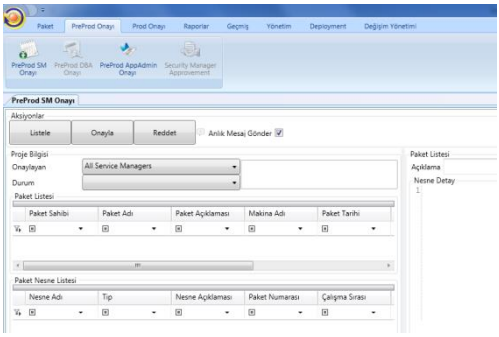
Şekil 4. Kullanıcı yetkilendirme ekran görüntüsü



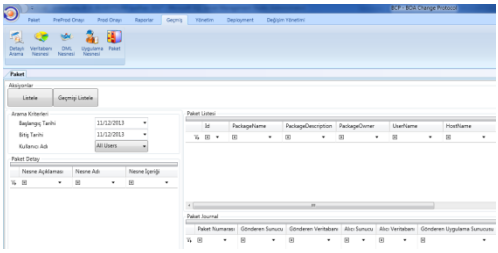
Şekil 5. Paket oluşturma ekran görüntüsü



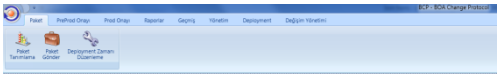
Şekil 6. Paket gönderme ekran görüntüsü



Şekil 7. Paket onaylama ekran görüntüsü



Şekil 8. Paket geçmişi ekran görüntüsü



Şekil 9. Ana menü ekran görüntüsü

5. Kaynaklar

[1] Yasuhiro S., "Producing and Managing Software Objects in the Process Programming Environment OPM", **Institute of Electrical and Electronics Engineers**, Japan, 270-274 (1994).

[2] Leon O., "Software Environment Research: Directions fort he Next Five Years", **Institute of Electrical and Electronics Engineers**, 35-42, (1981).

[3] Jianli D., Jianzhou W., Donghuai S., Haiyan L., "The Research of Software Product Line Engineering Process and Its Integrated Development Environment Model", **Institute of Electrical and Electronics Engineers**, 66-70, (2008).

[4] Manoj S., Venkateswara R M., Manish G., Pradipta D., "Tracking Configuration Changes Proactively in Large IT Environments", **Institute of Electrical and Electronics Engineers**, 522-524, (2012).

[5] Syed A. F., Ho-Jin C., "Life Cycles for Component-Based Software Development", **Institute of Electrical and Electronics Engineers**, 637-639, (2008).

[6] http://msdn.microsoft.com/enus/library/cc296714(v=bts.10).aspx

[7] http://en.wikipedia.org/wiki/Development,_testing,_acc eptance_and_production

[8] http://www.networkworld.com/newsletters/techexec/2012/050412bestpractices.html