Coğrafi Dağıtık Yazılım Geliştirme Ortamında Yazılım Konfigürasyon Yönetimi

Hayrullah KALE[1] R. Bülent GÖKALP[2]

[1,2]Barış Kartalı Projesi, HAVELSAN A.Ş. ANKARA

[1]e-posta: hkale@havelsan.com.tr [2]e-posta: bgokalp@havelsan.com.tr

Özet

Günümüz iş dünyasında yazılım geliştirme ortamlarının daha esnek olması zorunluluk haline gelmiştir. Coğrafi olarak birbirinden uzak grupların birlikte yazılım geliştirmeleri bunun bir örneğidir. Bu bildiride yazılım konfigürasyon yönetimininin coğrafi dağıtık yazılım geliştirmeyi nasıl desteklemesi gerektiği, Barış Kartalı Projesi'nde edinilen tecrübeler ışığında, sunulmaktadır.

Abstract

The complexity of today's enterprise software development requires a flexible approach for the development environment. The need of close cooperation among geographically distributed sites during the development is an example to this requirement. In this paper, we will show how the software configuration management should support a distributed software development environment using the experience gained during the Peace Eagle Project.

1. Giriş

HAVELSAN, BARIŞ KARTALI (BK) Havadan Erken İhbar Komuta ve Kontrol Uçağı'nın temininde, Görev Yazılımları ve Yer Destek Sistemleri konusunda, yerli alt yüklenici sorumluluğunu üstlenmiştir. HAVELSAN, uçak ve yer destek sistemleri için geliştirilecek Türk Hava Kuvvetleri'ne özel yazılımların geliştirilmesinde, modifikasyonunda ve entegrasyonunda görev aldığı gibi, aynı zamanda bu yazılımların testini ve BOEING tarafından temin edilen 737 AEW&C ana sistem yazılımına entegrasyonunu da gerçekleştirecektir.

HAVELSAN sistem çözümleme sürecinden başlayarak test ve değerlendirmeye kadar projenin tüm mühendislik süreçlerinde yer almaktadır. HAVELSAN, proje kapsamında toplam olarak yaklaşık 200.000 satır kaynak kod boyutunda, 12 yazılım modülünü (yazılım parçacığı) geliştirmiş/modifiye etmiş olacaktır.

Görev bilgisayarı yazılımı kapsamında BOEING'in ve HAVELSAN'ın yazılım gerçekleştirme sorumlulukları bulunmaktadır. Bu kapsamda mevcut kod üzerinde iki taraf da modifikasyon

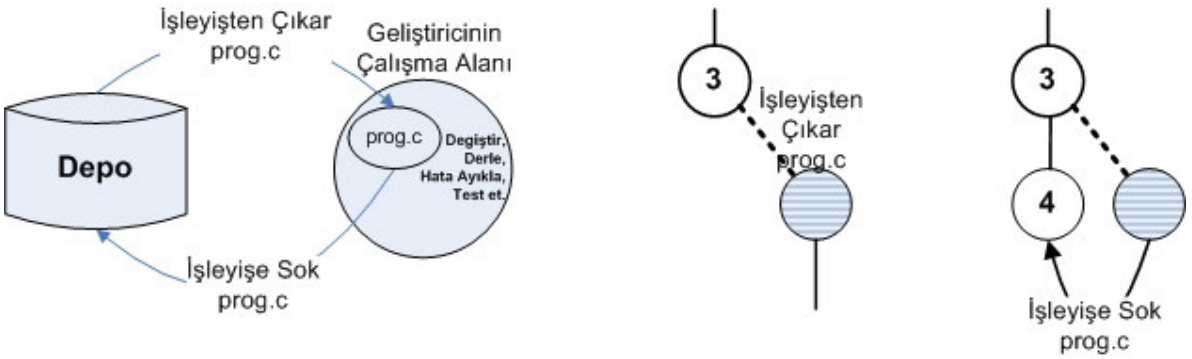
yapmaktadır. Bunun dışında HAVELSAN yazılımın bir kısmını kendisi tasarlayıp geliştirmektedir. BOEING'in yazılım ekibi Seattle'da, HAVELSAN'ın yazılım ekibi ise Ankara'da bulunmaktadır.

Yukarıda bahsedilenler ışığında bakıldığında, BK Projesi yazılım geliştirme faaliyetlerinin etkin bir Yazılım Konfigürasyon Yönetimi ile desteklenmesi gerekmektedir. Aşağıdaki paragraflarda bu kapsamda uygulanan model ve süreçlerin bir kısmı anlatılmaktadır.

2. Yazılım Geliştirme

2.1 İşleyişten Çıkar (Checkout)/ İşleyişe Sok (Checkin) Modeli

Genelde sürüm yönetimi araçları **İşleyişten Çıkar-Degiştir-İşleyişe Sok** modeliyle çalışmaktadır. Başlangıç durumda dosya çalışma alanında salt okunabilir halde bulunur. Geliştirici bir dosyada değişiklik yapmak isterse o dosyayı işleyişten çıkarır, bu durumda dosya çalışma alanında yazılabilir hale gelir. Değişiklikler yapıldıktan sonra dosya işleyişe sokulur. Bu durumda dosyanın yeni bir sürümü oluşur[1]. Bu durum Şekil 1'de gösterilmektedir.



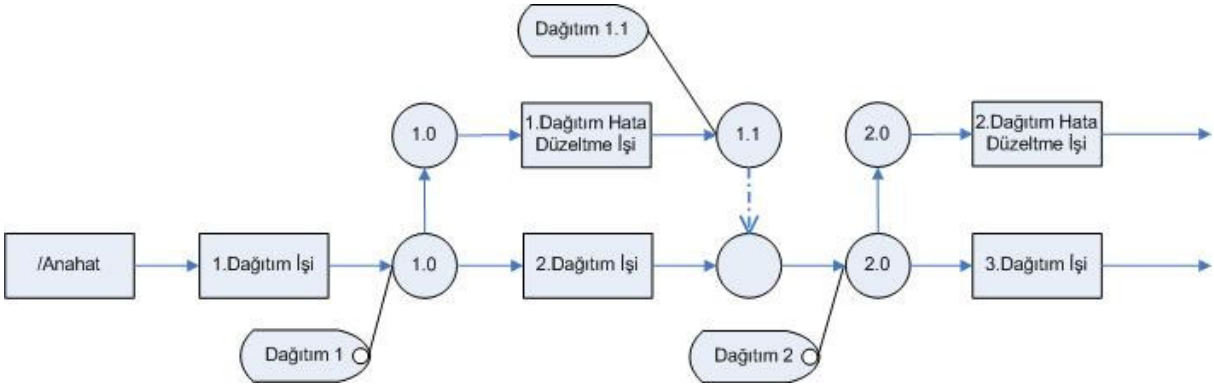
Şekil 1. İşleyişten Çıkar/İşleyişe Sok Modeli

BK projesinde Rational ClearCase konfigürasyon yönetim aracı ve Rational ClearQuest değişiklik yönetim araçları tümleşik olarak kullanılmaktadır. Bir dosyanın işleyişe sokulabilmesi için mutlaka yazılım değişiklik süreci kapsamında onaylanmış bir Yazılım Değişiklik İsteği (YDİ) ile ilişkilendirilmesi gerekmektedir (bu durum bir betik ile zorunlu hale getirilmiştir). Bunun anlamı;

"Değişiklik yapmak isteniyorsa YDİ yazıp Yazılım Konfigürasyon Kontrol Kurulu'ndan (YKKK) onay almak zorunludur, aksi halde değişiklik yapılamaz."

2.2 Yan Dal Kullanımı ile Eş Zamanlı Yazılım Geliştirme

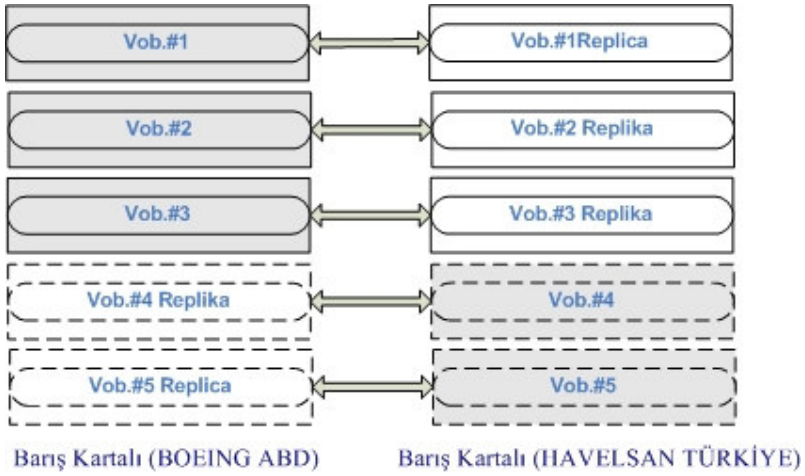
Bu projede -bazı durumlarda- birden fazla kullanıcının aynı dosya üzerinde aynı anda değişiklik yapması gerekmektedir. Örneğin bir grup sonraki dağıtım için gerekli kodlama çalışmalarını yürütürken başka bir grup dağıtımı yapılan sürümdeki hataları gidermek için gerekli kodlama işini yürütmek durumunda kalmaktadır[2]. Bu durum için Şekil 2'deki örüntü kullanılmaktadır.



Şekil 2. Yan Dal ile Eş Zamanlı Yazılım Geliştirme

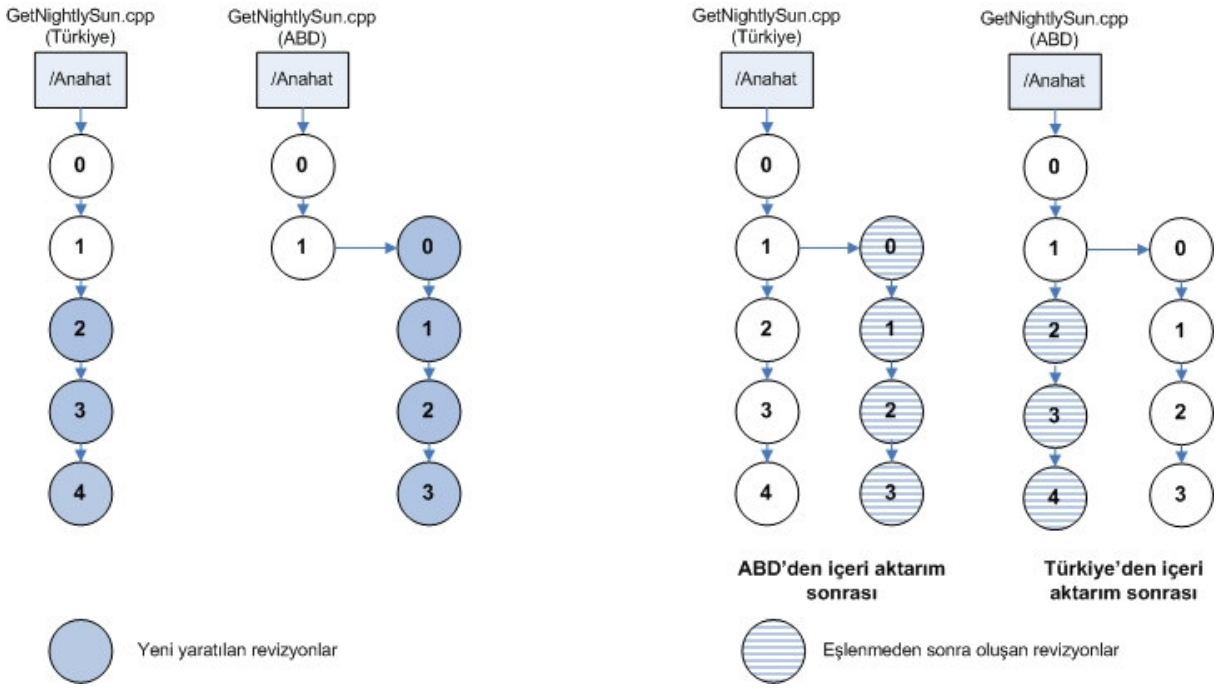
2.3 Coğrafi Dağıtık Yazılım Geliştirme

Yazılım Konfigürasyon Yönetimi aracı olarak ClearCase MultiSite kullanılmaktadır. BOEING'in sorumlu olduğu VOB'lar (ClearCase'in Veri Tabanı) HAVELSAN'a, HAVELSAN'ın sorumlu olduğu VOB'lar ise BOEING'e çoklanmıştır. Şekil-3'te bu durum gösterilmiştir.



Şekil 3. HAVELSAN ve BOEING arasındaki Coğrafi Dağıtık yapı

İki taraf ayrı yan dallar üzerinde çalışmaktadır. VOB'un sahibi birleştirme sorumluluğunu yüklenmiş durumdadır. Sıklıkla VOB'ların eşlenmesi yapılmaktadır. Şekil 4'te coğrafi dağıtık yapıda geliştirme örüntüsü görülmektedir.



Şekil 4. Coğrafi dağıtık yapıda geliştirme örüntüsü

3. Yazılım Değişiklik Süreci

Proje ekibi ihtiyaç duyduğunda yazılımın değiştirilmesi için istekte bulunabilir. Bu istek BOEING ya da HAVELSAN ekibinden gelebilir. Bu istekler karşı tarafı etkilediği durumda, karşı tarafın karar sürecinde bulunması zorunluluğu vardır. Bu değişikliklerin belirli bir süreç içerisinde sonuçlandırılması ve izlenebilmesi için ClearQuest Multisite değişiklik yönetim aracı kullanılmaktadır.

3.1 Roller ve Sorumluluklar

Yazılım Konfigürasyon Yönetimi Sorumlusu (YKYS), yazılım geliştirme sürecine katkıda bulunan tüm kullanıcılara rol ve sorumluluk atar. Bu rol ve sorumluluklar Tablo 1.'deki gibidir.

Tablo 1. Rol ve Sorumluluklar

Rol Adı	Açıklama
İstek Sahibi	YDİ'yi Yazılım Değişiklik Süreci'nin tarif ettiği şekilde bildiren kişi.
Teknik Lider	Ürün ya da Ürün Grubu Sorumlusu olan ve çözümleyiciyi belirleyip atamasını yapan kullanıcı. YDİ ile ilgili süreç tamamlanıncaya kadar Teknik Liderin takibi ve sorumluluğu altındadır.
Çözümleyici	YDİ için çözümleme yapması istenen teknik sorumlu. Çözümleme sonunda düzeltici faaliyet mutlaka tanımlanmış olmalıdır.
YKKK Üyeleri	YDİ'leri çözümleme sonucunu dikkate alarak değerlendirip nihai kararı veren kuruldur. Bu kurulun asil üyelerinin kimler olacağı Yazılım Konfigürasyon Yönetimi Planı'nda (YKYP) yer almalıdır. BK projesinde BOEING temsilcisi kurulun asil üyesidir.
Test Mühendisi	Test faaliyetlerini yürütmekten sorumlu kullanıcı.
Yazılım Kalite Sorumlusu	Süreç kapsamında Yazılım Kalite Güvence gereklerinin uygulanmasından sorumlu kullanıcı.
Yazılım Konfigürasyon Yönetimi Sorumlusu	Bu süreç kapsamında, YKKK'ya idari ve teknik destek verir. ClearQuest'in yönetiminden ve raporlamadan sorumludur.

3.2 Öncelik ve Önem Tanımları

Değişiklik yönetimi süreci kapsamında kullanılan YDİ'lerin öncelik ve önem tanımları Tablo 2 ve Tablo 3'te verilmiştir.

Tablo 2. YDİ Öncelik Tanımları

Öncelik 1	Bir problem: 1. Bir gereksinimin karşılanması engelleniyorsa, 2. Kullanıcının faaliyetini engelliyorsa, 3. Personel güvenliğini etkiliyorsa Takip eden ara sürüm dağıtımında mutlaka yer almalıdır.Örneğin dağıtım 4.3'te, ana dağıtım 4 ara dağıtım 3'tür. Bu durumda problem 4.4'te mutlaka giderilmelidir.
Öncelik 2	Bir problem: Başarımı düşüren ve işlevsel olarak olumsuz bir etkisi oluyorsa ve çözüm bilinmiyorsa, takip eden ara sürüm dağıtımında mutlaka yer almalıdır.
Öncelik 3	Bir problem: Başarımı düşüren ve işlevsel olarak olumsuz bir etkisi oluyorsa ve çözüm biliniyorsa, takip eden ana sürüm dağıtımında mutlaka yer almalıdır. Örneğin dağıtım 4.3'te, ana dağıtım 4 ara dağıtım 3'tür. Bu durumda problem 5.0, 5.1, 5.2 'de yer alabilir.
Öncelik 4	İşlevsellik ya da başarım açısından bir sıkıntı yok ancak kullanımı zorlaştıran ya da kısıtlayan bir durum ise.
Öncelik 5	Diğer tüm problemler.

Tablo 3. YDİ Önem Tanımları

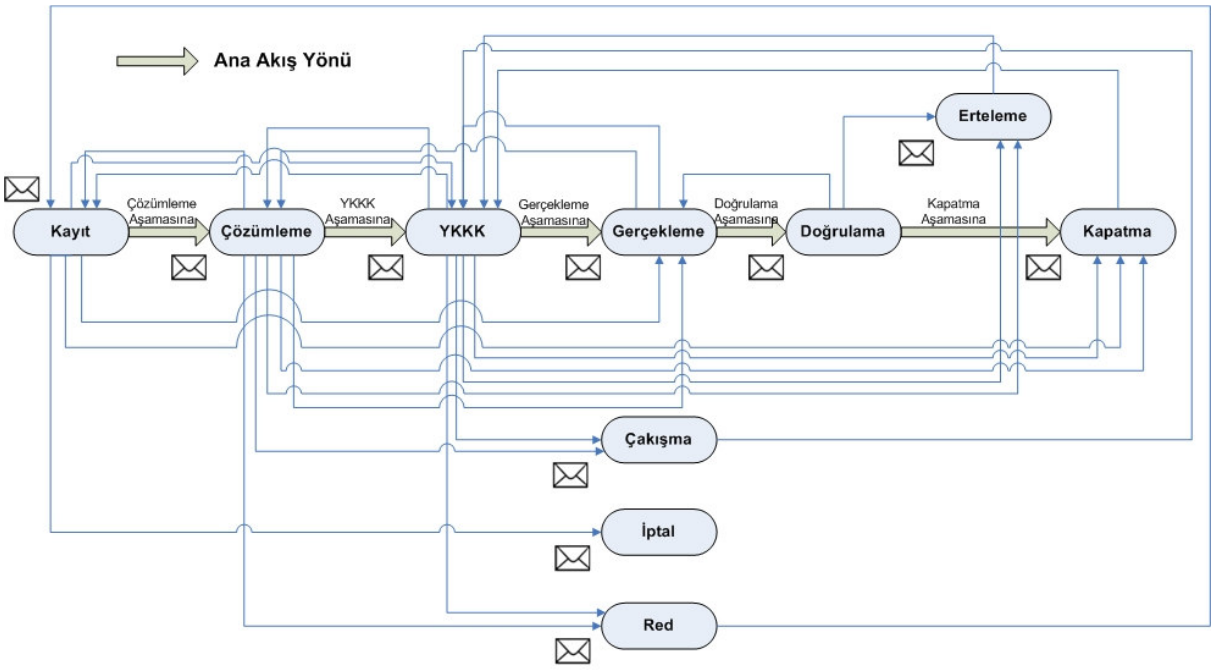
Önem 1	Kritik: Operasyonel ya da görev'e ilişkin yeteneğin kullanılması engelleniyorsa.
Önem 2	Gerekli: Performansı düşürüyor ve operasyonel ya da görev olarak negatif bir etkisi varsa.
Önem 3	Sakıncalı: Operasyonel ya da performans açısından bir sıkıntı yok ancak operatöre sıkıntı veren bir durum varsa.
Önem 4	Diğer

3.3 Yazılım Değişiklik İsteği

YDİ ClearQuest'te bir kayıt tipi olarak bulunur ve Yazılım Konfigürasyon Yönetimi (YKY) kontrolü altında bulunan yazılım ürünlerinin değişikliklerinin yönetilmesi için kullanılır.

3.3.1 YDİ Süreç Akış Tanımı

YDİ aşama modeli, YDİ'nin tüm yaşam döngüsünü yansıtan bir modeldir. Bu model 10 aşamadan oluşmaktadır. Şekil 5'te YDİ süreç akışı görülmektedir.



Şekil 5. YDİ Süreç Akışı

YDİ'nin aşama geçişi bir aşamadan diğerine şeklinde olmaktadır. Bu aşamaların açıklamaları Tablo 4'te verilmiştir. Her aşama geçişinde ilgili kişilere otomatik olarak e-posta gönderilmektedir.

Tablo 4. YDİ Akış Tanımları

Süreç Adımı:	Süreç Tanımı
Adım 0 – YDİ Yaratılması	Bir YDİ herhangi bir proje personeli tarafından başlatılabilir. Kullanıcı problemi detaylı şekilde girer. Bu bir test sonucu ise, testin seviyesi, altsistem, yazılım ve ilgili dokumanlar da girilmelidir. Ayrıntılar girildikten sonra YDİ giriş tuşuna basılarak YDİ veritabanina Kayıt aşamasında kaydedilmiş olur. Bununla birlikte Proje yöneticisine ve Teknik liderlere otomatik olarak e-posta gönderilir.
Adım 1 Kayıt Aşaması	Teknik lider YDİ'nin sorumluluk alanına girip girmediğine bakar. Kendi sorumluluğunda ise, Aşağıdaki hususları gözeterek YDİ'yi inceler: Problem tanımının yeterli olup olmadığına bakar. Önem seviyesini belirler. Öncelik seviyesini belirler. Daha önceki bir YDİ ile aynı olup olmadığına bakar. Bunun sonucunda aşağıdaki iki karardan birini verir: (1) YDİ'yi Çözümleme aşamasına alır ya da (2) YDİ'yi reddeder.
Adım 2 Çözümleme Aşaması	Atanmış olan Çözümleyici problem için bir düzeltici faaliyet belirler. Bu kapsamda YDİ'ye dokuman ekleyebilir. Çözümleme sonuçlanınca Teknik lider YDİ'nin aşamasını YKKK'ya getirir.
Adım 3 YKKK Aşaması	YKKK YDİ'nin gereğinin yapılmasına karar verebilir, bunun dışında çözümlemeyi yeterli bulmayıp ek çözümleme için geri gönderebilir ya da YDİ'yi reddedebilir. Son kararı verme sorumluluğu kurul başkanına aittir.Diğer üyeler başkana yardımcı olmakla yükümlüdürler. YKYS YDİ'yi Gerçekleme aşamasına getirir.
Adım 4 Gerçekleme Aşaması	Teknik Lider bu işi yapması için bir ya da birden fazla kişiyi görevlendirir. Düzeltici faaliyet tamamlandıktan sonra Teknik Lider YDİ'yi Doğrulama aşamasına getirir.
Adım 5 Doğrulama Aşaması	Çözümleyici, test sorumlusu, kalite sorumlusu ve YKYS'ninde katıldığı YKKK toplantısında doğrulaması yapılan YDİ'ler kapatılmaya hazırdır.
Adım 6 Erteleme Aşaması	Bazı durumlarda (Takvim problemi olmayan, özel test gerektiren v.b.) YDİ Erteleme aşamasında tutulur. Haftalık YKKK toplantılarında bu YDİ'lerde değişiklik gerekip gerekmediğine bakılır.
Adım 7 Kapatma Aşaması	Testi ve doğrulaması yapılan YDİ YKKK kararıyla YKYS tarafından Kapatma aşamasına getirilir.
Adım 8 Çakışma Aşaması	Teknik lider ya da YKKK YDİ'nin daha önceki bir YDİ ile aynı olduğuna karar vermişse bu aşamaya getirir. Daha sonra YKKK kararıyla YDİ İptal aşamasına getirilir.,
Adım 9 İptal Aşaması	YDİ Teknik Lider tarafından Kayıt aşamasından itibaren İptal aşamasına getirilebilir. Daha sonra bu YDİ Red aşamasına getirilir.
Adım 10 Red Aşaması	YDİ içerik olarak,gereksinim olarak ve başka nedenlerle uygun bulunmayabilir bu durumda Red aşamasına getirilir.

4. Gecelik İnşa Süreci

Görev Bilgisayarı Yazılımı, yazılım bileşen parçalarından oluşan sektörlere ayrılmıştır, her sektör ClearCase VOB'unda ya da VOB'larında bulunan belirli dizinlerden meydana gelmektedir. Her sektörün geliştirme takımı bağımsız olarak kodlama ve birim testi faaliyetlerini yürütür.

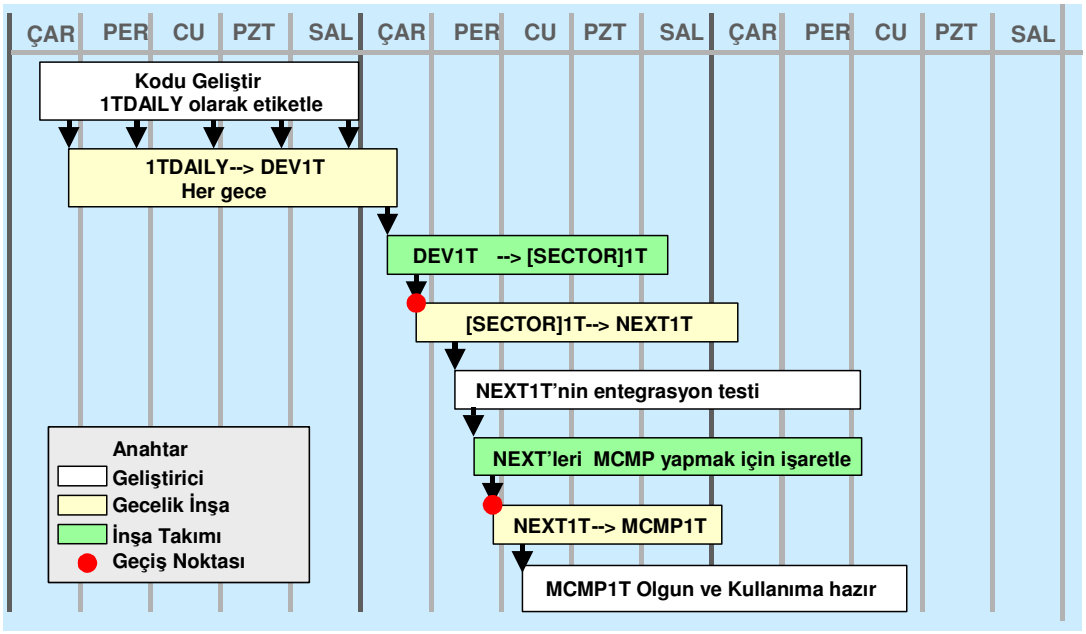
Yazılımın tamamının derlenmesi yaklaşık 12 saati bulmaktadır. Böyle bir durumda yazılımın tamamını derleyen bir gecelik inşa kaçınılmazdır.

Bir kaynak dosyanın gecelik inşaya girmesi isteniyorsa geliştirici tarafından o dosyanın istenen revizyonuna [BLD]DAILY formatında bir etiket eklenir, burada [BLD] program inşa numarasıdır.

Gecelik inşa bir betik ile başlatılır. Bu betik önce tüm [BLD]DAILY etiketi olan revizyonlara DEV[BLD] etiketini ekler. Her gecelik inşa bağımsız olarak belirli bir inşa görünümünde (ClearCase çalışma sahası) yapılır.

İnşa Takımı içerisindeki sektör sorumlusu, tümleşik inşa'ya (tüm sektörlerin bir arada derlenmesi) girecek dosyaların revizyonlarına bir betik yardımıyla [SECTOR][BLD] formatında bir label ekler.Entegrasyon inşa'ya başlamadan önce tüm [SECTOR][BLD] etiketi olan yerlere NEXT[BLD] etiketini ekler. Entegrasyon inşa her çarşamba NEXT[BLD] etiketi olan dosyalar kullanılarak yapılır. İnşa sonucunda oluşan hatalar sektör sorumluları tarafından değerlendirilir ve gerek görülürse gün içerisinde yeni bir inşa yapılır. Şekil 6'da inşa süreç akışı görülmektedir.

Şekil 6. Tüm İnşa Akışı



İnşa takımı NEXT inşa'sının anaçizgi olmasına karar verirse tüm NEXT[BLD] etiketi olan yerlere MCMP[BLD] etiketi eklenir. Her hafta inşa görünümleri temizlenir ve yeniden inşa yapılır.

Geliştirici istediği inşa görünümünden istediği derlenmiş kısımları kendi çalışma alanına getirir.

5. Sonuç

Projenin başlangıcında Yazılım Konfigürasyon Yönetimi gereksinimleri belirlenmeli, bu gereksinimleri tam olarak karşılayan, uygulanabilir süreçler tanımlanmalı ve bu kapsamda uygun araçlar -Konfigürasyon Yönetimi, Değişiklik Yönetimi vb- belirlenmelidir.

Proje personeline süreçler ve araçların kullanımı konularında eğitimler verilmelidir. Projeye yeni katılan her personelin bu eğitimi alması sağlanmalıdır. Bu eğitimler süreçlerin ve araçların etkin kullanımı için çok önemlidir

Proje Yönetimi tarafından uygulamanın takibi ciddiyetle yapılmalıdır.

Süreçler sürekli iyileştirmeye açık olmalıdır.

Yukarıda bahsedilen model ve süreçler, bu prensipler ışığında Barış Kartalı Projesinde başarı ile uygulanmaktadır. Yazılımın ilk dağıtımı başarıyla yapılmış olup, şu anda 2. dağıtım için çalışmalar devam etmektedir.

Kaynakça

[1]. Gupta A., "ClearCase Multisite: Supporting Geographically-Distributed Software Development", 19 Eylül 2004.

[2] Berczuk S. ve Appleton B., "Software Configuration Management Patterns: Effective Teamwork, Practical Integration", 2002.