BÜYÜK ÖLÇEKLİ BİR ELEKTRONİK HARP PROJESİNDE UÇ PROGRAMLAMA (XP) DENEYİMİ

Ali Özzeybek[1] Özgü Özköse Erdoğan[2] Serap Bozbey[3]
Gökhan Işık[4] Selami Taşel[5]

[1-5]Mikrodalga Sistem ve Teknolojileri, Aselsan A. Ş., Ankara
[1]e-posta: ozzeybek@aselsan.com.tr [2] e-posta: ozkose@aselsan.com.tr
[3]e-posta: bozbey@aselsan.com.tr [4] e-posta: gisik@aselsan.com.tr
[5] e-posta: stasel@aselsan.com.tr

Özetçe

Elektronik Harp sistemleri birçok yazılım ve donanım biriminin yer aldığı büyük ölçekli sistemlerdir. Bu sistemlerin geliştirme zamanı 3-5 yıl kadar sürebilmektedir.
UP (Uç Programlama - eXtreme Programming) [1] basitlik, iletişim, geri bildirim ve cesaret üzerine kurulu çevik bir yazılım geliştirme yöntemidir. Bu geliştirme yöntemi daha çok küçük yazılım projeleri için önerilmiş olmasına rağmen; içerdiği değerler, müşteri odaklı çalışma ve yazılım kalitesini artırmaya yönelik pratikleri nedeniyle büyük bir Elektronik Harp projesinin Kullanıcı Arayüzü (KA) yazılımı geliştirmesinde kullanılmasına karar verilmiştir.
Bu bildiride, büyük ölçekli bir askeri projede uygulanan UP deneyimi anlatılmakta ve UP pratiklerinin değerlendirilmesi yapılmaktadır.

1. Giriş

ASELSAN yaklaşık 20, MST (Mikrodalga ve Sistem Teknolojileri) Grubu ise 15 yıldır Radar ve Elektronik Harp (REH) sahasında önemli sistemler tasarlamakta ve üretmektedir; bu sistemleri oluşturan donanım ve yazılım konfigürasyon birimleri büyük oranda şirketimiz tarafından geliştirilmektedir.
Makalede referans alınan proje Radar Elektronik Harp konusundaki büyük projelerden biridir. Proje kapsamında "Kullanıcı Arayüzü" (KA) adını verdiğimiz bir Yazılım Konfigürasyon Birimi (YKB) UP uygulanarak geliştirilmiştir. KA yazılımı geliştirme ekibi, zaman zaman değişmekle birlikte ortalama yedi kişiden oluşmuştur.
Proje ve YKB büyüklükleri Tablo 1 ve Tablo 2'de verilmiştir. Projede, C/C++ ve Java programlama dilleri kullanılmıştır. Tablo 1'deki ölçümler genel toplamı göstermektedir.

Tablo 1: Proje büyüklük bilgileri

Toplam YKB sayısı	19
Toplam sınıf sayısı	4367
Toplam kod satır sayısı	772000

Tablo 2: KA YKB'si büyüklük bilgileri (kodlama dili Java)

Donanım arayüzleri sayısı	4
Yazılım arayüzleri sayısı	5
Sınıf sayısı	1713
Satır sayısı	106636

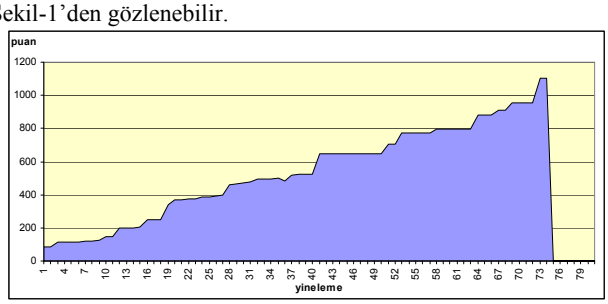
KA yazılımı sistemin kullanıcı arayüzünün yanı sıra veri tabanı ve harita arayüzünü de sağlamaktadır. Ayrıca sistemin genel senaryo kontrolünü yaparak gömülü yazılımları yönlendirmektedir.
Proje, SSM (Savunma Sanayii Müsteşarlığı) ile yapılan sözleşme kapsamında MIL-STD-498 standardına uygun olarak geliştirilmektedir. Proje kapsamında Sistem Gereksinim Özellikleri ve Sistem Tasarım Dokümanları hazırlanmıştır. Proje sadece yazılım projesi olmadığı için sistem seviyesindeki gereksinimler, donanımlar ve çeşitli yazılımlar tarafından karşılanabilmektedir. Bu açıdan UP ile geliştirilen KA yazılımı için gereksinimler, temel olarak Sistem Tasarım dokümanı içindeki "Sistem İşleyiş Senaryoları" (sistem senaryoları) bölümünden ve sistemin mimarisinden gelmektedir.
Projede UP uygulaması sistem senaryoları belirlendikten sonra başlamıştır ve hikayelerin oluşturulmasında sistem senaryoları temel alınmıştır. Ancak, hikayelerin yazılması ve gerçeklenmesi sırasında sistem senaryolarındaki detayın yeterli olmadığı görülmüş ve hikayeler daha detaylı yazılmıştır.
Projede UP müşteri ekibi yazılım ekip lideri ve gömülü yazılım sorumlularından oluşturulmuştur. Yazılım ekip lideri KA yazılımı geliştirmesinde yer almamış, sadece müşteri olarak çalışmıştır. Müşteri ekibi yazılım mühendislerinden oluşmasına rağmen, sistem mühendisi bakışı ile yazılımı kullanacak bir müşteri olarak görev almıştır.

2. Uygulanan UP Pratikleri

2.1. Planlama Oyunu

Planlama Oyunu, temel olarak UP çalışmasının planlanması ve yineleme içeriğinin belirlenmesidir; hikayelerin yazılması ve hikayelere puan verilmesi aktivitelerinden sonra, önceden seçilen aralıklarla tekrarlanan yineleme toplantısı, geliştirme ve ilerlemenin izlenmesi alt adımlarından oluşur [8]. Projede yineleme süresi olarak bir hafta seçilmiştir ve planlama oyunu bu yineleme süresi ile son iki ay haricinde düzenli olarak uygulanmıştır.

Hikayelerin yazılması: Projede sistem senaryoları önceliklerine göre aşamalı bir şekilde hikayeler olarak müşteri ekibi tarafından yazılmıştır. Her bir hikaye tam çalışan tek bir ihtiyacı tanımlamalıdır, bu açıdan sistem senaryoları birçok hikaye ile ifade edilmiştir. Senaryolardan kaynaklanan hikayelere zaman içinde yeni ihtiyaçlar, hatalar, yeniden yapılandırma ve dokümantasyon hikayeleri de eklenmiştir. Hikayelerin zaman içinde aşamalı olarak yazılması durumu Şekil-1'den gözlenebilir.



Şekil 1: Yineleme – Toplam puan çizelgesi

Hikayeler müşteri ekibi tarafından MS Excel ile takip edilmiş, ancak geliştiricilere hikaye kartları ile verilmiştir. Hikaye kartları, hikayenin kısa tanımı, varsa açıklaması, yazılış tarihi ve hikaye puanından oluşmuştur. Yazılan hikayelere örnekler Tablo 3'de verilmiştir.

Tablo 3: Örnek hikayeler

no	konu	hikaye	yineleme	puan	tarih
5	grafik	grafiklerde ızgara koy/kaldır işlevinin çalışması	6	3	05.04.05
18	grafik	grafiklerde "Yenile" işlevinin çalışması.	7	2	05.04.05
76	grafik	grafiklerde "yenile" işlevinde grafiklerde yayınların temizlenmemesi	15	2	30.06.05
34	grafik	genlik grafiğinde tepe-tut olması	6	3	26.04.05

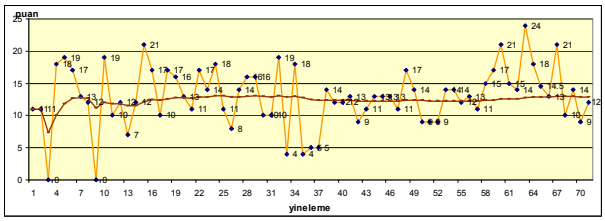
Hikayelere puan verilmesi: Yinelemelerin planlanması ve proje takvimine uygunluğun izlenebilmesi için hikayelere puan verilmesi gerekmektedir. Hikayelerin yazılmasından sonra geliştirme ekibi hikayeleri puanlandırmıştır. Hikayelere puanlar göreceli olarak verilmiştir. Örneğin Tablo-3'teki 5 numaralı hikayeye 3 puan verilerek puanlamaya başlanmıştır; 18 nolu hikaye, 5 numaralı hikayeye göre gerçeklenmesi daha kolay olarak değerlendirilmiş ve 2 puan verilmiştir. Puanlama sırasında müşteri ekibinin de bulunmasının hikayelerin doğru algılanmasını sağladığı görülmüştür, ancak puan verilmesi sırasında müşteri ekibinin verilen puanları çok bulması zaman zaman tartışmalara yol açmıştır.
Geliştirme ekibi sürekli puanlama yaptığı için, zamanla hikayeleri daha doğru puanlayabilme yetisi kazanmıştır. Böylece geliştirme döneminin başında hikayelere verilen puanlar tutarsızlık gösterirken, zamanla tutarlı duruma gelmiştir.

Yinelemenin planlanması (geliştirilecek hikayelerin seçilmesi) : Her yineleme toplantısının ardından geliştirme ekibi bir sonraki yineleme için istediği puanı belirtmiştir. Geliştirme ekibi istediği puan miktarını daha önceki yinelemelerde aldığı puanlara ve gelecek yinelemede ekibin durumuna göre belirlemiştir. Müşteri ekibi de istenen puanda hikayeyi önceliklerine göre seçerek bir sonraki yineleme için vermiştir. Müşteri ekibi içerisinde yazılım ekip lideri de yer aldığı için hikaye seçiminde KA yazılımının gömülü yazılımlar ile entegrasyonu da planlanmıştır.

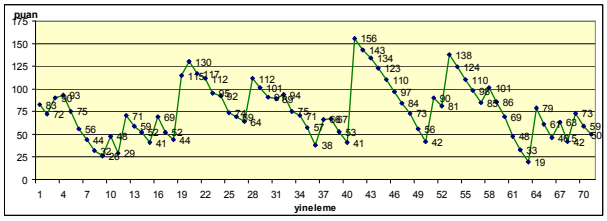
Geliştirme: Geliştirme ekibi elemanları sorumlu oldukları hikayeleri gerçeklemeye, makalede anlatılan pratikleri uygulayarak başlamışlardır. Geliştirme sırasında sadece verilen hikayeler gerçeklenmiştir, öngörülmeyen işler yeni hikaye olarak müşteri ekibine önerilmiştir. Geliştirme sırasında anlaşılmayan, kararsız kalınan noktalar müşteri ekibine danışılarak belirsizlikler giderilmiştir. Aynı zamanda müşteri ekibi de kabul testlerini yazmışdır. Kabul testlerinin yazılma zamanının geliştirme ile senkron olmasının önemli olduğu tecrübelerle anlaşılmıştır.

Yineleme toplantısı: Yineleme toplantısında, müşteri ekibi önce otomatik kabul testlerini çalıştırmış, daha sonra ise hikayelerdeki işlevleri uygulama üzerinden test etmiştir. Toplantı sonunda hikayelerin bitip bitmediğine karar verilmiş ve bir sonraki yinelemenin planlanmasına geçilmiştir.

İlerlemenin izlenmesi: İlerlemenin izlenebilmesi için haftalık puan çizelgesi (Şekil 2) ve kalan puan çizelgesi (Şekil 3) olmak üzere iki çizelge kullanılmıştır. Bu çizelgeler, her yinelemenin sonunda biten hikayelerin MS Excel üzerinde işaretlenmesi ile oluşturulmuştur. Planlama oyunu sayesinde, her zaman ne kadar işin kaldığı ve işlerin yaklaşık ne zaman bitebileceği gerçek verilere dayanarak kestirilebilmiştir.



Şekil 2: Yineleme – Haftalık puan çizelgesi



Şekil 3: Yineleme – Kalan puan çizelgesi

Projenin teslimat aşaması sırasında planlama oyunu sekteye uğramış, 2 ay kadar yineleme toplantıları yapılamamıştır. Yine aynı dönemde, müşteri ekibi gömülü yazılım sorumlularından oluştuğu için kabul testleri de yazılamamış, teslimat sürecindeki boşluklarda tamamlanmaya çalışılmıştır. Yoğun geçen 2 aylık dönemin ardından, kalan işler hikayeler olarak belirlenip UP sürecine geri dönülmüştür. Bunun yanında, yineleme toplantılarının yapılamadığı 2 aylık teslimat sürecinde yapılan işler için toplam 2 aylık dönemi içeren bir yineleme adımı kullanılarak, bu dönemde yapılan işlerin de takip edilmesi sağlanmıştır.

2.2. Küçük ve Kısa Aralıklı Sürümler

Küçük ve kısa aralıklı sürümler her yineleme sonunda yeni bir sürümün çıkarılmasıdır. Ancak bu sürüm ürüne dönüştürülen bir sürüm olmayabilir, önemli olan her sürümde yeni yetenekler eklenmiş devamlı çalışan bir kod olmasıdır. Müşterinin, her yineleme sonunda çalışan yeni bir sürümü alıp kullanabilmesi hedeflenir [1].
Her yineleme toplantısı sonunda çalışan bir yazılıma ulaşılmıştır. Ancak sistem entegrasyonunda kullanılacak ve sistem testlerine girecek sürüm planlaması daha büyük adımlarla yapılmıştır.

2.3. Sistem Metaforu

Sistem metaforu sistemin ne yaptığını ve mimarisini anlatan kısa bir (1-2 sayfalık) doküman olması ve tüm ekip elemanlarının bu dokümanı bilmesidir [1].
Sistemin büyüklüğünden dolayı genel sistem mimarisini ve işleyişini anlatan sistem metaforu hazırlanamamıştır. Ancak projenin başında ve gelişme aşamalarında sistemi, yazılım-donanım altyapısını ve sistem senaryolarının anlatıldığı toplantılar düzenlenmiş, bütün yazılım ekibinin sistemi tanıması hedeflenmiştir. Bunun yanında KA yazılımının mimarisi hazırlanmış ve geliştirme ekibine duyurulmuştur, mimari ve mimari kurallar kısa bir dokümanda anlatılmıştır.

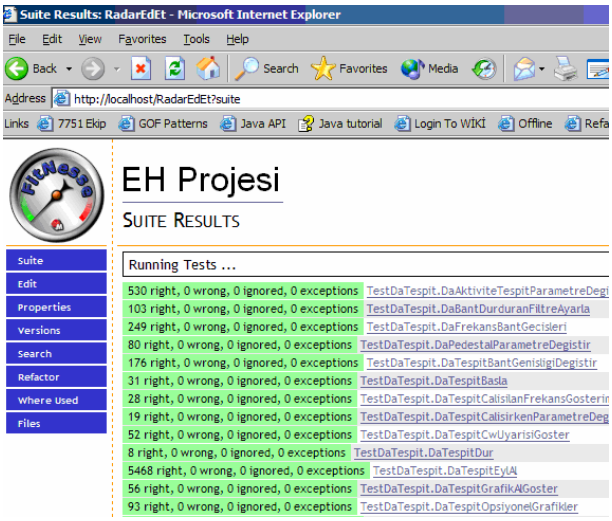
2.4. Basit Tasarım

Basit tasarım UP'nin ulaşmayı istediği en temel pratiklerden biridir. UP, tasarımı en başta yapılması gereken bir iş olarak görmeyip tüm geliştirme sürecinde yapılması gereken bir iş olarak tanımlar ve tasarımın devamlı yeniden yapılandırma ile daha basit hale getirilmesini hedefler.
Projenin başında sadece mimari tasarım yapılmış, yazılım için daha detaylı bir tasarım yapılmamıştır. Geliştirme sırasında, yeni yetenekler eklenmeden önce küçük ön tasarım toplantıları gerçekleştirilmiş, burada alınan tasarım kararlarına uygun olarak geliştirme yapılmıştır. Geliştirme sonrasında, gerçeklenen yazılım parçası yeniden yapılandırılarak daha basit hale getirilmeye çalışılmıştır. Ancak, yineleme toplantısına yetişme baskısı, teslimat baskısı gibi nedenlerden dolayı uygulamanın bazı yerleri yeniden yapılandırılacak notu konularak geçilmek zorunda kalmıştır.
Başta verilen katmanlı mimari kararları, test edilebilirlik gerekleri ve kullanılan tasarım kalıpları bazı noktalarda tasarımın karmaşıklaşmasına neden olmuştur.

2.5. Test

Test UP'nin en önemli pratiklerinden biridir. Tasarımı basitleştirmek için yeniden yapılandırma yapmaya ancak yazılımın otomatik testleri varsa cesaret edilebilir. Testler, birim testleri ve kabul testleri olarak ikiye ayrılabilir [1].

Birim testleri: UP'de birim testlerinin yazılması için UP geliştirme yöntemi olarak TDD (Test Driven Development) yöntemi önerilmektedir. Bu yönteme göre önce test yazılır, testler çalıştırılır ve testlerin kaldığı görülür. Sonra testi geçirecek kod yazılır ve en son yazılan kod yeniden düzenlenir. Uygulamaya, önce testi yazılmadan tek satır kod eklenmez [1].
Projenin başlarında JUnit kullanılarak TDD uygulanmaya başlanmış, ancak taklit nesnelerin kullanımındaki zorluklar, eşli programlamanın azalması ve zaman kısıtı gibi nedenlerden dolayı TDD uygulanamamıştır. TDD yönteminin uygulanamaması toplamda birim testlerinin az olmasına neden olmuştur. Birim testlerinin az olması ise yeniden yapılandırma çalışmalarında sadece kabul testlerine güvenilmesi sonucunu doğurmuştur. Her ne kadar TDD uygulanamasa ve sonucunda birim testleri eksik kalsa da bazı önemli senaryoların birim testleri daha sonra eklenmiştir.
Kabul testleri: Kabul testleri işlevlerin müşterinin istediği şekilde çalıştığını doğrulayan otomatik testlerdir. Bu testler, müşterinin gerçekte ne istediğinin anlaşılmasını ve çıkan ürünün müşterinin istediği ürün olmasını sağlamaktadır [1].
Müşteri testleri için Fit Framework [2] ile çalışan Fitnesse kullanılmıştır (Şekil 4). Müşteri ekibi hikayeleri test olarak ifade etmiş, hikaye üzerindeki tartışmalar kabul testleri üzerinde gerçekleştirilmiştir. Kabul testlerini çalıştıracak ara kodlar yazılırken görünüm öğeleri (panel, frame gibi…) otomatik testlere alınmamıştır, bu da zaman zaman otomatik testlerde geçtiği görülen bazı hikayelerin el ile testlerde geçmemesi gibi durumlara yol açmıştır. Ancak görünüm öğelerinde akıl olmamasını sağlayan tasarım kalıpları [3,4,5] kullanıldığı için işlevsellik tam olarak test edilebilmiştir.
KA yazılımının arayüzü olan yazılımların taklitleri yazılmış, taklit nesneler otomatik testler ile çalıştırılarak kabul testlerinde arayüzlerin de test edilmesi sağlanmıştır. Yazılımlar arası haberleşme için ara katman teknolojisi olarak CORBA [6] kullanılması, otomatik testlerde taklit nesnelerin daha kolay yazılmasını sağlamıştır. Bu sayede diğer yazılımlar ile entegrasyonlar kolaylaşmıştır.



Şekil 4. Kabul testleri

Projede 400 adet birim testi ve 95 kabul testinde 21488 test durumu tanımlanmıştır. Zamanla kabul testleri hantallaşmış ve çok uzun süremeye başlamıştır (toplam 3-4 dakika), kabul testlerinde de yeniden yapılandırma yapılarak testlerin hızlanması sağlanmıştır.

2.6. Yeniden Yapılandırma

Yeniden yapılandırma, kodun işlevselliğini değiştirmeden iç yapısının daha iyi hale getirilmesidir. Hedeflenen, kod tekrarlarının giderilmesi, bağımlılıkların azaltılması, her sınıfın tek bir işten sorumlu olması gibi iyileştirmelerdir [1, 7].

Geliştirme sırasında, yeni yetenekler eklenirken devamlı olarak yeniden yapılandırma uygulanmıştır. Bunun yanında, yeni yeteneğin tasarımı zorladığı durumlarda daha kapsamlı yeniden yapılandırmalar için de hikayeler alınmıştır. Yeniden yapılandırma kapsamında sürekli olarak kod tekrarları azaltılmaya çalışılmış, metod isimleri değiştirilmiş, büyük sınıflar parçalanarak sadece kendi görevlerini yapar duruma getirilmiştir. Yeniden yapılandırma sırasında birim testleri az olduğu için otomatik kabul testlerine ve geliştirme ortamının yeteneklerine güvenilmiştir.
Teslimat döneminin yaklaşması ile yeniden yapılandırma düzenli olarak uygulanamamıştır, ama bir doküman oluşturularak maliyeti yüksek yeniden yapılandırma işleri bu dokümana girilmiştir.

2.7. Eşli Programlama

Eşli programlama iki kişinin aynı bilgisayarda birlikte kod yazmasıdır, geliştirme sırasında eşler devamlı değişir. Eşli programlama ile kodlama standardına uyulması, basit tasarım yapılması ve testlerin iyileştirilmesinin yanında ortak kod sahipliği de sağlanır [1].
Eşli programlama, projenin erken aşamasında tam olarak uygulanmış ve verimimizi çok olumlu etkilemiştir. Geliştirme ekibi içerisinde herkes birbiriyle eşli çalışmıştır. Alan bilgisinin paylaşılması, mimari bilgisinin geliştirme ekibine dağıtılması, ortak kodlama stilinin gelişmesi, birim testlerinin yazılması konusunda çok faydalı olmuştur. Projenin gelişme ve sonuçlanma aşamasında ise her ekip üyesi için bir rol ortaya çıkmaya başlamıştır. Projenin ilerleyen aşamalarında yazılım mimarisi oturup, geliştiriciler arasında ortak bir programlama yöntemi gelişince eşli programlama bırakılmıştır. Ancak, geliştirme ekibine yeni elemanın katıldığı durumlarda, yeni katılan kişiyle eşli çalışılarak daha çabuk adaptasyonu sağlanmıştır.
Bu pratiğin uygulanmasında geliştirme ekibi elemanlarının çalışma şekilleri önemli bir etken olmuştur. Çevresel koşullar da (masa, monitör gibi) bu pratiğin uygulanmasında etkili olabilmektedir. Tablo-1'de proje boyunca uygulanan eşli programlama yüzdeleri verilmektedir.

Tablo 4: Eşli programlama yüzdesi

Proje takvimi	Eşli programlama uygulanma oranı
1. 6 ay	% 80
2. 6 ay	%60
3. 6 ay	%30
4. 6 ay	%10

2.8. Ortak Kod Sahipliği

Ortak kod sahipliği her zaman herkesin uygulamanın bir yerini değiştirebilme yetisini ve cesaretini kazanmasını hedeflemektedir [1].
Projede ortak kod sahipliği hedeflenmiştir, bu amaçla eşli programlama yapılmış ve ekip elemanlarının yazılımın çeşitli bölümlerinde görev alması sağlanmıştır. Eşli programlama, günlük kısa toplantılar ve tasarım değerlendirme toplantıları sayesinde geliştirilen kodların herkes tarafından bilinmesi sağlanmıştır. Ancak, eşli programlamadan uzaklaşıldıkça ve problemleri kısa sürede çözme baskısı arttıkça, değişiklikler için geliştirmeyi yapan elemana bağımlılık artmıştır.
Projede, herkesin koddaki herhangi bir yere hızlı müdahale edebilmesi sağlanamamakla birlikte, kodun belli bir yeri için en az iki kişinin müdahale edebilmesi sağlanmıştır.

2.9. Sürekli Entegrasyon

Sürekli entegrasyon, yeni eklenen kodların devamlı olarak bütün yazılımla entegre bir şekilde çalışması ve bütün ekip tarafından kullanılabilir olmasıdır [1]. Bu amaçla, yazılım geliştirme, konfigürasyon kontrol sisteminden kodun alınması (update) ve değişecek dosyanın alınması (check-out), yeni kod eklenmesi, testlerin çalıştırılması ve kaynak kodların yeniden konfigürasyon kontrol sistemine konulması (check-in) döngüsünde gerçekleştirilir.
Projede konfigürasyon kontrol sistemi yazılım geliştirme ortamına entegre olarak kullanılmıştır. Kodlama sırasında UP'nin önerdiği döngü temel olarak uygulanmıştır. Birim testlerinin azalması ile birlikte birim testlerinin bu döngüde çalıştırılma sıklığı azalmıştır, ancak kabul testleri en az günde bir çalıştırılmıştır. Bu sayede, her zaman yeni yetenekleri de içerecek şekilde çalışan bir kodun konfigürasyon kontrol sisteminde bulunması sağlanmıştır.

2.10. 40 Saat/Hafta

UP mesai saatlerinin en verimli şekilde kullanılması ile verimsiz fazla mesai uygulamalarına gerek kalmamasını hedefler. Amaç geliştirmenin dinç, yaratıcı ve dikkatli bir şekilde yapılması ile kodun kalitesinin artmasıdır. [1]
Bu pratiğe genel olarak uyulmuştur. Her yinelemede sadece planlanan hikayeler için iş gücü harcanması ve pratiklerin bir bütün olarak uygulanması sayesinde iş gücünün doğru ve verimli kullanılması sağlanmıştır. 2 senelik aktif geliştirme süresinin her haftası teslimat havasında geçtiği için projenin sonuna doğru da değişen bir şey olmamıştır.

Tablo 5: Haftalık çalışma saati

Proje takvimi	Haftalık çalışma saati
1. 6 ay	40
2. 6 ay	40
3. 6 ay	40
4. 6 ay	45

2.11. Müşterinin Geliştirme Ortamında Olması

UP'de gerçek müşterinin geliştirme ortamında geliştiricilerle birlikte çalışması ile çıkacak ürünün müşterinin isteğine uygun olması hedeflenir [1]. Şartname ve sistem gereksinim dokümanları gibi sistemden ve yazılımdan neler beklendiğini anlatan dokümantasyon bulunsa da ortaya çıkan ürün gerçekten müşterinin istediği ürün olmayabilir.
Proje sırf yazılım projesi olmadığı ve askeri projelerde gerçek müşterinin geliştirme ortamında olmasını sağlamak zor olduğu için müşteri ekibi şirket içerisinden seçilmiştir.
Projede müşteri ekibi, hikayeleri yazmış, yineleme içeriğini belirlemiş ve kabul testlerini yazmıştır. Bu sayede

- Senaryolarda belirsiz olan kısımlar ortak bir şekilde görülerek, zamanında netleşmesi sağlanmıştır ve geliştirme ekibi kendi karar vermek zorunda kalmamıştır.
- Sadece KA yazılımı için UP uygulanmasına rağmen, projedeki bütün yazılımların paralel bir şekilde ilerlemesi sağlanarak ara entegrasyonlar planlanabilmiştir.

Müşteri ekibi, özellikle senaryolardaki açık noktalarla ilgili görüş almak için sistem mühendisi, algoritma uzmanı ve teknik yöneticiden destek almıştır. Bunun yanında projenin ara aşamalarında bu grupla toplantılar yapılarak gelinen nokta aktarılmış ve yeni istekler hikaye olarak alınmıştır.
Bunların yanında gerçek müşteri olan askerler ile de bir ortak çalışma planlanmıştır. Bu kapsamda ilk olarak prototip yazılım alıcı makama tanıtılarak görüşleri alınmıştır. İkinci aşamada operatörler ile bir haftalık ortak çalışma yapılarak son müşterinin yazılımı kullanması ve geri bildirimde bulunması sağlanmıştır. Yine son müşteriden alınan yeni istekler de hikaye olarak iş listesine eklenmiştir.

2.12. Kodlama standardı

Kodlama standartı kullanılması, kodun tutarlı ve kolay okunabilir olması ile tüm geliştirme ekibinin yeni yetenek ekleyebilmesini ve yeniden yapılandırma yapabilmesini sağlar [1].
Projenin başında uyulacak kodlama standardı belirlenmiş ve ilgili kurallar geliştirme ortamına tanıtılmıştır. Eşli programlama ile ortak kod yazma kültürü oluşturulmuştur. Kodlama standardına uyulup uyulmadığı kod okumalarla denetlenmemiş, eşli programlama ve geliştirme ortamının yeteneklerine güvenilmiştir.

3. Sonuçlar

Aselsan'da genel kabul gören yazılım geliştirme yöntemi döngüsel (spiral) süreç uygulanmasıdır ve döngüler 6 ay – 3 yıl arasında değişebilmektedir. Bu yöntemde yazılım gerek analizi, mimari tasarım, detaylı tasarım ve kodlama, birim test ve entegrasyon ile yazılım yeterlilik testleri adımları sırası ile uygulanmaktadır. Bir sonraki döngüde yeni gereklerin eklenmesi ve mevcut gereklerin değişmesi sık rastlanan bir durumdur. Ancak döngüsel yöntem bu tür değişikliklere esnek olmadığı için değişikliklerin maliyeti yüksek olabilmekte ve yazılım geliştirme aşamasında etkileri tam test edilemeyebilmektedir. Buna karşılık UP uygulanarak aşağıdaki değerlendirmelerde belirtildiği gibi yazılımın değişikliklere esnek olması ve doğru yetenekler ile geliştirilmesi sağlanmıştır.
Projede uygulanması istenen MIL-STD-498 standardı, bazı standart doküman setlerinin belli zamanlarda hazırlanmasını ve formal olarak gözden geçirilmesini istemektedir. UP uygulanan yazılım için de MIL-STD-498 standardına uygun dokümanlar hazırlanarak formal gözden geçirmeler gerçekleştirilmektedir. Bu nedenle bildiride bahsedilen yazılımın gerekleri hem hikayeler olarak, hem de gereksinim izleme aracında formal cümleler olarak izlenmiştir ve kabul testlerine ek olarak bağımsız bir test ekibi tarafından (başka bir müdürlük elemanları) test tanımları hazırlanmış ve yazılım bu tanımlara uygun olarak da test edilmiştir. Sonuç olarak standarda uygunluk açısından bazı çalışmaların iki farklı araç/yöntem ile yapılması gerekmiştir.
Bu bildiride bahsedilen ve UP yöntemi ile geliştirilen KA yazılımında iki senelik geliştirme süresince 830 hikaye belirlenmiş, 18'i ertelenmiş, 31'i iptal edilmiş, 781 hikaye ise gerçeklenmiştir. Son dönemde proje alıcı makam ile yapılan sistem uygunluk testleri başarıyla geçilerek KA yazılımı tamamlanmıştır.
UP uygulamasının genel değerlendirmesi aşağıda verilmiştir.

- Sürekli yeniden yapılandırma ile yazılım tasarımının basit tutulması sağlanmıştır. İstenen değişikliklere ve isteklere tepki süresi birkaç günü geçmemiştir.
- Birim testleri eksik de olsa kabul testleri yeniden yapılandırma yapılabilmesi için bir güvence oluşturmuştur. Müşteri ekibi gömülü yazılım ekibinden oluştuğu için işlerin yoğunlaştığı dönemde bu testlerin yazılmasında aksamalar meydana gelmiştir, ancak yoğun dönemlerden sonra yazılmayan kabul testleri tamamlanmaya çalışılmıştır.
- Kabul testlerini ve testleri geçirecek ara kodları yazmak, geliştirme aşamasında harcanan işgücünü arttırmıştır. Ancak kolay entegrasyon, az hata çıkması ve kolay değiştirilebilirlik sayesinde toplam iş gücü azalmıştır.
- Yapılacak işlerin, belirlenen süre içerisinde bitirilebilecek kadar küçük parçalara ayrılması, çalışanları motive etmiş, kodlama yapan kişinin büyük resim içinde kaybolmasını engellemiştir.
- Planlama, sürekli yapılan bir faaliyet olduğu için, yazılımın ne zaman biteceği herhangi bir zamanda yaklaşık olarak hesaplanabilmiştir.
- Ekip liderinin müşteri ekibinde olması, yineleme toplantıları sırasında geliştirme ekibindeki çalışanlarda yazılım ekip lideri tarafından performansının değerlendirildiği izlenimini yaratmış ve bu durum zaman zaman tartışmalara yol açmıştır. Ancak konuyla ilgili toplantılar yapılarak müşteri ekibi ile iletişimi artırma yönünde kararlar alınmıştır. Yazılım ekip liderinin geliştirme ekibi içerisinde yer almamasına rağmen zaman zaman geliştirilen hikayeleri yazılım mühendisi bakış açısıyla değerlendirmiştir. Yazılım ekip liderinin de bu süreçte aktif rol oynaması testlerin yazılmasını ve işlerin sahiplenilmesini sağlamıştır.
- Tüm geliştiricilerin yazılımın herhangi bir yerine müdahale edebilme cesaretini kazanması sağlanmıştır. Bu sayede geliştirme ekibindeki eleman değişimleri ana programı çok az etkilemiştir.
- UP, yazılım ekibinin kendi içerisinde ve sistem mühendisleriyle iletişimini artırmıştır. Projenin ilerlemesi sırasında oluşan gereksinim değişikliklerine direnç gösterilmediği için sistem yöneticisi ve proje yöneticisi ile yapılan toplantıların daha verimli ve hedefe yönelik geçmesi sağlanmıştır.
- Geliştirme ekibinin birlikte çalışmaya alışması ile projenin sonlarına doğru bitirilen puanlarda artış olmuştur.
- Müşterinin isteğine uygun ürün üretme çabası ile senaryolarda belirsiz olan kısımlar erkenden netleştirilmiş ve müşteri ekibi de karar vermek durumunda kalmıştır. Karar verilemeyen yerlerde sistem yöneticisinin ve proje yöneticisinin görüşü alınarak müşteri isteğine uygunluk sağlanmaya çalışılmıştır.
- Müşteri ekibi içerisinde gömülü yazılım sorumlularının olması arayüz değişikliklerinin gömülü yazılımlarda da paralel şekilde yürütülmesini ve sistem entegrasyonunda hataların en aza indirgenmesini sağlamıştır. Sistem uygunluk testleri aşamasında da çıkan hata miktarının daha önceki KA yazılımlarına göre azaldığı gözlenmiştir.

Sonuç olarak UP tüm pratikleri ile tam olarak uygulanamamış olmasına rağmen, UP'nin hedeflediği dört ana hedefin sağlandığı değerlendirilmektedir:

- basitlik
- iletişim

- geri bildirim
- cesaret

Böylece müşterinin isteğine uygun, doğru bir ürünün iyi bir performansla çıkması sağlanmıştır.

4. Kaynakça

[1] Beck, K. Extreme Programming Explained: Embrace Change, Addison-Wesley Professional Publishers, 2000.

[2] Object Mentor Group, Fitnesse, www.fitnesse.org, 2006.

[3] Fowler M., "GUI Architectures", www.martinfowler.com/eaaDev/uiArchs.html, 2006.

[4] Alles M., Crosby D., Harleton D. ve arkadaşları,"Presenter First: Organizing Complex GUI Applications for Test-Driven Development", Agile 2006 International Conference, 2006.

[5] Fowler M., "Model View Presenter", www.martinfowler.com/eaaDev/uiArchs.html, 2004.

[6] Object Mentor Group, "Common Object Request Broker: Architecture and Specification", http://www.omg.org/docs/formal/99-10-07.pdf, 1998.

[7] Fowler M., Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional Publishers, 1999.

[8] Beck, K., Fowler, M. Planning Extreme Programming, New York, NY: Addison Wesley Longman, 2001