IRIS Gömülü Yazılım Ürün Hattında Yazılım Ekosistemine Geçiş Çalışmaları

Berkhan Deniz

Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü, SST Sektör Bşk.
ASELSAN A.Ş.

berkhand@aselsan.com.tr

Özet. Yazılım ürün hatları, geliştirilen farklı ürünlerdeki ortak bileşenlerin yeniden kullanımını sağlayarak, yazılım geliştirme sürelerinin ve maliyetlerinin azaltılmasında önemli rol oynamaktadır. Ancak, yazılım ürün hatlarının kapsamı genişledikçe ve hitap ettiği pazar arttıkça başta tasarlandığından çok daha karmaşık hale gelebilmektedir. Bu duruma bir çözüm olarak yazılım ürün hatlarının yazılım ekosistemlerine dönüştürülmesi önerilmektedir. Yazılım ekosistemlerinde ana yüklenici firma, yazılım geliştirdiği platformu firma dışından geliştiricilere açmakta ve bu geliştiricilerin var olan platforma eklemeler yapmasına izin vermektedir. Burada anlatacağımız çalışmada ise, Aselsan'da geliştirilen bir gömülü yazılım ürün hattının (IRIS), altyüklenici firmalar tarafından kullanımı yoluyla, Aselsan'ın kendi ekosistemine geçişi yönünde yapılan çalışmalar ve karşılaşılan zorluklar anlatılacaktır.

Anahtar Kelimeler: Gömülü yazılım geliştirme, Yazılım ekosistemi, Altyüklenici

Studies on transitioning from IRIS Software Product Line to Software Ecosystem

Abstract. Software product lines (SPLs) play an important role in reducing software development times and costs by ensuring the reuse of common components in the different products being developed. However, as the scope of SPLs and the market they address expand, they become increasingly more complex than originally designed. As a solution to this situation, it is recommended that software product lines be converted into software ecosystems (SECOs). In software ecosystems, the leading company opens the SPL platform to developers outside the company and allows developers to create extensions to existing platform. In this paper we will describe the studies and difficulties for the software ecosystem transition of an embedded software product line (IRIS) developed by Aselsan after being used by subcontractors.

Keywords: Embedded software development, Software ecosystem, Subcontractor

1 Giriş

Son yıllarda, Aselsan tarafından üretilen sistemlerin hem çeşitliliği hem de karmaşıklığı oldukça artmıştır. Ancak bu duruma zıt bir şekilde sistemlerin geliştirme ve teslimat süreleri her geçen gün azalmaktadır.

Bu gelişmelere paralel olarak, geliştirilen sistemleri yöneten gömülü sistem yazılımları da her yeni projeyle birlikte daha da karmaşıklaşmakta ve de bu yazılımların geliştirilmesi için ayrılan süreler azalmaktadır. Yazılımların geliştirilme süresini ve maliyetini azaltmak için en etkili metotlardan birinin yeniden kullanımı artırmak olduğu bilinmektedir. Yazılım ürün hatları geliştirilen farklı ürünlerdeki ortak bileşenlerin yeniden kullanımını sağlayarak, yazılım geliştirme sürelerinin ve maliyetlerinin azaltılmasında önemli rol oynamaktadır. Ancak, yazılım ürün hatlarının kapsamı genişledikçe başta tasarlandığından çok daha karmaşık hale gelebilmektedir. Aynı zamanda, yazılım ürün hattının sürdürülmesi ve gömülü sistemlere entegrasyonu için ihtiyaç duyulan işçilik miktarı da artmaktadır.

Yazılım ürün hattının, onu geliştiren ana ekibin dışındaki kullanıcılara açılarak; harici geliştiriciler tarafından kullanılması ve genişletilmesi kavramı yazılım ekosistemi olarak tanımlanmaktadır [1].

Burada anlatacağımız çalışmada, Aselsan'da geliştirilen IRIS gömülü yazılım ürün hattının, altyüklenici firmaların kullanımına açılması ve iki farklı seviyede kullanımı anlatılacaktır: IRIS ürün hattı için bileşen geliştirilmesi ve IRIS ürün hattının kullanılmasıyla son ürün geliştirilmesi.

Çalışmanın geri kalanı su şekilde düzenlenmiştir: Bölüm 2'de yazılım ekosistemi kavramı ile ilgili literatür bilgileri özetlenmiştir. Bölüm 3'te IRIS ekosisteminin altyükleniciler tarafından kullanımı ve karşılaşılan zorluklar aktarılmıştır. Bölüm 4'te ise sonuçlar ve gelecek için çalışma önerileri verilmiştir.

2 Literatür

Son yıllarda, araştırma-geliştirme ve inovasyon sürelerinin azalması ve bunun sonucunda yeni ürün ortaya çıkarmak için gerekli zamanların kısalması doğrultusunda, yazılım ürün hatları (YÜH) ilk tasarımlarından uzaklaşmakta ve kapsamları genişlemektedir [2]. YÜH'lerde amaç, farklı ürünlerde yeniden kullanılabilir bileşenler kullanarak ve hem platformda hem de bileşenlerde gerekli değişiklikleri yaparak, üretim maliyetlerini azaltmaktır. Ancak, YÜH büyüklüğü ve kapsamı genişledikçe; yani hattan çıkan ürün sayısı, ürün seviyesindeki gereksinimler, platformda ve bileşenlerde farklılıklar arttıkça, YÜH karmaşıklığı artmakta; yönetimi ve sürdürmesi zorlaşmaktadır [4].

YÜH ekosistemi yaklaşımı, geniş kapsamlı ürün hatları için bir çözüm olarak önerilmektedir [5]. Yazılım ekosistemlerinde ana yüklenici firma, yazılım geliştirdiği platformu firma dışından geliştiricilere açmakta ve bu geliştiricilerin var olan platforma eklemeler yapmasına izin vermektedir. Bu yaklaşımın başarılı olabilmesi için, bağımsız ürünler (bileşenler) geliştirecek farklı ekiplerin birbirinden bağımsız sürüm

çıkartabilme özgürlüğü olması gerekmektedir [5]. Bu da ancak, bileşenlerin birbirine olan bağımlılığının en aza indirgenmesiyle sağlanmaktadır.

Yazılım geliştirdiği platformu dış geliştiricilere açmak isteyen firmalar için iki farklı yaklaşım önerilmektedir [5]: Doğrudan ve dolaylı olarak. İkisi arasındaki fark, firmanın dış geliştiricilere herhangi bir erişim kısıtı koyup koymamasıdır. Güvenlik, gizlilik gibi sebeplerle paylaşılmaması gerekli bilgiler, doğrudan erişim hakkı bulunmayan firmalarla paylaşılmamaktadır.

2.1 Yazılım ekosistemlerindeki ana roller

Yazılım ekosistemi kavramı, yazılım endüstrisinde yeni fırsatlar ve beraberinde zorluklar ortaya çıkarmaktadır: Yeni iş yapış şekilleri, açık inovasyon, birlikte geliştirme gibi fırsatlara karşın; sahiplenme konusu, stratejik planlamalar, değişkenlik yönetimi gibi zorluklar.

Bir yazılım ekosisteminde üç ana unsur bulunmaktadır: Bir temel geliştirici (ana yüklenici), bir yazılım geliştirme platformu (ürün hattı) ve dış geliştiriciler. Çevre geliştiriciler platformu kullanarak kendileri için değer yaratırlar [15].

Bosch'a göre bir yazılım ekosisteminde dört seviye geliştirici bulunabilir [5]:

1. İç geliştiriciler: Ana yüklenici firmada bulunan geliştiricilerdir.

2. Stratejik geliştiriciler: Uzun süreli ortaklık için seçilmiş firmalardır, ürün hattına doğrudan erişimleri vardır.

3. Kısıtlanmış geliştiriciler: Ürün hattına doğrudan erişimleri olmayan dış geliştiricilerdir.

4. Bağımsız geliştiriciler: Platformu kullanarak, ana yüklenici firmadan bağımsız ürün geliştirirler.

Yukarıda belirtilen farklı rollerdeki geliştiriciler, ortak geliştirme platformuna çeşitli erişim noktaları ya da arayüzler üzerinden erişirler. Bir API veya SDK üzerinden, ya da çeşitli seviyede veri paylaşımı ve teknolojinin bir kısmını açık kaynaklı hale getirerek, geliştirme platformuna erişim sağlanabilir.

Ortak platformun yeniden kullanabilir hale getirilmesi ve çeşitli şekillerle dış erişime açılmasının ardından, ekosistemin gelişebilmesi için dış geliştiricilerin cesaretlendirilmesi ve ekosistem içerisinde aktif olmaları sağlanmalıdır [3]. Ana firmada bulunan geliştiriciler ekosistemin düzgün işlemesinden sorumludur [13]. Ortak platformun kullanıma yönelik oturmuş, tahmin edilebilir arayüzler ve dokümanlar ortaya koymak ana firmadaki geliştiricilerin sorumluluğundadır.

Hanssen tarafından yapılan alan çalışması sonucunda, ana yüklenici firma temelli yazılım ekosistemleri için kavramsal bir model oluşturulmuştur [3]. Bu modele göre, aşağıdaki teorik önermelerin yazılım ekosistemlerine uygun olduğu belirtilmiştir:

1. Yönetici rolünde bir firma olmalı ve çevresindeki diğer firmalar, arayüz ilişkileri ile bu firmaya bağlantılı olmalıdır.

2. Yazılım ekosistemleri kendi kendisini denetlemelidir; hem ana firma hem çevre firmalar birbirine uyum sağlamalıdır.

3. Yazılım ekosistemlerinin, hem müşterilerle, hem üçüncü taraf şirketlerle ve hatta rakiplerle ağ tabanlı bir ilişkisi vardır.

4. Yazılım ekosistemlerinin düzgün işlemesi açısından web toplantıları benzeri teknoloji kullanımları önem arz etmektedir.

5. Yazılım ürün hattı, yazılım ürünleri ve iş alanı tüm ekosistemin ortak değerleridir. Ürün hattı gelişimi de tüm ekosistemin ortak sorumluluğu altındadır.

2.2 Yazılım ekosistemlerinin gelişmesi ve sürdürülmesi

Harici paydaşlarla etkileşim ve iletişim halinde olunması ürün hattının birlikte yaratılmasını sağlamaktadır. Ürün hattının harici kullanıcıları da yazılım süreçlerine dâhil olmalıdır. Aynı zamanda, uzun vadeli stratejiler ve planlar paylaşılarak, harici geliştiriciler desteklenmelidir. Ana geliştirici firmanın temel görevi, iş alanının geliştirilmesi ve daha fazla harici geliştiricinin ekosisteme dâhil olmasını sağlamak olmalıdır [3].

Harici geliştiricilerle sağlıklı iletişim kurmanın en iyi yolunun, herkese açık olacak şekilde ortak platformla (ürün hattı) ilgili uzun dönem yol haritasının paylaşılması olduğu belirtilmiştir [5]. Bu sayede, harici geliştiriciler kendi stratejik planlamalarını ve geliştirme hedeflerini ortak platformla uyumlu olarak ortaya koyabileceklerdir. Ancak, ekosistemlerdeki hızlı değişiklikler ve şirketler arasındaki organizasyonel sınırlar, tahmin etmesi ve yönetmesi daha zor süreçlere neden olmaktadır [7]. Bu yüzden, geleneksel tanımlanmış süreçler gözden geçirilmeli ve süreçlerin basitleştirilmesi değerlendirilmelidir.

Ürün planları ve yol haritalarının paylaşılması yazılım ekosistemlerinin gelişmesine yol açacağı gibi, aynı zamanda harici partner şirketlerle planlama ve geliştirme konularında işbirliği yapılması da şirketler-arası inovasyonu güçlendirecektir. Seichter ve arkadaşlarına göre [10], uzun vadeli stratejik planların paylaşılması kadar, daha kısa vadeli geliştirme faaliyetlerinin de (gereksinim mühendisliği süreci, sürüm çıkartma planları vb.) harici paydaşlarla paylaşılması ekosistemin gelişimine katkıda bulunacaktır.

Bir ekosistemi belirleyen temel özellikler, kullanılan araçlar ve teknolojilerdir [13]. Yeni bir aktör ekosisteme dâhil olmak istediğinde, öncelikle bu özelliklere uyum sağlaması gerekmektedir.

Yazılım ekosistemlerinin var olması ve gelişimi açısından diğer temel iki prensip ise şeffaflık ve modüler sistem tasarımıdır [8]. Şeffaflık, tasarımla ilgili dokümanların (kaynak kod, tasarım dokümanları vb.) ekosistem aktörlerinin erişimine açık olmasıyla; modüler sistem tasarımı ise, sistemin birbiriyle etkileşimi azaltılmış, bağımsız yönetilebilir bileşenlere parçalanabilmesiyle ilgilidir. Ancak bu prensipler faydalı oldukları gibi, bazı zorlukları da beraberinde getirmektedir. Şeffaflık, çok fazla bilginin organize edilmesini gerektirir. Bunun da yönetimi zordur. Aynı zamanda, tam şeffaflık fikri hakların korunması, ticaret sırları ya da güvenlik gibi konulardan dolayı her zaman tercih edilmeyebilir. Modüler sistem tasarımı da, yazılım sistemlerinin değişken doğası nedeniyle her zaman mümkün olamamaktadır.

Ekosistemde şeffaflık ilkesinin uygulanması sayesinde, ekosistem aktörleri ekosistem içerisindeki iş yapış şekillerini, kullanılan teknik altyapıları öğrenebilirler. Ayrıca, bu sayede ekosistem üyeleri geliştirme aktivitelerinin durumu hakkına bilgi sahibi olur ve ekosistem içerisinde koordinasyon sağlanmış olur. Ancak, ortaya çıkabilecek

doküman ve bilgi yoğunluğu sebebiyle, ekosistemin işlemez hale gelmesini engelleyecek ayıklama ve güvenlik, gizlilik gibi sebeplerle paylaşılmaması gerekli bilgiler için filtreleme mekanizmalarının da kurulması gerekmektedir [9].

Modüler birimler sayesinde ekosistemde farklı aktörlerin birbirinden etkilenmeden ürün geliştirmesi sağlanmaktadır; ancak kesişen özellikler söz konusu olduğunda bu farklı aktörlerin birlikte çalışması gerekliliği gibi sorunların da dikkate alınması gerekmektedir. Bu probleme Cataldo [8] tarafından getirilen çözüm önerisi şu şekildedir: Arayüzlerin belirsizliği, karmaşıklığı ve ölçeklenebilirliği belirlenmeli ve bu bilgiler tüm ekosistemin erişimine açılmalı. Böylelikle, ekosistemin modülerliği ve farklı paydaşların beraber çalışabilirliği bir seviye daha artmış olacaktır.

2.3 Yazılım ekosistemlerde etkileşim, iletişim ve yönetim

Yazılım ekosistemi yönetimi, ekosistem aktörlerinin ve karşılıklı ilişkilerinin koordinasyonu ve yönetimi olarak tanımlanmaktadır. Kalite kriterlerini ve yazılım standartlarını paylaşmak, uzun vadeli planları paylaşmak, ortak bileşen havuzu oluşturmak, kalite standartlarını uygulatmak gibi yöntemler yönetim tekniklerine örnek olarak gösterilebilir [13]. Bu tekniklerin uygulanması ve geliştirilmesi ihtiyacının iki sebebi bulunmaktadır [10]: Ekosisteme yeni bir aktör katıldığında birden fazla paydaşı ilgilendiren arayüzleri, dokümanları tek kaynaktan öğrenebilmek, benzer şekilde ekosistemden ayrılan olduğunda da olası bilgi kayıplarını engellemek.

Bir yazılım ekosisteminde sürüm tarihleri, ürün yol haritası, proje hedefleri gibi temel ürün geliştirme kararları aktörler arası bilgi paylaşımı gerektirdiği için, karar verme süreci daha zorlayıcıdır. Bir ekosistem yöneticisi, hem aktörler arası, hem de başka sağlayıcılar, müşteriler vb. arasındaki iletişimin hızlı ve ucuz olmasını sağlamalıdır [14].

Brinkkemper [16] bir yazılım ekosisteminde yazılım operasyon bilgisi adını verdiği dokümanlarla ekosistemdeki ilişkilerin yönetilebileceğinden bahsetmiştir. Yol haritaları, ürün takvimleri, yeni sürümde bulunacak bileşenler, süreçlerdeki güncellemeler gibi konuların ekosistem aktörleriyle yazılım operasyon bilgisi yoluyla paylaşılabileceği önerilmiştir.

Seichter [10] ise, yazılım ekosistemlerinde farklı ekiplerce birlikte çalışılması gereken çok fazla ortak yapı olduğundan bahsetmiştir: Geliştirme araçları, mimari kararlar, ortak geliştirilecek bileşenler, ürünler, vb. Bu soruna getirdikleri öneri şu şekildedir: Paylaşılan bileşenlerin, ürünlerin, dokümanların birer ekosistem aktörü olarak kabul edilmesi önerilmiştir. Böylelikle, önemli kararların ve bilgilerin geliştiricilerde değil, ilgili yapının kendisinde saklanması ve tüm ekosisteme açık olması sağlanmaktadır. Bu sayede, ekosistemden bir aktör ayrıldığında olabilecek bilgi kayıplarının da önüne geçilmektedir.

3 IRIS Ekosistemi

IRIS, Aselsan'da üretilen sensör ve silah sistemleri için geliştirilmiş bileşen tabanlı bir yazılım ürün hattıdır [6]. IRIS kullanılarak, birim tipleri ve konfigürasyonları birbirinden farklı onlarca gömülü sistem yazılımı sistematik olarak üretilebilmektedir.

Geliştirilen sistematik konfigürasyon yaklaşımı [6] ile hem yazılımın üzerinde koşacağı platform ve işletim sistemi; hem de hangi tip birimlerin yaratılacağı belirlenmektedir. IRIS yazılım ürün hattında daha önce kullanılmamış bir birim ya da bileşenle çalışılması gerektiğinde ise, ihtiyaç duyulan yapıların ortak bileşen havuzuna eklenmesi ile ürün hattı genişletilmektedir.

IRIS kullanılarak hazırlanan sistem yazılımı sayısı arttıkça, sıfırdan geliştirilip ortak bileşenlere eklenmesi gereken bileşen sayısı da artmış ve IRIS geliştiricileri bu ihtiyaca zaman ve kaynak ayıramamaya başlamıştır. Bu soruna çözüm olarak altyüklenici kullanımının artırılması hedeflenmiş ve IRIS altyapılarının uygun olması sayesinde aynı anda birçok altyükleniciyle çalışmaya başlanmıştır.

Altyüklenicinin kullanımına yönelik yazılım ürün hattında bazı altyapıların sağlanmış olması ve geliştirilecek bileşenlerin ve son ürünlerin kalitesine yönelik birtakım kuralların önceden tanımlanmış olması sayesinde, bu çalışmalar Aselsan'ın kendi ekosistemine geçişi yönünde atılmış bir adım olarak değerlendirilebilir.

Bildirinin ilerleyen bölümlerinde, IRIS yazılım ürün hattının altyüklenici firmalar tarafından iki farklı seviyede kullanımı anlatılacaktır: IRIS için bileşen ve IRIS kullanılarak son ürün geliştirilmesi. Ardından, altyüklenici kullanımı konusunda yaşanan sıkıntılar ve çözüm önerileri aktarılacaktır.

3.1 IRIS için bileşen geliştirilmesi

IRIS yazılım ürün hattı, arayüz fonksiyonları ortaklanmış birim bileşenlerinden oluşmaktadır. Yapılan alan analiz çalışmaları ve birçok farklı bileşenin çeşitli projelerde kullanılmış olması sayesinde, bileşen arayüz fonksiyonları belli bir olgunluk seviyesine ulaşmış durumdadır.

IRIS bileşenlerinde, o bileşen tipine ait yapılan alan analizlerinde ortaya çıkan ortak ve muhtemel yetenekler "Özellikler" matrisinde toplanmıştır. Benzer şekilde, farklı proje isterlerine göre değişkenlik gösteren yetenekler de "Konfigürasyon" matrisi ile çözülmüştür [6].

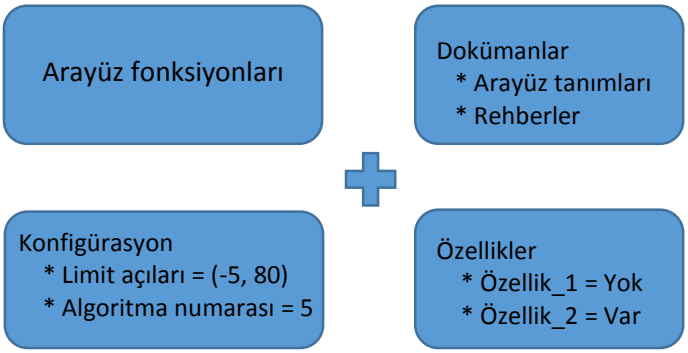
Aynı zamanda, ekip içerisinde aktif olarak kullanımda olan kodlama stilleri ve yazılım isimlendirme kuralları rehberleri bulunmaktadır. Ayrıca, MISRA C++ kodlama standardının da seçilmiş bir altkümesi kullanılmaktadır. Ekip tarafından geliştirilen programlar sayesinde ise, bileşenlerin kodlama stilleri ve isimlendirme kuralları rehberlerine ve MISRA altkümesine uygunlukları otomatik olarak ölçülebilmektedir. Bu ölçümler sonucunda bileşen kalite puanı hesaplanmaktadır [12]. Geliştirilen bileşenlerin belirli bir puan seviyesinin altında olması durumunda, IRIS ürün hattına eklenmesine izin verilmemektedir.

Bileşen arayüzleri ile birim ve proje seviyesindeki olası değişkenlikler yukarıda anlatıldığı gibi tanımlanmış durumdadır. IRIS ürün hattına eklenecek bileşenlerin uyması gereken kurallar da belirlenmiş ve dokümante edilmiş durumdadır. Bileşen geliş-

tirme yöntemlerinin var olması; ihtiyaç duyulduğunda, Aselsan haricindeki geliştiricilere bileşen geliştirme işinin verilmesinin önünü açmıştır.

Bu sayede Aselsan dışındaki geliştiriciler de, IRIS yazılım ürün hattı için, bileşen geliştirmeye başlamışlardır. Altyüklenicilerle paylaşılan dokümanlar için bkz. Şekil 1.

Bileşen geliştirme süreci boyunca, altyüklenicilerle haftalık düzenli toplantılar yapılmaktadır. Arayüzlerde -eğer varsa- altyükleniciler tarafından tespit edilen eksiklikler, bu toplantılarda değerlendirilmekte, gerekli görülürse arayüz fonksiyonları, konfigürasyon ya da özellik matrisleri güncellenebilmektedir.



Şekil 1. Bileşen geliştirme sürecinde paylaşılan dokümanlar

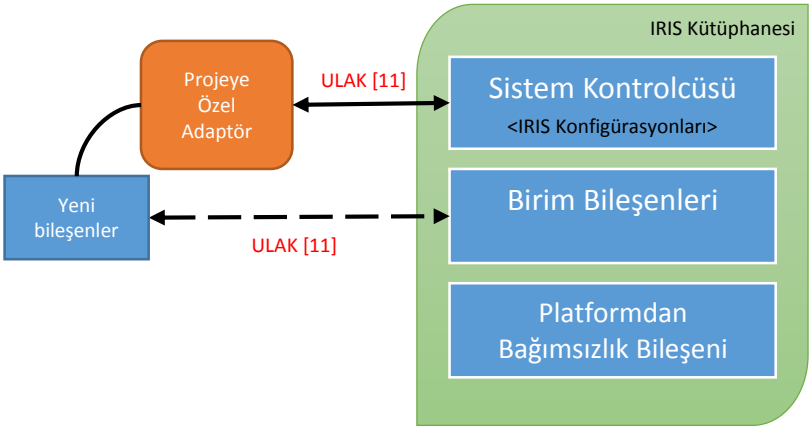
3.2 IRIS kullanılarak son ürün geliştirilmesi

Aselsan'da uzun yıllar boyunca edindiğimiz gömülü sistem yazılımı geliştirme deneyimimiz göstermiştir ki, gömülü yazılım bileşenlerinin sistem entegrasyonu için harcanan iş gücü, çoğunlukla bileşenin geliştirilme süresi ile kıyaslanabilir boyutta olmaktadır.

IRIS ekosistemi kapsamında birçok bileşen altyüklenici tarafından geliştirilse dahi, bileşenlerin sistem entegrasyonunun çoğunlukla Aselsan'daki geliştiricilerle beraber yapıldığı gözlenmiştir. Bu durumun nedenleri şu şekilde sıralanabilir: Aselsan dışındaki geliştiricilerin sistem tecrübesinin tek başına çalışmaya yeterli olmaması, sistem entegrasyon aşamasının bileşen geliştirildikten çok sonra başlaması, güvenlik nedenli kaygılar.

Bu sorunların önüne geçmek ve Aselsan dışındaki geliştiricilere daha fazla sorumluluk vermek amacıyla, örnek bir projede ürün sorumluluğunun altyüklenicilere verilmesi kararlaştırılmıştır. Böylelikle, IRIS ekosisteminin bir seviye daha genişlemesi hedeflenmiştir.

Şekil 2'de IRIS ekosistemindeki son ürün mimarisi gösterilmiştir. Ticaret sırları ve güvenlik gibi sebeplerle altyüklenicilerle IRIS ürün hattı kaynak kodu paylaşılmamıştır. Bunun yerine IRIS kütüphane olarak derlenmiş ve altyükleniciyle bu paylaşılmıştır. Altyükleniciyle paylaşılan diğer dokümanlar: Sistem kontrolcüsü dış arayüz fonksiyonları ve IRIS konfigürasyon bilgileridir.



Şekil 2. IRIS ekosisteminde son ürün mimarisi

Altyükleniciden IRIS ekosisteminde son ürün geliştirme sürecinde beklenenler: IRIS konfigürasyon altyapılarını kullanarak IRIS ürün hattını geliştirilecek proje ihtiyaçlarına göre ayarlaması, eğer IRIS bileşenleri içerisinde bulunmayan bir bileşen entegrasyonu gerekiyorsa bu bileşeni kodlaması, proje seviyesinde IRIS kontrolcüsünde bulunmayan bir yetenek gerekiyorsa da "projeye özel adaptör" geliştirip isterleri gerçeklemesi şeklinde sıralanabilir.

Şekil 2'de gösterildiği gibi, ihtiyaç halinde geliştirilecek projeye özel adaptör, IRIS sistem kontrolcüsüyle ULAK [11] adını verdiğimiz arayüz katmanı üzerinden haberleşmektedir. Benzer şekilde, yeni kodlanması gereken birim bileşenleri de ULAK yetenekleri sayesinde IRIS sistem kontrolcüsüne dışarıdan bağlanabilecektir. Altyüklenici tarafından geliştirilebilecek birim bileşenleri de Bölüm 3.1'de anlatıldığı gibi hazırlanacaktır ve yeterli olgunluğa ulaşmasının adından IRIS bileşen havuzuna eklenecektir.

Bu yöntemle, altyüklenicinin tüm sistem entegrasyon faaliyetlerini, IRIS yazılım ekibinin işçiliğinin minimum harcanmasını sağlayacak şekilde, tek başına yürütmesi hedeflenmektedir. IRIS ekibiyle yapılacak düzenli toplantılarla, IRIS ortak bileşenlerinde ya da arayüzlerde görülen değişiklik ihtiyaçları değerlendirilecek ve gerekli görülen arayüzler güncellenecektir.

3.3 Altyüklenici kullanımı sırasında karşılaşılan zorluklar

Bu bölümde IRIS yazılım ürün hattının, yoğun altyüklenici kullanımıyla beraber ekosisteme geçiş sürecinde karşılaşılan sıkıntılar özetlenecektir.

Öncelikle, karşılaşılan en temel problem "geliştirici seçimi" olmuştur. Aselsan'ın güvenlik ve emniyet gereksinimleri gereği bu süreç beklenenden uzun sürmüştür. IRIS ekosistemi dâhilinde birlikte çalışılan altyüklenici sayısı arttıkça, bu konunun sorun olmaktan çıkacağı değerlendirilmektedir.

Ekosistem kapsamındaki bir diğer sorun da, altyüklenici firmaların yapacağı çalışmalar için yapılan işçilik saati tahminlerinin, ekosistem kapsamındaki bu çalışmaların

ilk kez yapılacağı için öngörülemeyişidir. Bu konunun da ekosistem çalışmaları devam ettikçe, ortadan kalkacağı düşünülmektedir.

Ekip olarak görülen bir diğer sıkıntı da, süreç sırasında kullanılan birçok özelliğin dokümantasyonun eksik olmasıdır. Birçok bilginin dokümante edilmediği ekip dışı geliştiricilerin bu altyapıları kullanması sayesinde fark edilmiş ve hızlı bir şekilde bu eksiklerin kapatılması çalışmalarına başlanmıştır.

Yazılım süreçleri konusunda da güncelleme ihtiyacı duyulmuştur. Birçok yazılım geliştirme adımı hem Aselsan tarafında, hem altyükleniciler tarafında yapılmaktadır: Konfigürasyon yönetimi, gereksinim yönetimi vb. Bu gibi durumlar için, altyüklenicilerle ortak erişilen alanlarda tek elden bu işlerin yönetilmesi hedeflenmektedir.

4 Sonuç ve gelecek için planlar

Bu çalışmada Aselsan'ın askeri ürünler geliştirmesinden dolayı dış geliştiricilere kapalı bir yazılım ürün hattı olan IRIS'in, çeşitli ve güvenli seviyelerde altyüklenicilere açılması yoluyla IRIS ürün hattının IRIS ekosistemine dönüştürülmesi çalışmaları anlatılmıştır.

Bundan sonraki süreçte, altyüklenici çeşitliliğinin artırılması hedeflenmektedir. IRIS ekosistemi dokümantasyonunun ve ürün geliştirme rehberlerinin iyileştirilmesi ve bu sayede ekosistem kalitesinin yükseltilmesi amaçlanmaktadır.

Teşekkür

Saygıdeğer hocam Prof. Semih Bilgen'e yazılım ekosistemi kavramıyla tanıştırdığı için teşekkürlerimi sunarım.

IRIS yazılım ürün hattını birlikte geliştirdiğimiz ve ekosisteme dönüştürmekte olduğumuz, Aselsan Savunma ve Sistem Teknolojileri (SST) Sektör Başkanlığı Hava Savunma Silah Sistemleri ve Görüntü İşleme (HSSS-Gİ) yazılım ekip arkadaşlarıma özverili çalışmaları ve verdikleri destekten dolayı teşekkür ederim.

Kaynaklar

1. Bosch, Jan, ve Bosch-Sijtsema, Petra. "From integration to composition: On the impact of software product lines, global development and ecosystems." Journal of Systems and Software 83.1 (2010): 67-76.
2. Bosch, Jan. "The challenges of broadening the scope of software product families." Communications of the ACM 49.12 (2006): 41-44.
3. Hanssen, Geir K. "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory." Journal of Systems and Software 85.7 (2012): 1455-1466.
4. van Gurp, Jilles; Prehofer, Christian; ve Bosch, Jan. "Comparing practices for reuse in integration-oriented software product lines and large open source software projects." Software: Practice and Experience 40.4 (2010): 285-312.

5. Bosch, Jan. "From software product lines to software ecosystems." Proceedings of the 13th international software product line conference. Carnegie Mellon University, 2009.
6. Deniz, Berkhan; Öztas, Gökhan ve Çinar, Soner. "Askeri bir Gömülü Yazılımın Bileşen Tabanlı Bir Mimari Kullanılarak Refaktör Edilmesi." UYMS. 2015.
7. Hanssen, Geir Kjetil. "Opening up software product line engineering." Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering. ACM, 2010.
8. Cataldo, Marcelo; Herbsleb, James D. Architecting in software ecosystems: interface translucence as an enabler for scalable collaboration. In: Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ACM, 2010. p. 65-72.
9. Bosch, Jan. "Architecture challenges for software ecosystems." Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ACM, 2010.
10. Seichter, Dominik, et al. "Knowledge management in software ecosystems: software artefacts as first-class citizens." Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ACM, 2010.
11. Karasoy, Burcu, ve Çinar, Soner. "Dağıtık Sistemler için Haberleşme Otomasyon Ara Katmanı: ULAK." UYMS. 2014.
12. Deniz, Berkhan, ve Çinar, Soner. "Bileşen Kalitesi Ölçümünde Statik Kod Analizi Yaklaşımı." UYMS. 2014.
13. Jansen, Slinger; Brinkkemper, Sjaak; Finkelstein, Anthony; ",Business network management as a survival strategy: A tale of two software ecosystems,"Proccedings of the 1st International Workshop on Software Ecosystems, CEUR-WS",34-48,2009.
14. Boucharas, Vasilis; Jansen, Slinger; ve Brinkkemper, Sjaak. "Formalizing software ecosystem modeling." Proceedings of the 1st international workshop on Open component ecosystems. ACM, 2009.
15. van den Berk, Ivo; Jansen, Slinger; ve Luinenburg, Lútzen. "Software ecosystems: a software ecosystem strategy assessment model." Proceedings of the Fourth European Conference on Software Architecture: Companion Volume. ACM, 2010.
16. van der Schuur, Henk; Jansen, Slinger; ve Brinkkemper, Sjaak. "The power of propagation: on the role of software operation knowledge within software ecosystems." Proceedings of the International Conference on Management of Emergent Digital EcoSystems. ACM, 2011.