Atış Kontrol Yazılımlarında Ürün Hattı Yaklaşımının Uygulanması

Adnan Kalay

ASELSAN A.Ş. SST-GGZYTM P.K.1 06172, Yenimahalle/Ankara, Türkiye
akalay@aselsan.com.tr

Özet. Yazılım geliştiren birçok firma, içerdikleri özelliklerin değiştirilmesiyle birbirinden ayrılan fakat temelde birbirine çok benzeyen yazılım aileleri üretmektedir. Bu yazılım ailelerinin Yazılım Ürün Hattı yaklaşımıyla geliştirilmesi başta maliyet, kalite ve markete çıkış süresi olmak üzere pek çok alanda önemli avantajlar getirmektedir. Merkezinde sistematik yeniden kullanımın yer aldığı bu yaklaşım *Alan Mühendisliği* ve *Uygulama Mühendisliği* olmak üzere iki temel geliştirme fazından oluşmaktadır. Alan mühendisliği fazında ilgili yazılım ailesine yönelik referans mimari oluşturulmakta ve bu mimariyi temel alan temel varlıklar, örneğin; yeniden kullanılabilir bileşenler, geliştirilmektedir. Uygulama mühendisliği aşamasında ise alan mühendisliği aşamasının çıktıları olan referans mimari ve yeniden kullanılabilir bileşenlerin kullanımı için gerekli konfigürasyonların yapılması ve projeye özel yapıların eklenmesiyle proje yazılımları ortaya çıkarılmaktadır.

Bu bildiride, ASELSAN SST Sektör Başkanlığı bünyesindeki Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü bölümünde izlenen yazılım ürün hattı yaklaşımı çerçevesinde gerçekleştirilen uygulama mühendisliği çalışmaları anlatılacaktır. Bu kapsamda alan mühendisliği gerçekleştirilmiş atış kontrol yazılımları (AKY) alanından pilot bir yazılım ailesinin oluşturulması ve bu aileye mensup bir yazılım bireyinin üretilmesi sırasında gerçekleştirilen gereksinim analizi, yazılım gerçekleme ve test çalışmaları aktarılacaktır. Son olarak, mevcut yaklaşımla ilgili tespit edilmiş eksiklikler belirtilerek iyileştirme önerileri sunulacaktır.

Anahtar kelimeler: Yazılım Ürün Hattı, Sistematik Yeniden Kullanım, Alan Mühendisliği, Uygulama Mühendisliği, Yazılım Ailesi, Birey Yazılım

1 Giriş

Yazılım ürün hattı (YÜH) yaklaşımında yüksek kaliteye sahip yazılım ürünlerini klasik yaklaşıma göre çok daha düşük maliyetlerle (süre, işçilik vb.) geliştirmek hedeflenmektedir. Bu amaçla YÜH yaklaşımı, klasik yaklaşımda da fırsatçı şekilde (İng. *opportunistic reuse*) kullanılan yeniden kullanım yönteminin çok daha sistematik bir şekilde (İng. *systematic reuse*) kullanılması ve sadece yazılım kodlarında değil, her türlü yazılım varlığında (gereksinim, tasarım, kod, test tanımı,

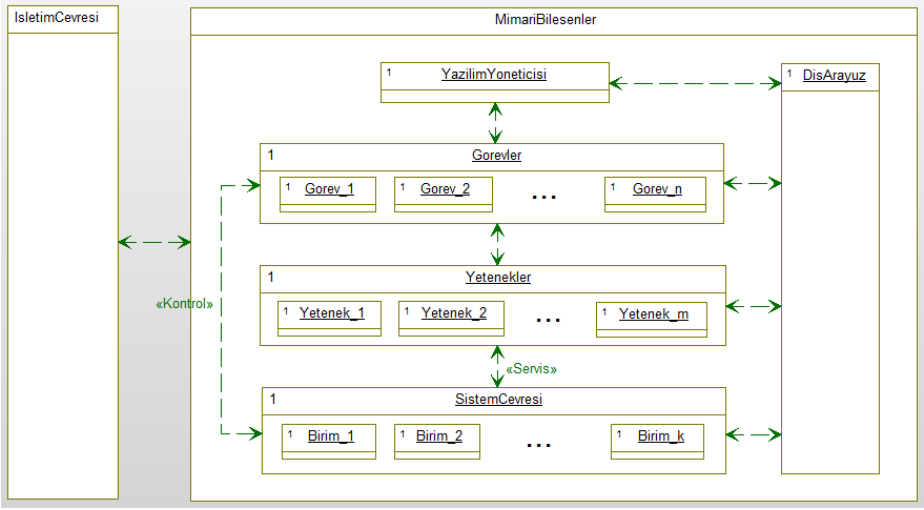
teslim dokümanları vb.) uygulanmasını temel alır [1]. Bu yaklaşım literatürde çeşitli alt yaklaşımlarla karşımıza çıksa da, bunlardan en çok kabul göreni özellik-yönelimli olanlardır [2, 3, 4, 5, 6, 7, 8, 9, 10]. YÜH yaklaşımında alan mühendisliği ve uygulama mühendisliği olmak üzere iki ana çalışma safhası yer almaktadır. Alan mühendisliği çalışmasının ilk adımını alan analizi oluşturur. Belli bir alana özel yazılımlar topluluğunun yazılım ürün hattı yaklaşımıyla geliştirilmeye uygun olup olmadığının tespit edilmesi amacıyla öncelikle alan analizi çalışması yapılır. Bu analiz sırasında alandaki ortaklık ve farklılıklar tespit edilerek o alana ait özellik* modeli (İng. *feature model*) oluşturulur. Ortaya konan özellik modelindeki ortak özelliklerin yeterli oranda olması ve değişkenliklerin de yönetilebilir bulunması bu alanda bir yazılım ailesinin varlığına ve dolayısıyla YÜH yaklaşımının uygulanması fırsatına işaret etmektedir. Alan mühendisliğinin ilerleyen aşamalarında özellik modelinden hareketle ilgili yazılım ürün hattına özel referans mimari(ler) ve yeniden kullanılabilir bileşenler geliştirilir [3, 4].

YÜH yaklaşımının ikinci safhası olan uygulama mühendisliği fazında ise yazılım ürününü ortaya çıkarma çalışmaları yer almaktadır. Bu doğrultuda öncelikle uygulama gereksinim analizi yapılarak ürün hattı gereksinimlerine ilgili yazılım ürününe özel gereksinimler de eklenir [4]. Bu analiz sırasında ürüne özgü değişkenlik noktaları (İng. *variability point*) ve değişkenlerin (İng. *variant*) seçilmesiyle değişkenliğin bağlanması sağlanır ve alan analizinde ortaya konmuş özelliklerden ilgili olanların seçimi tamamlanır. Alan mühendisliği safhasında birden fazla mimari ortaya konmuşsa uygun olanın seçimi yapılır. Bu mimariyi temel alan ürün ailesi yazılımı ilgili proje yazılımı özellikleri doğrultusunda yapılandırılarak karşılık gelen yeniden kullanılabilir bileşenlerin yazılım ürününe dâhil edilmesi sağlanır. Ayrıca yeniden kullanılabilir bileşenlerden uygun olanları da yine desteklenen özellik kümesine göre yapılandırılır. Son olarak ürüne özel yazılım parçalarının da eklenmesiyle proje yazılımı ortaya çıkarılmış olur.

Bu bildiride, alan mühendisliği daha önceden özellik-yönelimli ve model-güdümlü olarak gerçekleştirilmiş bir yazılım ürün hattı için yapılan uygulama mühendisliği çalışmaları pilot olarak seçilmiş bir AKY ailesi özelinde aktarılmaktadır [11]. Bu kapsamda alana ait bir yazılım ailesinin ortaya konması ve bu aileden birey yazılımların üretilmesi sürecindeki uygulama gereksinim mühendisliği, yazılım gerçekleme ve test safhalarından bahsedilmektedir. Bildiride bahsi geçen ürün hattı ve aile yazılımı hakkında genel bir fikir vermesi amacıyla öncelikle kısa bir bilgilendirme yapılacaktır. Takip eden bölümlerde uygulama mühendisliği kapsamında sırasıyla gereksinim yönetimi, yazılım gerçeklemesi ve test deneyimleri sunulup ardından mevcut yaklaşımla ilgili tespit edilen eksiklikler ve iyileştirme önerileri aktarılacaktır. Son bölümde ise yaklaşımla ilgili genel bir değerlendirme yapılacaktır.

2 AKY Ürün Hattı ve Pilot Aile Yazılımı

AKY ürün hattının oluşturulma süreci 2007 yılı başında bölüm bünyesinde bir çalışma ekibi oluşturularak ve akademik destek alınarak başlamış, 2009 yılı sonunda hazırlanan referans mimari raporuyla tamamlanmıştır. Bu süreçte yeniden kullanım ve YÜH odaklı literatür taramaları, silah sistemleri ve atış kontrol yazılımları alanına özel araştırmalar ve mevcut ASELSAN çalışmalarının incelenmesiyle pek çok ara rapor da hazırlanmıştır. Referans mimari raporunda AKY ürün hattı referans mimarisi alınan mimari kararlarla birlikte tanımlanırken; bileşen gereksinimleri ve arayüzleri, mimari birimler arası sorumluluk paylaşımı ve mimari uyumluluk kontrol listesi gibi ek dokümanlarla da desteklenmiştir. Ayrıca ürün hattından çıkarılacak yazılım ailelerinin işçilik tahminini kolaylaştırmak amacıyla, işçilik öngörülerinin referans mimari ekseninde bileşen tabanlı olarak yer aldığı ve buradan toplam işçiliğin kestirilmesine olanak sağlayan bir işçilik tahmini şablonu da ortaya konmuştur. Çalışmada ürün hattının özellik-yönelimli olarak ortaya konması benimsenmiş ve FORM yaklaşımı temel alınarak çalışmalar yürütülmüştür [3]. Oluşturulan ürün hattına ait özellik modeli hazırlanan ara raporlardan biri olan silah sistemleri alanı analiz raporunda tariflenmiştir. Bu raporda tanımlanan özellik modeli Şekil 1'de gösterilen silah sistemleri referans mimari modeline uygun olarak görev seviyesinden başlamak üzere en üst seviyede görevler, bir alt seviyede yetenekler ve en alt seviyede ise servisler olacak şekilde oluşturulmuştur.



Şekil 1. Silah Sistemleri Referans Mimari Modeli

Özellik modellerinde belirlenen kapsamın genişliğine göre özellik sayısı değişkenlik göstermektedir. Hazırlanan raporda belirtilen özellik modelinde 71 özellik yer almıştır. Görevlerin gerek içerik gerekse sayı bakımından projeden projeye en çok değişkenlik gösteren kısımlar olması sebebiyle özellik modelinin ortaya konması sırasında tamamen tespit edilemeyeceği değerlendirilerek önemli olanların yer alması

sağlanmış, bazı görevlerin yetenek detayları ise açılmamıştır. Benzer şekilde yeteneklerin kullandığı temel servislere de örnekler verilirken bu servislerin arttırılması ve detaylandırılmasının mümkün olduğu ifade edilmiştir. Modelde özellikle görevlerin gerçeklenmesine alt yapı sağlayan yeteneklerin tamlığının sağlanması amaçlanmıştır.
Bu sayede projeler özelinde yeni tanımlanan görevlerin kısa sürede yazılımlara eklenmesi hedeflenmiştir.

Bu ürün hattından çıkarılan pilot aile yazılımının kökenleri 2006 yılına dayanmaktadır. Bahsi geçen yazılımının tamamen ürün hattı yaklaşımıyla hazırlanarak teslim edilmesi ise 2011 yılında gerçekleşmiştir. Ürün hattı yaklaşımına geçmeden önce
yazılımın her dağıtımı için ayrı konfigürasyon birimi hazırlanmaktayken, YÜH yaklaşımıyla birlikte tüm dağıtımların tek bir konfigürasyon birimi çatısı altında çıkarılması amaç edinilmiştir. Ürün hattı yaklaşımına geçmeden önceki 5 yıllık süreçte bu yazılımın 9 farklı dağıtımı yapılmıştır. Ürün hattı geçişini takiben ise bu aileden resmi olarak 12 farklı yazılım ürününün oluşturulması sağlanmıştır. Dağıtım sayılarındaki artış ilk bakışta çok yüksek gözükmese de bu noktada asıl kazanım paralel yazılım geliştirmede elde edilmiştir. Aynı geliştirme ekibi YÜH sonrasında önceki döneme göre çok daha fazla sayıda farklı projeyi paralel olarak geliştirme başarısı göstermiştir. Bu sayede ekip verimliliğinde önemli artış gözlenmiştir. Ayrıca YÜH öncesi dağıtıldığı belirtilen yazılımların önemli bir bölümü, ortak konfigürasyon biriminden çıkarılmamış olmasına karşın YÜH geçiş döneminde çıkarılması sebebiyle hazırlanan referans mimari ve yeniden kullanılabilir bileşenlerin önemli oranda kullanımı ile geliştirilmiştir. YÜH yaklaşımıyla geliştirilen AKY aile yazılımının ilk dağıtımı 340.000 satır kod büyüklüğüne sahipken zamanla yeni özelliklerin eklenmesiyle beraber mevcut durumda 500.000 satır kod büyüklüğüne ulaşmıştır. Gelinen noktada aile yazılımının çeşitli konfigürasyonlarında yer alan sadece değişken özelliklerin sayısının 50'nin üzerinde olması, AKY ürün hattının raporda belirtilenden oldukça fazla özellik sayısına ulaştığını göstermektedir. Gelecek dönemde yapılması planlanan alan mühendisliği çalışmaları arasında ürün hattı özellik modelinin güncellenmesi de yer almaktadır.

3 Gereksinim Yönetimi

Ürün hattı yazılımlarına ait gereksinimler genel olarak; olduğu gibi kullanılan gereksinimler, parametrik kullanılan gereksinimler (metinsel tanımlamaları sabit olan ve yazılım aileleri/bireyleri arasında sadece içerdikleri sayısal değerleri değişen gereksinimler), ürün konfigürasyonuna bağlı gereksinimler ve ürüne özel gereksinimler olarak ifade edilmektedir [12]. Bunlardan ilk üçü alan gereksinim mühendisliğinde ortaya çıkarılmaktadır. Uygulama gereksinim mühendisliğinde ise ortak gereksinimlerin toplanması, konfigürasyona bağlı gereksinimlerin seçilmesi, parametrik gereksinimlerin netleştirilmesi ve son olarak ürüne özel gereksinimlerin eklenmesi sağlanır. Çalışmada pilot olarak seçilen AKY ailesinin gereksinim yönetimini özellik-yönelimli bir şekilde gerçekleştirebilmek için, kullanılan

gereksinim yönetim aracının sütun ve tip tanımlama alt yapılarından faydalanılmıştır. Özellik ağacında yer alan ve değişkenlik arz eden özelliklerin her biri ayrı birer tip olarak tanımlanmıştır. Gereksinim tablosuna bu özelliklerin bir veya daha fazlasının yer alabildiği *Gerekli Özellikler* (İng. *Required Features*) ve *Özellik Bağımlılığı* (İng. *Feature Dependency*) olmak üzere iki ayrı sütun eklenerek özellik tabanlı gereksinimlerin ifade edilebilmesi sağlanmıştır. Belli bir özelliğe veya özellik kümesine bağlı gereksinimlerin "Gerekli Özellikler" sütununda bu özellikler belirtilmektedir. Bir gereksinimin varlığı direk olarak bir özelliğe bağlı olmasa da o özelliğin varlığından etkileniyorsa, örneğin; bu gereksinimin test tanımları oluşturulurken belirtilen özelliğin varlığına göre test tanımı içeriği değişiyorsa, bu özelliklerin de "Özellik Bağımlılığı" sütununda yer alması sağlanmıştır. Belirtilen sütunlarda herhangi bir özellik bulunmayan gereksinimlerse her üründe kullanılan ortak gereksinimler olarak görülmektedir. Şekil 2'de bu doğrultuda belirlenen örnek gereksinim maddeleri görülmektedir.

ID	YAZILIM GEREKSİNİM ÖZELLİKLERİ	Required Features	Feature Dependency
AKBYGÖ 162	Yazılım, hedef mesafe bilgisi olarak, kullanıcı arayüzünden girilen mesafe bilgisini aktif hedef mesafesi olarak kullanabilecektir.		
AKBYGÖ 164	Yazılım, Video Hedef İzleme işlemi sırasında sistemin monte edildiği platformun kerteriz, hız ve rota bilgileri de elde edilebiliyorsa, hedefin hakiki kerteriz, hakiki rota ve hız bilgilerinin kullanıcı arayüzünden gösterilmesini sağlayacaktır.	SUPPORTS_TARGET_INFO	SUPPORTS_LAZER_RANGE_FINDER SUPPORTS_LN270
AKBYGÖ 353	Yazılım, sesli kullanıcı arayüzü uyarılarının (bazer gibi) kullanımına izin verilip verilmemesi seçimini kullanıcı arayüzünden sağlayacaktır.	SUPPORTS_ISIB	

Şekil 2. Özellik Tabanlı Gereksinimler

Bu çalışmada parametrik gereksinim tanımlaması kullanılmamıştır. Tamamen ürüne özgü gereksinimlerin belirtilmesi içinse *Sistem* sütunu tanımlanmıştır.

4 Yazılım Gerçeklemesi

Uygulama mühendisliği aşamasında yazılım geliştirme çalışmaları olarak [11]'de belirtilen alan mühendisliği çıktıları olan referans mimari ve yeniden kullanılabilir bileşenlerin kullanıma alınması sağlanmıştır. Bu doğrultuda ilgili AKY ailesine ait bir yazılım bireyini ortaya çıkarmak amacıyla referans mimari modelinde belirtilen bileşenler alan tasarımında tanımlanan mimari kurallara uygun olarak ortaya konmuştur. Bu bileşenlerden bir kısmı alan mühendisliği çıktıları olan yeniden kullanılabilir bileşenlerken, bir kısmı ise geliştirilmesi uygulama mühendisliği aşamasına bırakılmış bileşenlerdir. Silah sistemleri referans mimarisinde (SSRM) yer alan bileşenler Şekil 1'de belirtilen modelde gösterilmektedir. Uygulama yazılımını ortaya çıkarmak için bu modelde kabaca yer alan *İşletim Çevresi*, *Sistem Çevresi*, *Yetenekler*, *Görevler*, *Dış Arayüz* ve *Yazılım Yöneticisi* bileşenlerinin tanımlı kurallar çerçevesinde bir araya getirilmesi gerekmektedir.

İşletim çevresi bileşeni, yazılımı çalıştığı ortamdan (donanım, işletim sistemi vb.) soyutlarken bir yandan da diğer yazılım bileşenlerine kullanışlı işlevler sunmaktadır.

Bu konuda literatürde yapılmış bazı çalışmalar bulunmaktadır [13, 14, 15, 16]. Bu bileşen belli bir yazılım ailesi üyeleri için ortakken farklı yazılım ailelerinin farklı ortamlarda çalıştığı durumlar için değişkenlik göstermektedir. Bu nedenle yeni bir yazılım ailesi referans mimari üzerine kurgulanırken işletim çevresi gereksinimi yeniden kullanılabilir bileşen havuzundan karşılanabiliyorsa buradan alınmakta, yeni bir işletim çevresi tanımlanıyorsa bu aile için yeni bir bileşen geliştirilip sonraki muhtemel kullanımlar için havuza dâhil edilmektedir. İşletim çevrelerinin analizi sonucu farklı çevreler arasında da birçok ortak özellik olduğu görülmüş ve bu bileşenlerin geliştirilmesi amacıyla ortak bir işletim çevresi projesi oluşturulmasına karar verilmiştir.

Sistem çevresi birimleri, kullanıcılarına çeşitli servisler (atış servisi, balistik servisi, sürücü servisi vb.) sunan bileşenlerdir. Bu bileşenlerin yazılıma dâhil edilmesi aşamasında öncelikle bileşen havuzunda mevcut olanlar alınmaktadır. Havuzda yer almayan sistem çevresi bileşenleri ise öngörülen bir yeniden kullanım potansiyeli olduğunda geliştirilip havuza aktarılmakta, diğer durumda projeye özel olarak geliştirilmektedir.

Yetenek bileşenleri alan mühendisliği aşamasında geliştirilmiş olup atış kontrol alanında gerekli görülen ortak ve üst seviye kabiliyetleri (atış yönetimi, hedef yönetimi, platform yönetimi vb.) sunmaktadır. Bu nedenle bu bileşenler yeniden geliştirilmemekte, yeniden kullanılabilir bileşen havuzundan kullanıma alınmaktadır. Yetenek bileşenleri sundukları üst seviye kabiliyetleri yerine getirmek adına sistem çevresinden aldıkları servisleri kullanmaktadır. Yeniden kullanım oranı en yüksek olan yazılım parçaları yetenek ve sistem çevresi bileşenleridir. Bu bileşenlerin farklı yazılım ailelerinde ortak kullanımını mümkün kılmak adına kontrol arayüzleri de sunması sağlanmıştır. İlgili yazılım ürünü, bu arayüzlerden gerekli konfigürasyonları yaparak bileşeni kendi kullanım durumuna uygun hale getirebilmektedir.

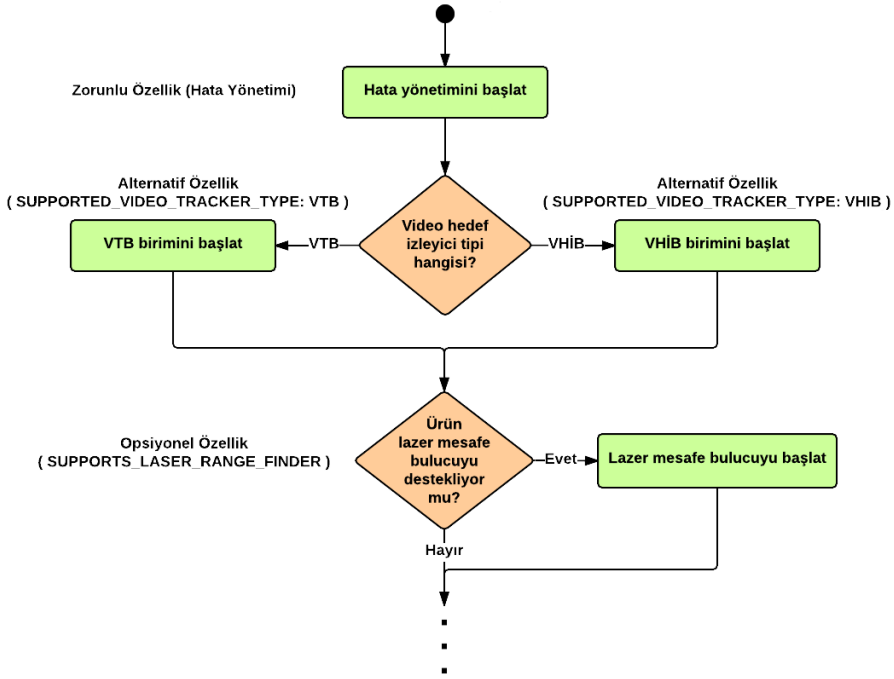
Görev bileşenleri en üst seviye senaryo işleticileri olup koşturduğu senaryoya göre bir veya daha fazla yetenek bileşeninden faydalanabilmektedir. Bu bileşenler daha çok projeye özgü olmakla ve proje yazılımına özel olarak geliştirilmekle beraber ürün hattına ortak olduğu düşünülen görev bileşenleri yeniden kullanıma dâhil edilmektedir. Modelde kabaca gösterilen dış arayüz bileşeni, kullanıcı arayüzü veya yazılımın bir diğer yazılımla ilişkide olduğu bir arayüzü (komuta kontrol sistemi, tank kontrol sistemi vb.) ifade etmektedir. En üst düzeyde bulunan yazılım yöneticisi ise sistem yazılımının çalışma modlarını belirler ve yazılım akışının bu modların gereksinimlerine uygun olarak işlemesini sağlar. Dış arayüz ve yazılım yöneticisi bileşenleri de uygulama mühendisliği aşamasında ilgili yazılım ailesine özgü geliştirilmektedir.

Çalışılan ortamla ilişkisi en fazla olan yazılım parçaları sistem çevresi olduğundan işletim çevresini en yoğun kullanan grup bu bileşenler olmaktadır. Bununla birlikte görevler, yetenekler gibi diğer yazılım parçaları da ihtiyaçları doğrultusunda işletim çevresinin desteğini almaktadır. Bu amaçla işletim çevresi tüm yazılım parçalarının erişebileceği ortak bir çalışma ortamı olarak yazılıma entegre edilmektedir. SSRM bileşenlerinin birbiriyle entegrasyonu içinse UML portları ile [17]'de belirtilen ve kodu otomatik üretilen yazılım veri yolu (İng. *Software Bus*) kullanılmaktadır. Bileşenlerin haberleşmesinde *Yetenek Veri Yolu* ve *Kontrol Veri Yolu* olarak 2 yazılım

parçası hazırlanmıştır. Yetenek bileşenlerinin sunduğu yeteneklere erişim yetenek veri yolu üzerinden yapılmaktadır. Yetenekler ve sistem çevresinin konfigürasyonları ise kontrol veri yolu vasıtasıyla yerine getirilmektedir. Yetenekler ihtiyaç duydukları servisleri sistem çevresine portlar aracılığıyla direk olarak bağlanıp elde etmektedir.

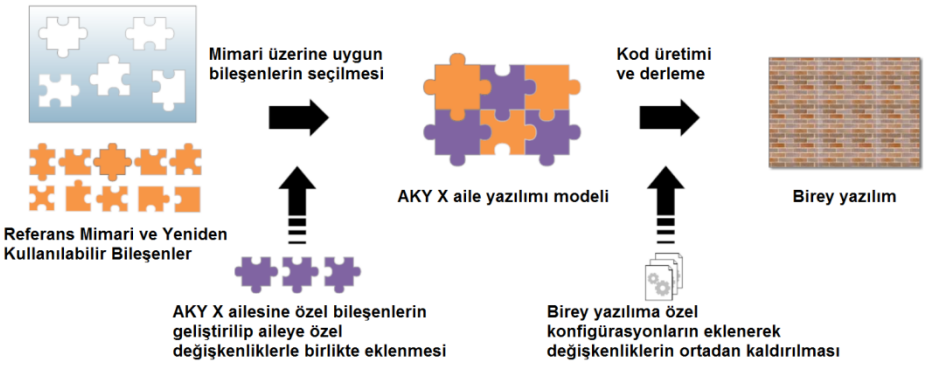
Literatürde pek çok çalışmada da belirtildiği üzere ürün hattı sürecinde uygulama mühendisliğinden alan mühendisliğine geri bildirim mekanizmaları bulunmaktadır [3, 18, 19]. Buna göre uygulama safhasında elde edilen tecrübeler doğrultusunda tüm alan mühendisliği çıktılarının güncellenmesi gerekebilir. Başlangıçta ilgili yazılım bireyine ya da ailesine özel olarak geliştirilen sistem çevresi veya görev bileşenlerinin zamanla başka yazılım ailelerince de kullanılmaya başlaması durumunda bu bileşenlerin konfigürasyon kontrolü altında tutulan yeniden kullanılabilir bileşen havuzuna dâhil edilmesi bu duruma bir örnek olarak verilebilir.

Bölümümüzde geliştirilen ürün hattından farklı AKY aileleri çıkarılmaktadır. Bu bağlamda uygulama yazılımlarını ortaya çıkaracak çalışmaların kapsamına hem ilgili AKY ailesinin ortaya konması hem de bu aileye ait birey yazılımın oluşturulması girmektedir. Bu gözle bakıldığında SSRM bileşenlerinden yetenekler ile yeniden kullanılabilir sistem çevresi ve görevler bileşenlerinden uygun olanları yazılım aileleri tarafından ortak olarak kullanılmaktadır. Yeniden kullanılabilir olmayan görevler, dış arayüz elemanları ve yazılım yöneticisi ise ilgili yazılım ailesine özel kalmaktadır. Bu yazılım ailesinin bireyleri arasındaki farklılıklar ise alan analizinde belirlenmiş olan değişken özelliklerin her bir birey için ayrı hazırlanmış konfigürasyon dosyalarında tutulup yazılım başlangıcında buradan okunması ile sağlanmaktadır. Yazılım açılış senaryosunda Şekil 3'te örneklendiği gibi ilgili ürüne ait özellik kümesi doğrultusunda gerekli özelleştirmeler yapılmaktadır. Bireysel özellikler normal çalışma senaryosunun yürütülmesine de etki etmektedir. Bu sayede yazılım ailesine özel, fakat aile bireyleri arasında ortak olarak kullanılan bileşenler ürün konfigürasyonuna göre işlev kazanmaktadır. Yetenek bileşenlerinin aileler arası yeniden kullanımını daha da kolaylaştırmak adına, bileşenlerin kontrol arayüzlerine ek olarak konfigürasyon dosyaları ile yapılandırılma alt yapısı da getirilmiştir. Bu dosyaların içerikleri de aynı aile bireyleri arasında farklılık gösterebilmekte ve yazılım bireylerinin üretilmesine katkı sağlamaktadır.



Şekil 3. Ürün Konfigürasyonu ile Açılış Senaryosunun Belirlenmesi

Bu bilgiler ışığında, ilgili yazılım ailesinin çalışır hali şu adımlarla ortaya çıkarılmaktadır. Öncelikle mimari arayüzleri içeren ürün hattı mimarisi ve işletim çevresi bileşeni kütüphane olarak ilgili yazılım ailesi modeline eklenmektedir. Yeniden kullanılabilir yetenek ve sistem çevresi bileşenleri ise gri kutu yeniden kullanım ile bileşen havuzundan referans olarak (salt-okunabilir biçimde) alınmaktadır [20]. Bu bileşenlerin kullanım oranları aileden aileye değiştiği için kütüphane üretimi, aile yazılımına alındıktan sonra toplu olarak gerçekleştirilmekte ve bu şekilde kullanıma alınmaktadır. Yeniden kullanılabilir nitelikteki görev bileşenleri de yine referans yoluyla dâhil edilmekte, geriye kalan yazılım bileşenleri ise en çok değişen yazılım parçaları olup yapılan model-tabanlı değişikliklerle yeniden kod üretimi ve derleme fazlarına girerek yazılım oluşturma sürecinin son halkasını oluşturmaktadır. Nihai durumda üretilen çalışır yazılım, aslında bir aile yazılımı mahiyetindedir. Yazılım, ailenin ilgili bireyine özel nitelik kazanması amacıyla sisteme özel ürün ve yetenek konfigürasyon dosyalarıyla birlikte paketlenmekte ve sistemlere bu paket yazılım yüklenmektedir. Bu sayede ilgili aile yazılımı açılış senaryosunda ve sonrasında sisteme özel konfigürasyonları yaparak birey yazılım işlevini kazanmaktadır. Şekil 4'te SSRM ve yeniden kullanılabilir bileşenlerden birey yazılımın üretilme aşamaları kabaca gösterilmektedir.



Şekil 4. YÜH ile Birey Yazılımın Üretimi

5 Yazılım Testi

YÜH yaklaşımında testin önemi daha da artmaktadır. Yeniden kullanımın en yoğun şekilde uygulandığı bu süreçte hatasız bileşenlerin yeniden kullanımı ne kadar maliyet kazancı sağlarsa, hatalı olanların kullanımı da aynı oranda kayıplara neden olacaktır. Temel varlıkların doğrulandığı alan mühendisliği testi belirtildiği gibi büyük öneme sahip olmakla birlikte bu bölümde bildirinin kapsamına uygun olarak uygulama mühendisliği test çalışmalarından kısaca bahsedilmektedir. [21]'de de belirtildiği üzere alan mühendisliği testleri doğruluk ve bütünlük odaklı yürütülürken, uygulama mühendisliği testlerinde ise gereksinimlere olan uygunluk ön plana çıkmaktadır. Geliştirilen yazılım aileleri ilgili test mühendisliği ekibi tarafından yine YÜH yaklaşımına uygun olarak yeterlilik testlerine tabi tutulmaktadır. Bu amaçla öncelikle yaklaşıma uygun test planı, sonrasında ise AKY ailesinin sahip olduğu *zorunlu*, *alternatif* ve *opsiyonel* özelliklerin karşılık geldiği yazılım gereksinimlerini test eden test tanımları hazırlanmaktadır. Hazırlanan tanımlardan ilgili gereksinimlere bağlantı kurulmasıyla yeniden kullanıma uygun bu varlıklar arasında izlenebilirlik sağlanmaktadır. Test tanımlarının hazırlanıp derlenmesini takiben ilgili yazılım ailesindeki tüm bireylerin özelliklerini kapsayacak ve bu hedefi en az iş gücüyle sağlayacak şekilde çeşitli test konfigürasyonları oluşturulmaktadır. Bu testlerde ana odak aile bireylerinin bağımsız testi yerine tüm ailenin doğrulanması olduğundan, test konfigürasyonları 4. bölümde bahsedilen bireysel ürün konfigürasyonlarına bire bir karşılık gelmeyebilir. Ailedeki ortak gereksinimlerin ve test konfigürasyonlarında belirtilen değişkenliklerin test edilmesiyle birlikte ilgili sürüme ait yeterlilik testi tamamlanmış olmaktadır. Test bulguları raporlanıp gerekiyorsa regresyon testleri ile aile testine devam edilmektedir. Bu süreçte belirtilen ürün hattı test adımları ürün ailesine eklenen her yeni özellik ile güncellenmektedir. Süreç sonunda doğrulanmış yazılım bireylerinin elde edilmesinin yanı sıra yeni hazırlanmış veya güncellenmiş olan yeniden kullanıma uygun test planları, tanımları, raporları ve yazılımları da yazılım gereksinimleri, referans mimari ve yeniden kullanılabilir bileşenler gibi YÜH yaklaşımı temel varlıkları arasında yerini almaktadır.

6 Yaklaşımın Kısıtları ve Çözüm Önerileri

Bölümümüzde kullanılan gereksinim yönetim aracının ürün hattı yaklaşımına tam olarak uymaması nedeniyle bazı zorluklarla karşılaşılmıştır. Aracı, alandaki özellikleri zorunlu, alternatif ve opsiyonel olarak sınıflandıran YÜH yaklaşımına uygun olarak kullanmak adına özellik modeli elemanlarının ayrı sütunlarda belirtilmesi sağlanmıştır. Buna karşın mevcut alt yapıyla alternatif özelliklerin tanımlanmasında bazı sıkıntılar yaşanmıştır. Bu sıkıntıların zorlama çözümler yerine yaklaşımın doğasına uygun bir şekilde çözülebilmesinin, aracın yaklaşımla uyumlu hale getirilmesiyle mümkün olacağına inanılmaktadır. Literatürde bu soruna çözüm arayan çalışmalarda aracın entegre genişletme dili sayesinde yaklaşıma göre özelleştirilebileceği belirtilmektedir [22, 23]. Ayrıca bu konuda sunulan bazı ticari çözümler de bulunmaktadır [24]. Yakın gelecekte benzer bir çalışma yapılarak gereksinim yönetiminin de yaklaşıma tam bir uyum göstermesinin sağlanması hedeflenmektedir.

Belirtilen uygulama mühendisliği yöntemiyle oluşturulan yazılımın aslında bir aile yazılımı olması ve birey işlevselliğini çalışma zamanında kazanması kod kapsaması açısından bir dezavantaj olarak görülebilir. İlgili yazılım bireyi senaryolarında yer almayan yazılım parçaları da aynı yazılım bünyesinde bulunacağından bireyin testi sırasında kapsanan kod oranı geleneksel yönteme göre daha düşük çıkacaktır. Bu dezavantajın göze alınmasının temel sebebi ilgili AKY ailesi için tek bir konfigürasyon yönetimi yapılarak önemli işçilik kazanımlarının sağlanması arzusudur. Yazılım bireyleri yine üretim hattı yaklaşımıyla fakat her bireye özel ayrı çalışır yazılım üretilecek şekilde geliştirildiğinde, ayrı derlenmelerinden dolayı her yazılım bireyi için ayrı test, dokümantasyon ve konfigürasyon takibi çalışması yapılması gerekecektir. Bu durumun da ürün hattı yaklaşımından sağlanacak faydayı azaltacağı düşünülmüştür. İlgili yazılım ailesi için gerçekleştirilen yeterlilik testlerinde tüm konfigürasyonların dikkate alınmasıyla bu olumsuzluğun ortadan kalkacağına inanılmaktadır. Tüm konfigürasyonların sınandığı yeterlilik testleri sonucunda yüksek kod kapsamasına sahip aile yazılımı elde edilecek ve böylece birey yazılımları için bu anlamda endişe duymaya gerek kalmayacaktır. Öte yandan yaklaşımda tecrübe edilen kısıtlardan bir diğeri ise yazılım ailesi testlerinde tüm konfigürasyonların her zaman dâhil edilememesidir. Özellikle çok dinamik doğası olan AKY ailelerinde farklı bireyler için farklı zamanlarda hızlı olarak sürüm çıkarılması talep edilmekte ve zaman baskısı nedeniyle tüm aile konfigürasyonlarının test sürecinden geçememe durumu ortaya çıkmaktadır. Ayrıca ortak kullanılan bir bileşende yapılan değişiklik bir birey özelinde test edilemediğinde ve bu birey için ileriki safhada başka bir değişiklik nedeniyle ürün çıkarma gereksinimi doğduğunda test yükü artmaktadır. Tüm bu örnekler ürün hattı yeterlilik testlerinin kısa zamanda ve mümkün olan en az iş gücüyle gerçekleştirilmesinin önemini göstermektedir. Bu hedefe giden yolun ise testlerin otomatikleştirilmesinden geçtiği aşikârdır. Bu amaçla yazılım ürün ailelerinin doğruluğunu sınayan yeterlilik testlerinin otomatikleştirilmesi için sektör başkanlığımız bünyesinde bir çalışma başlatılmıştır. Bu çalışma sonlandığında bahsedilen kısıtların giderilmesi hedeflenmektedir.

7 Sonuç ve Değerlendirme

Bu bildiride, alan mühendisliği çalışmaları hali hazırda yapılmış bir gömülü atış kontrol yazılımları ürün hattından aile yazılımlarının ve aileye mensup birey yazılımların oluşturulması süreci; gereksinim yönetimi, yazılım gerçekleme ve test safhaları özelinde sunulmuştur. Belirtilen yaklaşımla sadece pilot olarak seçilen AKY ailesinden 12 adet birey yazılım resmi olarak üretilmiştir. Yaklaşımın avantajlarının görülmesiyle birlikte diğer AKY ailelerinin geliştirilmesinde de uygulamaya alınmıştır. Bildiri çerçevesinde bölümümüzde izlenen yaklaşımın tecrübe edilen bazı eksikliklerinden de bahsedilmiştir. Tespiti yapılan eksikliklere iyileştirme önerileri sunulmuş ve hedefler doğrultusunda yapılması planlanan yakın gelecek çalışmaları aktarılmıştır.

8 Teşekkür

YÜH uygulama mühendisliği yazılım geliştirme çalışmalarına olan katkılarından dolayı Silah Sistemleri ve Modernizasyonları ekibimiz ile test çalışmalarını yürüten ve değerli görüşlerini esirgemeyen yazılım test ekibimize teşekkür ederiz.

Kaynaklar

1. Krueger, C.W.: The emerging practice of software product line development. Military Embedded Systems (2nd semester), pp. 34–36 (2006)
2. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, CMU/SEI-90-TR-21 (1990)
3. Kang, Kyo C., Kim, S., Lee, J., Shin, E., Huh, M.: FORM: A Feature-Oriented Reuse Method with Domain-Specific References Architectures. Annals of Software Engineering, pp. 143–168. ACM (1998)
4. Kang, K.C., Lee, J., Donohoe, P.: Feature-Oriented Product Line Engineering. IEEE Software, Vol.19 No.04, July 2002, pp. 58–65 (2002)
5. Simos, M.A.: Organization Domain Modeling: A Tailorable, Extensible Framework for Domain Engineering. In: Proceedings of the 4th International Conference on Software Reuse (ICSR ’96), pp. 230–232 (1996)
6. Griss, M.L., Favaro, J., d’Alessandro, M.: Integrating Feature Modeling with RSEB. In: Proceedings of Fifth International Conference on Software Reuse, pp. 76–85 (1998)
7. Frakes, W., Prieto-Diaz, R., Fox, C.: DARE: Domain analysis and reuse environment. Annals of Software Engineering, Vol.5 No.1, pp. 125–141 (1998)
8. Sochos, P., Riebisch, M., Philippow, I.: The Feature-Architecture Mapping (FArM) Method for Feature-Oriented Development of Software Product Lines. In: ECBS’06, pp. 308–318. IEEE (2006)
9. Apel, S., Batory, D., Kästner, C., Saake, G.: Feature-Oriented Software Product Lines. Springer, Heidelberg (2013)
10. Fuentes, L., Nebrera, C., Sánchez, P.: Feature-Oriented Model-Driven Software Product Lines: The TENTE approach. In: Proceedings of the Forum of the 21st International Conference on Advanced Information Systems (CAiSE), pp. 67–72 (2009)

11. Kahraman, E., İpek, T., İyidir, B., Bazlamaçcı, C.F., Bilgen, S.: Bileşen Tabanlı Yazılım Ürün Hattı Geliştirmeye Yönelik Alan Mühendisliği Çalışmaları. In: UYMS'09, pp. 283–287 (2009)
12. Karataş, E.K., İyidir, B., Birtürk, A.: Yazılım Ürün Hattı Yaklaşımında Gereksinimlerin Yeniden Kullanımı: ASELSAN Deneyimi. In: UYMS'12, pp. 8–11 (2012)
13. Kim, J., Lee, J., Bae, D.H., Ryu, D., Lee, S.: Developing a Common Operating Environment for Military Application. In: FTDCS'03, pp. 367–373. IEEE (2003)
14. Femmer, H.: Equivalence Analysis for Software Abstraction Layers. Master's Thesis, Augsburg University, Institute for Software and Systems Engineering (2012)
15. Pritchett, W.W., Bartkiewicz, J.D.: A Real-Time Operating Environment for Army Weapon Systems. In: Proceedings of 18th Digital Avionics Systems Conference, pp. 8.A.3-1–8.A.3-7 (1999)
16. Loyall, J., Rohloff, K., Pal, P., Atighetchi, M.: A Survey of Security Concepts for Common Operating Environments. In: 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops, pp. 244–253. IEEE (2011)
17. Kahraman, E., İpek, T.: Gerçek Zamanlı Gömülü Yazılım Geliştirmede Bileşen Entegrasyon Deneyimleri. In: UYMK'08, pp. 206–215 (2008)
18. Pohl, K., Böckle, G., Linden, F.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer, Heidelberg (2005)
19. McGregor, J.D.: Software Product Lines. Journal of Object Technology, Vol.3 No.3, March-April 2004, pp. 65–74 (2004)
20. İyidir, B., Kalay, A.: Gömülü Yazılımlarda Yeniden Kullanım Zorlukları ve ASELSAN Çözümleri. In: UYMS'09, pp. 277–281 (2009)
21. McGregor, J.D.: Testing a Software Product Line. Technical Report, CMU/SEI-2001-TR-022 (2001)
22. Eriksson, M., Börstler, J., Borg, B.: Managing requirements specifications for product lines – An approach and industry case study. The Journal of Systems and Software, Vol.82 No.3, March 2009, pp. 435–447. ACM (2009)
23. Thurimella, A.K., Janzen, J.: metadoc Feature Modeler: A Plug-in for IBM Rational DOORS. In: SPLC'11, pp. 313–322. IEEE (2011)
24. Krueger, C.W., Jackson, K.: Requirements engineering for systems and software product lines. Technical Report, IBM Corporation (2009)