Yazılım Ürün Hatlarında Alana Özgü Bileşenleri Belirleme Yaklaşımı

İbrahim Onuralp YİĞİT[1], Ali Hikmet DOĞRU[2]

[1]ASELSAN A.Ş. SST Sk. Bşk.lığı-Komuta Kontrol Yazılım Tsr. Mdl.
[2]Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü
[1]ioyigit@aselsan.com.tr, [2]dogru@ceng.metu.edu.tr

Özet. Bu bildiride, yazılım ürün hatlarında değişkenlik modelini dikkate alarak nasıl alana özgü bileşenlerin belirleneceğine yönelik bir yaklaşım anlatılmaktadır. Yazılım ürün hatlarında yazılım ürünleri önceden geliştirilmiş alana özgü bileşenlerin bir araya getirilmesi ile oluşturulmaktadır. Geliştirilecek her bir yazılım ürününde alana özgü bileşenleri yeniden kullanabilmek için bu bileşenleri çalışılan alandaki ortaklıkları ve değişkenlikleri göz önüne alarak belirleyip geliştirmek gerekmektedir. Önerilen yaklaşımla alan analizi aşamasında tanımlanan alandaki ortaklıklar ve değişkenlikler doğrudan tasarım aşamasına taşınmaktadır. Tasarım aşamasında yaklaşımda anlatılan adımlar izlenerek alana özgü bileşenler belirlenmektedir. Bu yaklaşımın uygulanması sonucunda alana özgü yeniden kullanıma uygun bir bileşen kümesinin ortaya çıkarılması planlanmaktadır. Bu sayede ortaya çıkan bileşenler yeniden kullanılarak hızlı ve etkin bir şekilde, kaliteli yazılımların geliştirilmesi hedeflenmektedir.

Anahtar Kelimeler: Yazılım Ürün Hattı, Bileşen Tabanlı Yazılım Ürün Hattı, Değişkenlik Yönetimi, Alana Özgü Bileşenler, Yeniden Kullanım, Alana Özgü Yeniden Kullanım

1 Giriş

Günümüzde hızlı ve etkin şekilde, kaliteli yazılım ürünlerinin geliştirilmesi için önceden geliştirilmiş yazılım ürünleri yeniden kullanılmaktadır [1]. Özellikle belirli bir alanda ve birbirine benzer yazılım projelerinin geliştirilmesinde alana özgü yeniden kullanım yaklaşımları benimsenmektedir. Her bir proje için sıfırdan yazılım ürünleri oluşturmak yerine faaliyet gösterilen alan için bir yazılım ürün ailesi geliştirilmektedir. Bu sayede müşterilerden gelen farklı isteklere hızlı bir şekilde cevap verilebilmekte ve faaliyet gösterilen alanda yeni ürünlerin çıkarılması için geçen süre azaltılmaktadır.

Yazılım ürün hattı, alana özgü yeniden kullanım odaklı bir yaklaşımdır [2].Yazılım ürün hatlarında yazılım ürünleri önceden geliştirilmiş alana özgü bileşenlerin bir araya getirilmesi ile oluşturulur [3]. Her bir yazılım ürününde alana özgü bileşenleri yeniden kullanabilmek için bu bileşenleri çalışılan alandaki ortaklıkları ve

değişkenlikleri göz önüne alarak belirleyip geliştirmek gerekir. Böylece hem mevcut yazılım ürünlerinde hem de gelecekte geliştirilecek yazılım ürünlerinde yeniden kullanımın sağlanması için uygun ortam hazırlanmış olur.

Bu bildiride, yazılım ürün hatlarında değişkenlik modelini dikkate alarak nasıl alana özgü bileşenlerin belirleneceğine yönelik bir yaklaşım anlatılmaktadır. Önerilen yaklaşımla alan analizi aşamasında tanımlanan alandaki ortaklıklar ve değişkenlikler doğrudan tasarım aşamasına taşınmaktadır. Tasarım aşamasında yaklaşımda anlatılan adımlar izlenerek alana özgü bileşenler belirlenmektedir. Bu yaklaşımın uygulanması sonucunda alana özgü yeniden kullanıma uygun bir bileşen kümesi ortaya çıkmaktadır. Bu sayede ortaya çıkan bileşenler yeniden kullanılarak daha düşük maliyetli ve kaliteli yazılımların daha hızlı bir şekilde geliştirilmesi hedeflenmektedir.

Bildirinin geri kalanı şu şekilde düzenlenmiştir: ikinci bölümde Yeniden Kullanım, Yazılım Ürün Hattı ve Değişkenlik Yönetimi konuları hakkında bilgiler verilmiştir. Üçüncü bölümde alana özgü bileşenleri belirlemek için önerilen yaklaşım paylaşılmaktadır. Dördüncü bölümde yaklaşımın uygulandığı örnek çalışma ve bu yaklaşımın değerlendirmesi yapılmaktadır. Son bölümde yapılan çalışma sonucunda gelinen noktanın değerlendirmesi yapılmakta ve gelecek dönemde yapılacak çalışmalardan bahsedilmektedir.

2 Yeniden Kullanım, Yazılım Ürün Hattı ve Değişkenlik Yönetimi

2.1 Yeniden Kullanım ve Seviyeleri

Yazılım yeniden kullanımı, yazılımların sıfırdan geliştirmek yerine var olan yazılımları kullanarak oluşturulması işlemidir [4]. Yazılım yeniden kullanımı, sadece kaynak kodların farklı yazılımların geliştirilmesinde kullanılmasıyla sınırlı bir yaklaşım değildir. Yazılım geliştirme süreci boyunca ortaya çıkan tüm varlıklar yeniden kullanılabilir varlıklara dönüştürülebilir [5]. Yazılım mimarisi, gereksinimler, tasarımlar, test tanımları ve dokümanlar yazılım projelerindeki yeniden kullanılabilir varlıklara örnek gösterilebilir [6,7].

Yeniden kullanım için beş tane yetkinlik seviyesi belirlenmiştir [21]. Her yetkinlik seviyesinde bir önceki seviyeden daha fazla yeniden kullanım sağlamaktadır. Bu nedenle yetkinlik seviyelerinde ilerlenmesi, yeniden kullanım seviyesinin, dolayısıyla sağlanan faydanın artmasına karşılık gelmektedir. Yeniden kullanım seviyesi arttıkça ürünlerin daha hızlı şekilde geliştirilmesi mümkün olmakta, geliştirme maliyetleri azalmakta ve ortaya çıkan ürünlerin kalitesi artmaktadır [8].

Yeniden kullanım yetkinlik seviyeleri incelendiğinde en etkin yeniden kullanımın "Alana Özgü Yeniden Kullanım" yaklaşımları ile sağlanacağı görülmektedir [9]. En üst düzey yeniden kullanım olanağı sağlayan bu seviyeye ulaşılabilmesi için çalışılan alanın kapsamını net bir şekilde ortaya koyulmalıdır. Alan analizi ile çalışılan alan içerisindeki ortaklıklar ve değişkenlikler göz önüne alınarak alana özgü mimari ve alan bileşenleri tanımlanmaktadır. Yazılım ürünleri alana özgü mimariye uygun bir şekilde alan bileşenleriyle geliştirilmektedir.

2.2 Yazılım Ürün Hattı

Yazılım ürün hattı yaklaşımı, belirli bir alanda, düşük maliyetli ve kaliteli yazılımları hızlı bir şekilde geliştirmeye yönelik ortaya atılmıştır [10]. Bu yaklaşım, belirli bir ürün ailesinin ortaklıkları ve değişkenlikleri göz önünde bulundurarak temel varlıkları oluşturmayı ve bu varlıkları kullanarak yazılımların geliştirilmesini söylenmektedir [11].

Yazılım ürün hattı, alan ve uygulama mühendisliği olmak üzere birbirine paralel iki ayrı süreçten meydana gelmektedir. Alan mühendisliği sürecinde alandaki ortaklıklar ve değişkenlikler belirlenerek yeniden kullanılabilir ve yapılandırılabilir temel varlıklar oluşturulmaktadır [12]. Uygulama mühendisliği süreci ise temel varlıkları kullanılarak ve yapılandırılarak yazılım ürünlerin geliştirilmesini kapsamaktadır [13].

2.3 Değişkenlik Yönetimi

Yazılım ürün hattı yaklaşımında gerçekleştirilmesi gereken en önemli faaliyetlerden birisi, belli ortaklıkları olan ürünler arasındaki değişkenliklerin yönetilmesidir. Yazılım ürün hattında ürünlerin temel varlıklar kullanarak geliştirilebilmesi için ürünlerin farklılaştıkları noktaları tespit edip ortaya çıkan değişkenliklerin yönetilmesi gerekmektedir. Alan mühendisliği sürecinin alan analizi aşamasında faaliyet gösterilecek alanın kapsamı belirlenmekte, alandaki ortaklıklar ve değişkenlikler ortaya koyulmaktadır. Alan analizi aşamasında problem uzayında bulunan ortaklıkları ve değişkenlikleri modelleyebilmek için değişkenlik modelleri kullanılmaktadır [14].

Değişkenlik modelinde yer alan değişkenlik yönetimi ile ilgili kavramlar şunlardır:

- ***Ortaklıklar***, yazılım ürün hattı kapsamındaki çıkacak ürünlerin paylaştıkları özellikler olarak tanımlanmaktadır.
- ***Değişkenlikler***, yazılım ürünleri arasında farklılık gösteren özellikler olarak nitelendirilmektedir.
- ***Değişkenlik noktası***, yazılım ürünlerinin nerede farklılaştığını belirtmektedir.
- ***Değişken***, bir değişkenlik noktasına karşılık gelen her bir seçenek olarak tanımlanmaktadır.

Değişkenlikler, ürünlerde değişkenlik noktaları ve değişkenlik noktaları altında yer alan değişkenler aracılığıyla gerçekleştirilirler. Aşağıdaki tabloda *değişkenlik tipleri* verilmiştir.

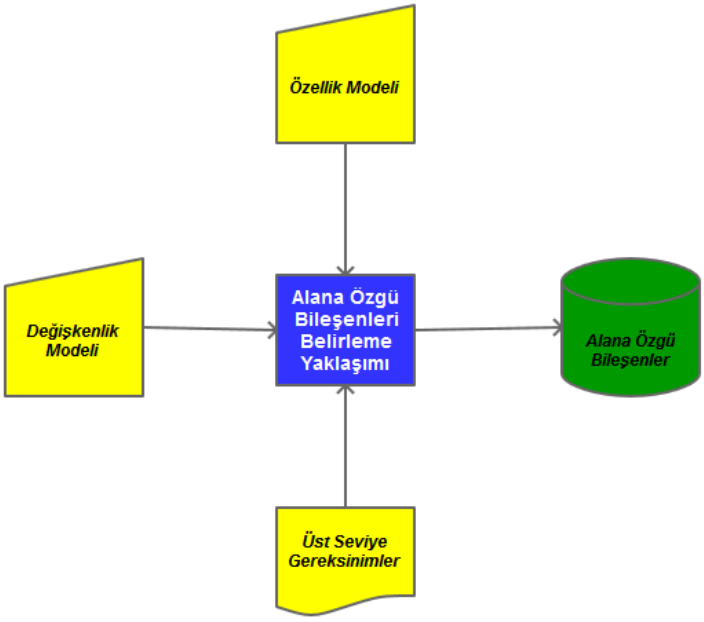
Tablo 1. Değişkenlik Tipleri

Değişkenlik Tipi	**Açıklaması**
Zorunlu	Her yazılım ürününde bulunması gereken özellik
Opsiyonel	Her yazılım ürününde bulunması zorunlu olmayan özellik
Alternatif	Bir grup özellik içerisinden sadece bir tane seçilebildiği özellik

Değişkenliği yönetiminde değişkenlik modelinde yer alan değişkenlikler arasında kısıtlar da tanımlanabilmektedir. Değişkenlik kısıtları 'gerektirir' (requires) ve 'dışlar' (excludes) olmak üzere ikiye ayrılmaktadır. 'gerektirir' değişkenlik kısıdına sahip bir değişkenlik seçildiyse o özelliğin ihtiyaç duyduğu başka değişkenliğin/değişkenliklerin de birlikte seçilmesi gerekir. 'dışlar' değişkenlik kısıdına sahip değişkenlikler beraber seçilemez.

3 Alana Özgü Bileşenleri Belirleme Yaklaşımı

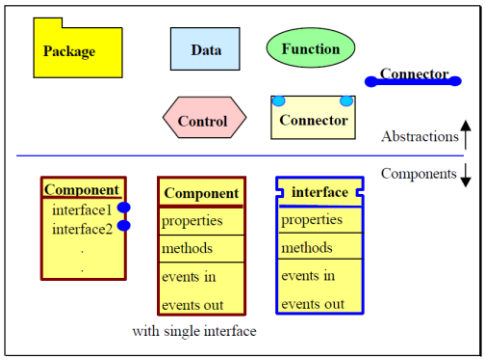
Yazılım ürün hattı yaklaşımında alana özgü bileşenler kullanılarak yazılım ürünleri üretilmektedir. Alana Özgü Bileşenleri Belirleme Yaklaşımı, yazılım ürün hatlarında özellik ve değişkenlik modellerini dikkate alarak nasıl alana özgü bileşenlerin belirleneceğine tarif eden bir yaklaşımdır. Bu yaklaşım kapsamında alan analizi aşamasında belirlenen üst seviye sistem gereksinimlerinden yola çıkarak bileşenler arasındaki ilişkilerin kurulur. Ayrıca, ortaya çıkan bileşen kümesinin tasarım aşamasında büyüklük ve bağımlılık yönlerinden değerlendirilerek bileşen kümesinin gözden geçirilmesini önerir. Bu yaklaşımın uygulanması sonucunda alana özgü, yeniden kullanıma uygun bir bileşen kümesinin ortaya çıkarılması hedeflenmektedir.



Şekil 1. Alana Özgü Bileşenleri Belirleme Yaklaşımı

Alana özgü bileşenleri belirleme yaklaşımı, bileşen yönelimli yazılım mühendisliği [15] yönteminden yola çıkılarak ortaya atılmıştır. Ürün ailesindeki ortaklıklar ve

değişkenlikler özellik modeliyle ifade edilmektedir. Ürün ailesini oluşturan bileşenler COSEML (Component Oriented Software Engineering Modeling Language) [16] modelleme dilini ve değişkenlik modelini temel alan, bu çalışma kapsamında geliştirilen VCOSEML (Variablity & Component Oriented Software Engineering Modeling Language) modelleme dili kullanılarak belirlenmektedir. Ayrıca, yine bu çalışma kapsamında VCOSEML dili ile modelleme yapabilmek için VCOSECASE aracı geliştirilmiştir. VCOSEML'de COSEML'deki paket, fonksiyon ve veri elemanları kullanılarak soyutlama gerçekleştirilmektedir. COSEML soyutlama elemanları Şekil 4'de gösterilmiştir.



Şekil 2. COSEML Soyutlama ve Bileşen Elemanları [16]

Bu yaklaşım, alana özgü bileşenler belirlenirken aşağıdaki adımların sırasıyla izlenmesini önermektedir. Bu bölümün bundan sonraki kısımlarında önerilen adımlara ilişkin detaylı açıklamalar verilecektir.

1. Özellik modelini VCOSEML soyutlama elemanları ile modellenmesi
2. VCOSEML soyutlama modelinin üzerine değişkenlik modelinin yerleştirilmesi
3. VCOSEML soyutlama modeline göre bileşenlerin belirlenmesi
4. Bileşenleri büyüklük ve bağımlılık yönünden değerlendirilesi

3.1 Soyutlama Modelinin Oluşturulması

Özellik modellemeye dayalı alan analizinden alana özgü bileşenlere geçiş süreci soyutlama modelinin oluşturulmasıyla başlar. VCOSEML modelleme dili kullanılarak soyutlama modeli grafiksel olarak oluşturulur. Özellik modelinde yer alan özellikler doğrudan soyutlama modeline aktarılarak VCOSEML soyutlama elemanları ile ifade edilir. Alan mühendisleri özellik modelinde yer alan özellikleri VCOSEML'deki paket, fonksiyon ve veri elemanlarını kullanarak soyutlamayı gerçekleştirirler. Paketler soyutlamanın ana parçalarını göstermektedir. Paket detayları fonksiyon ve veri elemanları ile ifade edilir.

3.2 Soyutlama Modeli Üzerine Değişkenliklerin Yerleştirilmesi

Alan içindeki değişkenliklerin tasarım aşamasında izlenebilirliğinin sağlanması alana özgü bileşenlerin değişkenlikleri de göz önüne alarak belirlenmesini kolaylaştıracağı düşünülmüştür. Bu nedenle soyutlama modelinden bileşenlere geçişte değişkenliğin modellenmesi ve izlenebilirliğinin sağlanması için soyutlama modeline değişkenliklerin gösterilmesini sağlayan grafiksel ifadeler eklenmiştir. VCOSEML modelleme dilinde yer alan grafiksel değişkenlik ifadeleri aşağıdaki tabloda gösterilmektedir. VCOSEML'deki değişkenlik ile ilgili grafiksel gösterimler için değişkenlik yönetiminde en çok kullanılan modellerden birisi olan OVM (Orthogonal Variability Model) [14] referans alınmıştır.

Tablo 2. VCOSEML Değişkenlik Gösterimleri

Değişkenlik	Gösterim
Değişkenlik noktası	
Değişken	
Zorunlu	
Opsiyonel	
Alternatif	

Alan analizi aşamasında hazırlanan değişkenlik modeli VCOSEML'in sunduğu grafiksel değişkenlik gösterim yetenekleri sayesinde soyutlama modeli üzerine taşınır. Soyutlama modeline değişkenlik noktaları, değişkenlikler, değişkenlik tipleri ve değişkenler arasındaki kısıtlar eklenir. Böylece alandaki değişkenlikler tasarım aşamasına taşınmış olur.

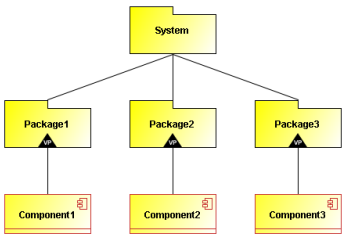
3.3 Bileşenlerin Belirlenmesi

Bileşenler, birlikte çalışabilen ve diğer bileşenlerle ile arayüzleri belirlenmiş yazılım yapıtaşlarıdır [17]. Yazılım ürün hattını oluşturacak yapıtaşlarının soyutlama elemanları ile ilişkisi, bileşenleri belirleme aşamasında kurulur. Çalışılan alandaki yapıtaşlarını ortaya koymak için sistem yukarıdan aşağıya doğru ayrıştırılır. Sistemi ayrıştırmanın amacı, sistemi alana özgü yeniden kullanılabilir bileşenlere denk düşecek şekilde daha küçük alt parçalara bölmektir. Bu parçaların yazılım ürünlerinde yeniden kullanım oranını arttırması beklenmektedir.

Yazılım bileşenleri çözüm uzayında, soyutlama aşamasında tanımlanan problem uzayı öğelerini gerçekleyen yazılım parçaları olarak değerlendirilebilir. Soyutlama modelinden bileşenlere geçişte bu başlık altında verilen kurallar uygulanarak alana özgü bileşenler belirlenmektedir. Bu yaklaşım kapsamında alan mühendislerinin alana özgü bileşenleri belirlerken bu kurallara uygun bir şekilde tasarım yapması tavsiye edilmektedir.

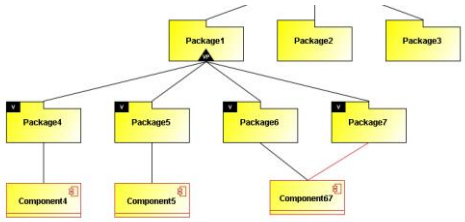
Kural1: Her değişkenlik noktasına karşılık en az bir bileşen gelecek şekilde gerçekle. Sistem değişkenlik noktalarına karşılık en az birer bileşen gelecek şekilde

tasarlanmalıdır. Aşağıdaki şekilde bu durumun bir örneği gösterilmiştir. Değişkenlik noktalarını barındıran paketler ayrı ayrı bileşenlerle gerçeklenmiştir.



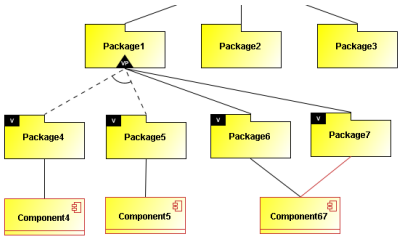
Şekil 3. Kural1 için örnek durum

Kural2: Aynı değişkenlik noktasında birbirini dışlayan değişkenleri ayrı ayrı bileşenlerle gerçekle. Yazılım ürün hatlarındaki yaygın sorunlardan birisi, değişkenliklerin birbiriyle çelişmesidir. 'dışlar' ilişkisi aynı bileşen kapsamında gerçeklenen değişkenlik elemanları arasında bulunmaması gerekir. Bir takım özelliklerin aynı anda var olması sistemin doğru bir şekilde çalışmamasına veya çalışmasını engelleyecek etkenlerin ortaya çıkmasına neden olabilir. Bu nedenle aynı değişkenlik noktası altında birbirini dışlayan değişkenlerin ayrı bileşenlerle gerçeklenmesi önerilmektedir. Aşağıdaki şekilde bu durum gösterilmiştir. Package4 ve Package5 paketleri aynı değişkenlik noktasında bulunmasına rağmen birbirini dışlayan değişkenliklere sahip oldukları için ayrı bileşenlerle gerçeklenmişlerdir. Birbirini dışlamayan Package6 ve Package7 paketlerinde bulunan değişkenlikler ise aynı bileşen kapsamında ele alınmıştır.



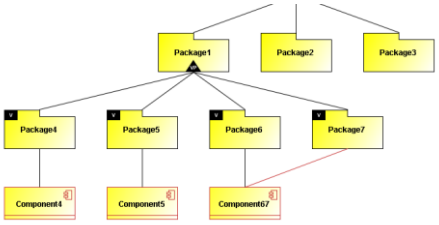
Şekil 4. Kural2 için örnek durum

Kural3: Aynı değişkenlik noktasında birbirinin alternatifi olan değişkenliklerin her birini ayrı ayrı bileşenlerle gerçekle. Kural2'de olduğu gibi birbirinin alternatifi olan değişkenler birbirini dışlamaktadır. Bundan dolayı ayrı bileşenlerde gerçeklenmelerinin daha doğru olacağı önerilmektedir. Şekil 5'de bu durum resmedilmiştir. Package4 ve Package5 paketleri birbirine alternatif değişkenler barındırdıkları için ayrı bileşenlerle gerçeklenmişlerdir.



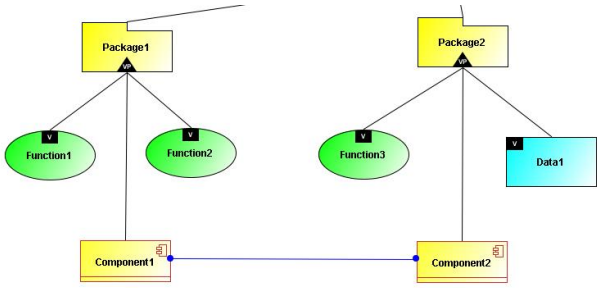
Şekil 5. Kural3 için örnek durum

Kural4: Aynı değişkenlik noktasında birbirini gerektiren değişkenlikleri aynı bileşen içerisinde gerçekle. Aynı değişkenlik noktasından birbirine ihtiyaç duyan değişkenlerinin aynı bileşen tarafından gerçeklenmesi iyi uyumun sağlanması ve başka bileşenlere bağımlılığı azaltmak için tavsiye edilmektedir. Şekil 6'da verilen örnek durumda Package6 ve Package7 paketlerindeki değişkenler birbirlerini gerektirdiği için aynı bileşen çerçevesinde gerçeklenmeleri tercih edilmiştir.



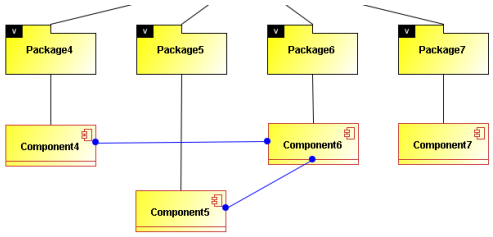
Şekil 6. Kural4 için örnek durum

Kural5: Farklı değişkenlik noktalarında bulunan ve birbirini gerektiren değişkenliklerin gerçekleyen bileşenleri birbirlerine bağlaç (connector) ile bağla. Kural1'de her bir değişkenlik noktasının farklı bileşenlerle gerçeklenmesi tavsiye edilmiştir. Bu değişkenlik noktalara arasında birbirini gerektiren değişkenlikler bulunması durumunda bu değişkenlikleri gerçekleyen bileşenler arasında iletişimin sağlanması için bağlaçlar aracılığıyla bağlanması gerekir. Şekil 7'de bu durumun bir örneğine yer verilmiştir. Farklı değişkenlik noktaları altında bulunan Function2 fonksiyonu Data1 veri öğeleri altında birbirini gerektiren değişkenler bulunduğundan dolayı bu değişkenleri barındıran Component1 ve Component2 bileşenleri bağlaç aracılığıyla birbirine bağlanmıştır.



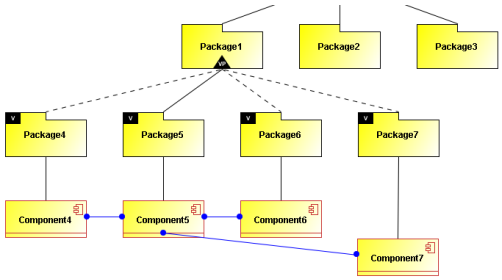
Şekil 7. Kural5 için örnek durum

Kural6: Aynı değişkenlik noktasında birbirini dışlayan değişkenliklerin gerek duyduğu değişkeni ayrı bileşen yap ve Kural5'i uygula. Kural2'de aynı değişkenlik noktası altında birbirini dışlayan değişkenler farklı bileşenler ile gerçeklenmesi önerilmiştir. Kural6 kapsamında ise birbirini dışlayan değişkenleri içeren bu bileşenlerin gerektirdiği değişkenliğin de ayrı bileşen yapılması ve diğer bileşenlere bağlaç aracılığıyla bağlanması tavsiye edilmektedir. Böylece birbirini dışlayan bileşenlerin ortak bağımlı olduğu kısımları daha küçülterek alandaki birbirinden bağımsız ortaklıkların artırılması amaçlanmaktadır. Şekil 8'de verilen örnekte Package4 ve Package5 paketlerindeki değişkenler birbirini dışlarken aynı değişkenler Package6 paketinde verilen değişkeni gerektirmektedir. Bu nedenle bu üç paket ayrı bileşenlerle gerçeklenmiştir. Component6 bileşeni kapsamında gerçeklenen değişken Component4 ve Component5 bileşenlerinde bulunan değişkenler tarafından ihtiyaç duyulduğu için bağlaçlarla bağlantı kurulmuştur.



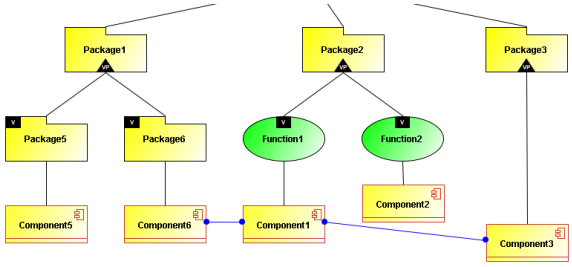
Şekil 8. Kural6 için örnek durum

Kural7: Opsiyonel olan değişkenlikleri ayrı bileşen yap ve zorunlu değişkenlikleri barındıran bileşene bağlaç ile bağla. Opsiyonel olarak tanımlanan soyutlama öğelerinin ayrı bileşenler olarak gerçeklenmesi önerilmektedir. Opsiyonel değişkenlerin seçim durumlarında zorunlu değişkenlerle bağlantılarının sağlanması için değişkenleri gerçekleyen bileşenlerin bağlaçlar aracılığıyla bağlanması gerekir. Şekil 9'daki örnek durumda zorunlu değişkeni gerçekleyen bileşenle opsiyonel değişkenleri gerçekleyen bileşenlerin nasıl birbirine bağlandıkları görülmektedir.



Şekil 9. Kural7 için örnek durum

Kural8: Farklı değişkenlik noktalarında bulunan en az iki değişkenlik öğesi bu öğelerden farklı bir değişkenlik noktasında bulunan değişkenliği gerektiriyorsa bu değişkenliği ayrı bileşen olarak gerçekle ve Kural5'i uygula. Farklı değişkenlik noktalarında bulunan değişkenler farklı bileşenler tarafından gerçeklenmekte ve bileşenler arasında birbirini gerektiren değişkenlerin bulunması durumunda aradaki bağlantı bağlaçlar aracılığıyla sağlanmaktadır. Farkı değişkenlik noktasında bulunan bir değişkene en az iki değişkenlik noktasında bulunan değişkenlerden 'gerektir' kısıdı bulunması durumunda bileşenler arasındaki bağımlılığı azaltmak için ihtiyaç duyulan değişkenin ayrı bir bileşen tarafından gerçeklenmesi tavsiye edilmektedir. Şekil 10'da verilen örnek durumlarda Package6 ve Package3 paketleri farkı değişkenlik noktalarında bulunduğu için ayrı bileşenlerle gerçeklenmiştir. Bu değişkenlikler Function1 fonksiyon soyutlama öğesinde bulunan değişkene ihtiyaç duyduğundan dolayı Function1 ayrı bileşen tarafından gerçeklenmiş ve diğer bileşenlerle bağlaç aracılığıyla bağlanması sağlanmıştır.



Şekil 10. Kural8 için örnek durum

3.4 Bileşen Kümesinin Değerlendirilmesi

Yaklaşım kapsamında belirlenen alana özgü bileşenlerin büyüklük ve bağımlılık açısından değerlendirilmesi önerilmektedir. Belirlenen bileşenlerin fonksiyonel büyüklüklerinin ölçmek için COSMIC [18] yöntemi kullanılmaktadır. Bileşenlerin fonksiyonel büyüklüklerini ölçmek için yapılan çalışmalar, COSMIC yönetiminin uygunluğunu işaret etmektedir [19].

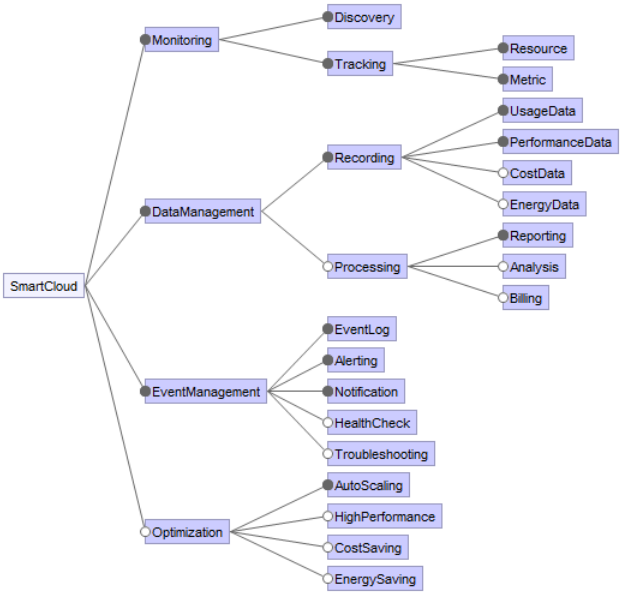
Ortaya çıkan bileşenlerin mümkün olduğunca aralarındaki bağımlılığın az olması tercih edilmektedir. Bileşenlerin aralarındaki bağımlılığı ölçmek için bileşen arayüz karmaşıklığı ve ortalama arayüz karmaşıklığı değerleri hesaplanmaktadır [20].

Yapılan ölçümler değerlendirilerek büyüklüğü ve diğer bileşenlere bağımlılığı fazla olan bileşenler tespit edilir. Bu ölçümler sonucunda bileşen kümesinin belirlenmesi için yapılan tasarım gözden geçirilir. Gözden geçirme sonrasında bileşenleri belirleme aşamasında verilen kurallar yeniden uygulanarak bileşen kümesi güncellenebilir.

4 Örnek Çalışma

Bu bölümde Alana Özgü Bileşenleri Belirleme Yaklaşımı ile ilgili yapılan örnek çalışma paylaşılacaktır. Bulut bilişim alanında bulut kaynaklarının takibi ve optimizasyonu ile ilgili yazılım ürünlerinin çıkarılacağı bir ürün hattı için alan mühendisliği çalışmaları yürütülmüştür. Alan mühendisliğinin alan tasarımı aşamasında önerilen yaklaşımda tarif edilen adımlar izlenerek alana özgü bileşenler belirlenmiştir.

Yaklaşım uygulanmadan önce örnek çalışma alanı için alan analizi çalışmaları yürütülmüştür. Alan analizi çalışmaları kapsamında üst seviye sistem gereksinimleri, özellik ve değişkenlik modelleri hazırlanmıştır. Şekil 11'de bulut kaynaklarının takibi ve optimizasyonu ile ilgili hazırlanan özellik modeli gösterilmiştir.



Şekil 11. Bulut Takibi ve Optimizasyonu Özellik Modeli

Yaklaşımın ilk aşamasında özellik modeli doğrudan VCOSEML soyutlama elemanlarıyla modellenerek tasarım aşamasına taşınmıştır. Soyutlama modeli oluşturulduktan sonra değişkenlik modelinde yer alan değişkenlikler ve değişkenlik kısıtları soyutlama modeline ifade edilmiştir. Bu işlemin ardından bileşenleri belirleme adımına geçilmiştir. Bileşenleri belirleme aşamasında yaklaşımda önerilen kurallar uygulanarak alana özgü bileşenler belirlenmiştir. Son olarak belirlenen bileşenler büyüklük ve bağımlılık yönlerinden değerlendirilip ortaya çıkan bileşen kümesine ilişkin yapılan tasarım gözden geçirilmiştir. Aşağıdaki tabloda yaklaşımın uygulanmasıyla ortaya çıkan bileşen kümesi, bu kümede yer alan bileşenlerin fonksiyonel büyüklükleri ve arayüz bağımlılık değerleri yer almaktadır.

Tablo 3. Ölçüm Sonuçları

Bileşen	**Fonksiyonel Büyüklük**	**Arayüz Bağımlılık Sayısı**
ResourceDiscovery	26	3
IaasManager	40	5
PaasManager	29	3
SaasManager	25	3
SystemTracker	35	2
ApplicationTracker	38	3
CostTracker	32	2
EnergyTracker	32	2
CustomTracker	41	5
UsageDataManager	42	5
PerformanceDataManager	47	4
CostDataManager	38	5
EnergyDataManager	38	3
Reporter	50	2
Analyzer	87	8
BillManager	36	1
EventManager	88	7
HealthChecker	40	2
TroubleShooter	42	1
UsageOptimizer	50	3
PerformanceOptimizer	36	3
CostOptimizer	20	3
EnergyOptimizer	20	3

Yukarıdaki tablo incelendiğinde yaklaşımın uygulanması sonucu belirlenen Analyzer ve EventManager bileşenlerinin ortalamanın çok üzerinde büyüklüğe ve bağımlılığa sahip olduğu tespit edilmiştir. Bunun üzerine alan analizi aşamasına hazırlanan özellik modeli tekrar gözden geçirilmiş ve bu bileşenlerin gerçekleştiği özellikler detaylandırılmıştır. Yenilenen özellik modelinde yer alan özelliklere ilişkin değişkenlik modeli de güncellenmiştir. Yapılan değişiklikler doğrultusunda yaklaşım

tekrar uygulanmıştır. Yeniden yaklaşımın uygulanması ile Analyzer bileşeni dört alt bileşene (PerformanceAnalyzer, CostAnalyzer, TrendAnalyzer, WhatIfAnalyzer), EventManager bileşeni ise üç alt bileşene (EventLogger, AlertManager, NotificationManager) bölünmüştür. Böylece yeniden kullanıma daha uygun, daha küçük ve daha az bağımlı bileşenlere ulaşılmıştır.

5 Sonuç

Önerilen yaklaşım, özellikle bileşen tabanlı yazılım ürün hatları için bir yol gösterici niteliği taşımaktadır. Bu yaklaşımla beraber değişkenlik modeli tasarım aşamasına taşınarak değişkenliklerin izlenebilirliği artırılmıştır. Yaklaşımın uygulanması sonucunda ortaya çıkan bileşenler değişkenlikler dikkate alınarak belirlendiği için yeniden kullanım olanaklarının iyileşmesi ve yazılım ürün hattından alınacak verimin artması beklenmektedir.

Önümüzdeki dönemde özellikle önerilen yaklaşımın ölçüm ve değerlendirme aşamasının geliştirilmesi amaçlanmaktadır. Ölçüm ve değerlendirme aşamasında belirlenen bileşenlerin fonksiyonel büyüklük ve bağımlılık metriklerinin yanı sıra bağlılık açısından da her bir bileşenin kendi içinde tutarlılığının değerlendirilmesi planlanmaktadır. Ayrıca örnek çalışmada yaklaşımın uygulanması ile belirlenen bileşen kümesini gerçekleştirilecektir. Gerçekleştirilen bileşen kümesi ile tasarım aşamasındaki belirlenen bileşen kümesi karşılaştırılacaktır. Örnek çalışmanın hayata geçirilmesi ile önerilen yaklaşım doğrulanıp endüstride kullanılabilir bir metodoloji olarak sunulması hedeflenmektedir.

Kaynaklar

1. W. Frakes ve K. Kang, "Software Reuse Research: Status and Future," *IEEE Transactıons on Software Engıneerıng*, Cilt 31, No. 7, pp. 529 - 536, 2005.
2. E. Barak, S. Erdem ve H. Yilmaz, "TADES:Komuta Kontrol Alanında Bir Yazılım Ürün Hattı Çalışması," *Ulusal Yazılım Mimarisi Konferansı*, Ankara, Türkiye, 2010
3. P. Clements ve L. Northrop, Software Product Lines Practices and Patterns, Addison Wesley, 2002 ISBN: 0201703327.
4. C. Krueger, Software Reuse, Pittsburgh, Pennsylvania: ACM Computing Surveys, 1992.
5. R. Prieto-Diaz, "Software Reusability," *IEEE Software*, pp. 61-66, 1993.
6. B. Barnes ve T. Bollinger, "Making Software Reuse Cost Effective," *IEEE Software*, pp. 13-24, 1991.
7. F. Belli, "Dependability and Software Reuse – Coupling Them by an Industrial Standard," *Software Security and Reliability-Companion (SERE-C), 2013 IEEE 7th International Conference on*, pp. 145-154,June 2013.
8. M. Griss, "Systematic Software Reuse: Architecture, Process and Organization are Crucial," *Fusion Newsletter*, October 1996.
9. B. Durak, E. K. Akbiyik ve I. O. Yigit, "Deniz Savunma Sistemleri Alanında Sistematik Yazılım Yeniden Kullanım Yaklaşımı," *Ulusal Yazılım Mühendisliği Sempozyumu*, Güzelyurt, KKTC, 2014.

10. E. Kahraman, T. Ipek, B. Iyidir, C. Bazlamaci ve S. Bilgen, "Bileşen Tabanlı Yazılım Ürün Hattı Geliştirmeye Yönelik Alan Mühendisliği Çalışmaları," *Ulusal Yazılım Mühendisliği Sempozyumu*, İstanbul, Türkiye, 2009.
11. B. C. Kasikci ve S. Bilgen, "Etkin Yeniden Kullanım: Yazılım Ürün Hatlarında Değişkenliğin Modellenmesi," *Ulusal Yazılım Mühendisliği Sempozyumu*, İstanbul, Türkiye, 2009.
12. R. Atas ve O. Kalipsiz, "Servis Tabanlı Yazılım Ürün Hattı Mimarileri," *Fırat Üniversitesi Elektrik-Elektronik ve Bilgisayar Sempozyumu*, Elazığ, Türkiye, 2011.
13. E. K. Karatas ve B. Iyidir, "Yazılım Ürün Hattı Yaklasımında Model Güdümlü Uygulama Mühendisliği," *Ulusal Yazılım Mühendisliği Sempozyumu*, İstanbul, Türkiye, 2009.
14. K. Pohl, G. Böckle, ve F. v. d. Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, Springer, Berlin Heidelberg New York, 2005.
15. M. M. Tanik ve A. Ertas, "Interdisciplinary Design and Process Science: A Discourse on Scientific Method for the Integration Age," *Journal of integrated Design and Process Science*, September 1997, s. 76-94.
16. A. H. Dogru, "Component-Oriented Software Engineering Modeling Language: COSEML", Aralık 1999, Türkiye.
17. A. H. Dogru ve M. M. Tanik, "A process model for component-oriented software engineering," *Software, IEEE*, vol.20, no.2, pp.34,41, Mart/Nisan 2003.
18. COSMIC – Common Software Measurement International Consortium: The COSMIC Functional Size Measurement Method - version 3.0.1 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003), May 2009.
19. O. Eren, B. Ozkan ve O. Demirors, "Yazılım Ürün Hatları için Otomatik İşlevsel Büyüklük Yaklaşımı," *Ulusal Yazılım Mühendisliği Sempozyumu*, Güzelyurt, KKTC, 2014.
20. S. Kumar, P. Tomar, R. Nagar, ve S. Yadav, "Coupling Metric to Measure the Complexity of Component Based Software through Interfaces," *International Journal of Advanced Research in Computer Science and Software Engineering*, *4*(4), 157–162, 2014.
21. M. Griss, "Reuse Strategies CMM as a Framework for Adopting Systematic Reuse," *Object Magazine*, pp. 60-62, 69, 1998.