Bileşen Modellerinde Değişkenlik Yönetimi Yaklaşımlarının İncelenmesi

Muhammed Çağrı Kaya[1], Alper Karamanlıoğlu[1], Mahdi Saeedi Nikoo[1,2], Sina Entekhabi[1], Selma Süloğlu[3], Ali H. Doğru[1]

[1]Orta Doğu Teknik Üniversitesi,
Ankara, Türkiye
{mckaya,alperk,entekhabi,dogru}@ceng.metu.edu.tr
[2]Spark Kalibrasyon Hizmetleri,
Ankara, Türkiye
mahdi_saeedi@sparkkalibrasyon.com.tr
[3]Sosoft Bilişim Teknolojileri,
Ankara, Türkiye
selma@sosoft.com.tr

Özet Bu çalışmada daha önce bileşen modelleri üzerine yapılmış bir yazın taraması çalışmasının sonucunda ortaya çıkan modellerde değişkenlik yönetimi kabiliyeti araştırılmaktadır. Yazarlar tarafından bu doğrultuda kapsamlı bir 'Sistematik Yazın Taraması' (SLR) çalışması yapılmaktadır, bu çalışmanın ara bulguları olarak da faydalı bilgiler ortaya çıkmıştır ve bu bilgiler ön değerlendirmeler olarak bu makalede sunulmaktadır. 24 bileşen modeli üzerinde çalışma yapılmış ve genelde bu modellerde değişkenlik yönetiminin yeterince desteklenmediği gözlemlenmiştir. Makale, bileşen modeli değerlendirmelerini sunduktan sonra değişkenlik konusunda bazı önerileri de dile getirmektedir.

Anahtar Kelimeler: bileşen modeli, değişkenlik modeli, değişkenlik yönetimi, yazılım bileşeni

1 Giriş

Günümüzde tekrar kullanılabilir bileşenleri biraraya getirerek oluşturulan sistemlerin sayısı gittikçe artmaktadır. Yazılım bileşenleri daha kompleks sistemleri oluşturmak üzere bir araya getirilen yapı taşlarıdır. Bu karmaşık ve büyük çaplı sistemleri gerçekleştirmek için çok sayıda bileşen modeli önerilmiştir. Ancak bu yaklaşımların başarılı sayılmaları için sistematik bir tekrar kullanımı desteklemeleri, kabul edilebilir bir sürede tamamlamayı sağlaması ve kalite isterlerini karşılamaları beklenmektedir. Bileşen tabanlı bir sistemin geliştirme sürecinde tekrar kullanımı desteklemek için hem ortak parçaların hem de değişken parçaların doğru ifade edilmesi gerekir [1].

Diğer taraftan, yazılım ürün hatlarında yaygınca faydalanılan ortaklık ve değişkenlik (commonality and variability) kavramlarının modellemesinin ne derece

etkin olduğu bilinmektedir. Değişkenliğin etkin bir şekilde yönetilmesi sonucunda yeniden kullanım yönelimli verimliliğin en yüksek düzeylerde ve kurumsal bir tabanda planlandırılmış olarak elde edilmesi hedeflenebilir.

Değişkenlik, yazılım ürün hatlarında olduğu gibi, verimli bileşimsel sistemler elde etmek için kullanılması gereken önemli bir kavramdır. Ortaklık ve değişkenlik yazılım ürün hatlarının kavramsal temelini oluşturmaktadır. Alan modeli belirli bir olgunluğa ulaştığında, ürün mühendisliği için ortaklık kavramını kullanmak daha az çaba gerektirir. Ancak ürünün belirtilmesi ve inşasına doğru değişkenlik çok önemli bir yere sahip olur. Ürün mühendisliğinin verimliliğini etkileyen en önemli faktörlerden biri bağlanma zamanlarını göz önüne alarak değişkenliği yönetebilmektir [2].

Bileşenler, sağladıkları tekrar kullanım teknolojisiyle yazılım geliştirmenin önemli bir unsuru haline gelmişlerdir. Bu çalışma temel olarak bileşen tabanlı yazılım geliştirmede değişkenliğin kullanımını araştırmaktadır. Değişkenlik kullanımını yaygınlaştırmak ve ilerletmek için öncelikle var olan yaklaşımların incelenmesi gerekir. Bu nedenle bu çalışmada öncelikle çeşitli bileşen modelleri için değişkenliğin ele alınma durumu araştırılmıştır. Uzun vadede hedef, bileşen tabanlı yaklaşımlarda etkin değişkenlik yönetimi sağlayacak kabiliyetlerin geliştirilmesidir.

Crnkovic vd. tarafından yapılan literatür çalışmasında [3] bir bileşen modelinin tanımlaması yapılmış ve var olan yaklaşımlardan bu tanımlamaya uyan 24 model olduğu belirtilmiştir. Ele alınan modeller ve sınıflandırma kriterlerine göre modellerin karşılaştırma tablolarına [3]'ten ulaşılabilir. Bu çalışmada [3]'teki bileşen modelleri referans alınmış ve hangilerinin değişkenliği desteklediği araştırılmıştır. Değişkenlik desteği olan modellerin hangi yaklaşımlarla modelleme yaptıkları incelenmiştir.

Referans çalışma [3], incelediğimiz kadarıyla bileşen modellerini sınıflayan en güncel çalışmadır. Çalışmamızda, referans çalışmanın yayınlandığı tarihten sonra önerilen bileşen modellerinin, yine belirtilen sınıflandırma kriterlerine uygunluğu başka bir çalışma konusu olarak değerlendirilmiş ve mevcut bileşen modellerinin değişkenlik destekleri araştırılmıştır.

2 Değişkenlik

Yazılım ürün hatlarının kullanımı için önerilen Nitelik Modeli (Feature Model) kavramı, Kang [4] tarafından ortaya atılmış ve kısa zamanda bu modellerin daha kapsamlı versiyonları geliştirilmiştir. Farklı endüstri alanlarında yaygınca uygulama alanı bulan yazılım ürün hatları ve dolayısı ile Nitelik Modelleri, değişkenlik modellemesi konusunda bir öncü olarak ortaya çıkmıştır. Nitelik modelleri içerisinde örgün olarak modellenen değişkenlik, ciddi projelerde çok kısa zamanda hem nitelik modelinin fazlaca büyümesi ve karmaşıklaşması hem de değişkenliğin bu karmaşıklık içerisinde ve ayrıca kendi karmaşıklığı ile ele alınmasının zorluklarını ortaya koymuştur. Bu gözlemin sonucu olarak değişkenliği ayrıca modelleyen yaklaşımlar belirmiştir.

Değişkenlik modellemesi konusunda öncü kabul edilebilecek iki yaklaşım olarak OVM [5] ve Covamof'tan [6] söz edilebilir. Nitelik modelleri de örgün olarak kapsamlı bir değişkenlik modeli içermektedir. Daha sonra birkaç hatırı sayılır değişkenlik modeli de ortaya çıkmıştır, ancak halen en çok bu iki model kullanılmaktadır. Değişkenlik modellerinde aranan nitelikler olarak:

1. Kendi başına ayrı bir model olması,
2. Bağlanma zamanının (örneğin tasarım, derleme, yükleme gibi zamanlarında değişkenliğin çözümlenmesi) yönetilebilmesi,
3. Dış ve iç değişkenliğin modellenebilmesi,
4. Değişkenlikler arasında kısıtların tanımlanabilmesi ve
5. Hiyerarşik değişkenlik modellemesi ve değişkenlik çözümlenmesinin (mümkünse otomatik) yayılımı

ortaya çıkmaktadır.

Değişkenliği modellerken tipik olarak değişkenlik noktaları (variation points) ve bu noktalarda sunulacak seçenekler (variants) ele alınmaktadır. Ayrıca, nitelik modellerinde bulunan kısıt bağlantıları da değişkenlikleri içererek daha genişletilmiş bir çerçevede ele alınmalıdır. Bir değişkenlik noktasında yapılacak bir tercihin, sistemin farklı noktalarında hangi seçeneklerin içerilme veya yasaklanmasına neden olacağı model tarafından desteklenmelidir.

3 Bileşen Tabanlı Yazılım Geliştirme

Bileşen tabanlı yaklaşımlar, yeniden kullanılabilir yapı taşları, yani bileşenlerin kullanılması ile karmaşık ve büyük ölçekli yazılım sistemlerinin gerçekleşmeleri için kullanılmaktadır. Bileşen Tabanlı Yazılım Mühendisliği (CBSE), bağımsız bileşenlerin kullanılması ile gevşek bağlı sistemleri tanımlama, modelleme ve uygulama konularında süreçler ve metodolojiler sunmaktadır. 1990'lardan itibaren, Bileşen Tabanlı Yazılım Geliştirme (CBSD) yaklaşımının temel fikri, önceden oluşturulmuş yazılım bileşenlerini entegre ederek büyük sistemlerin inşa edilmesi olmuştur. Bileşenler, uygulama ayrıntılarını gizleyebilirken işlevlerini ve davranışlarını gösterir arayüzler sunmaktadırlar. Belirtilen arayüzlerin kullanılması ile bileşenler bile üçüncü parti bileşenlerin tümleştirilmesi yoluyla daha kolayca kurumlandırılabilir. Bileşen teknolojileri farklı platformlarda olgunlaşırken ilk göze çarpan sunumları Szypersky [7] tarafından yayınlanan bir kitapla 1995 yılında yazılım geliştiricilerin dikkatine sunulmuştur. Bileşenlerin yalnızca bir teknoloji değil, temel bir yönelim kavramı olarak da ele alınması önerilmiştir [8]. Nesne yöneliminde nasıl gereksinimler düzeyinden başlanarak her kavram nesne sınıfları ile modelleniyorsa, bileşene yönelik bir yaklaşımda da soyut ve somut düzeyleri de kapsamak üzere her kavramı bileşenlerle modelleme fikri öne atılmıştır. Bileşen teknolojileri, kod yazmadan geliştirme yapma paradigmasını destekleyen teknolojiler olarak ele alınmıştır. Bu yaklaşım, problem tanımının elde edilmesinin hemen ardından problemin alt parçalarının, 'soyut bileşenler' modellenerek bir çözüm temsili ile eşlenmeleri ve daha sonra bu soyut bileşenlerin mevcut bileşenlerle gerçeklenmelerini önermektedir. Soyut tanımlar ile mevcut bileşenler

arasındaki uyumsuzluklar ise modeldeki ayrıştırma ve son çare olarak da bileşen uyarlama/geliştirme etkinlikleri ile çözümlenmelidir. Yazılım sistemleri için talep geliştiği sürece büyük boyutta ve daha karmaşık gereksinimlerin yeniden kullanılabilirlik ve bileşen yönetimi ile karşılanması gittikçe kaçınılmaz hale gelmektedir.

4 Mevcut Bileşen Modellerine Bakış

Bileşen modelleri, tümleşik birleştirme ortamlarınca desteklenmek üzere farklı araç üreticileri tarafından ve araştırmacılar tarafından önerilmiştir. Önceleri iki üç değişik modelden söz edilmiş ve hala bu modellerin etkinliği devam etmekte ise de zamanla fazla sayıda bileşen modeli ortaya çıkmıştır ve bu sayı hala artmaktadır. Diğer taraftan, nitelik modellerinin kullanılmaya başlamasının ardından farklı değişkenlik modelleri de üretilmektedir. Bileşen modelleri, UML gibi standartlaşmış modelleme araçlarını da etkilemiştir. Örneğin arayüz türlerinin tanımlanması önce bileşen modellerinde sonra UML'de aynı sembolleri kullanarak gerçekleştirilmiştir. UML araçları, kendilerini 'nesneye yönelik' olmanın yanısıra, 'bileşen tabanlı' olarak da tanıtabilmektedirler. Bu tanıtımın anlamı, esas olarak nesne kavramı yönelimi ile modeller geliştirilmekte, ancak bazı alt problemler için hazır çözümlere mimari tasarımda yer verilebilmesidir. Farklı soyutlama düzeylerini ele alan, farklı kapsamlı bileşen modelleme yaklaşımları ve bunları destekleyen araçlar bulunmaktadır. Temel hedeflerini göz önüne alarak, bir modelden yola çıkıldığında gerekecek bileşenlerin bulunup uyarlanıp birleştirilmesi adımlarının araçlar ile desteklenmesi önemli olmaktadır. Bileşenler yapıları açısından belirli bir uygulama sahasına yöneliktirler. Bu konuda en erken ve başarılı bir örnek, Grafiksel Arayüz Tasarımı alanıdır. Çoğu geliştirme ortamı, kaydırma çubuğu, ana menü, düğme gibi standart ekran elemanlarını bileşenler olarak sunmuşlardır. Dolayısı ile önceleri çok vakit alan kullanıcı arayüzü geliştirme işlemleri artık kod yazmadan yapılabilmektedir. Önceleri platform düzeyinde çözümler aranan bileşen teknolojileri, günümüzde farklı uygulama sahasına özel olarak başarı göstermektedirler. Örneğin AUTOSAR [9], otomotiv alanında bileşen tabanlı tasarım yapmak için yaygınca kullanılan bir bileşen modelidir. Bileşen modellerinin önemli bazı özelliklerini aşağıdaki gibi özetlemek mümkündür:

- Kapsadıkları soyutlama düzeyleri,
- Gerçekleştirme desteği verecek araçların bulunması,
- Uygulama alanı ve
- Değişkenlik yönetimi.

Referans makalemiz [3]'te listelenen bileşen modellerinden AUTOSAR ve KobrA [10] farklı düzeylerde değişkenlik yaklaşımları içermektedir. Bölüm 4.1 ve 4.2'de sırasıyla bu modellerdeki yaklaşımlara kısaca değinilmiştir. Diğer modellerde değişkenlik yönetimi ile ilgili beklentileri karşılayacak düzeyde bir modelleme desteği olduğu gözlemlenmemiştir. Ancak bazı modellerin değişkenliğe

benzer yaklaşımları vardır. Bunlara bölüm 4.3'te değinilmiştir. Referans makalemizde yer alan ancak bu çalışmada adı geçmeyen diğer bileşen modellerinde ise değişkenlik yaklaşımı ya da benzer bir yaklaşım saptanmamıştır.

4.1 AUTOSAR Değişkenliği

AUTOSAR otomotiv alanındaki gömülü sistemlerde kullanılmak üzere geliştirilmiştir. Temel amacı ortamdan bağımsız ve taşınabilir yazılım bileşenleri oluşturmaktır. AUTOSAR'ın hiçbir sürümünde tam anlamıyla değişkenlik yönetimi desteklenmemiştir. Ancak 4.0 sürümünden sonra değişkenliği kullanan bir yaklaşım getirilmiştir. 4.0'dan önceki bazı sürümlerde (3.x) model elemanlarına ek bilgiler eklenerek ve araca özel olmak üzere değişkenlik kullanılabilir [11]. 4.0 ve sonraki sürümlerde değişkenlik kullanımı nitelik modelleriyle kısıtlı da olsa tanımlanmıştır. Bu kısıtlara örnek olarak elemanlar arası bağlılıkların gösterilememesi verilebilir. 4.0 sürümünde dahi metamodelde değişkenlik yönetimi düşünülmemiştir. Ancak dışarıdan kullanılacak pure::variants [12] gibi bazı araçlarla değişkenlik yönetilebilir.

4.2 KobrA Değişkenliği

KobrA bileşen tabanlı geliştirme için kullanılan model güdümlü (model-driven) bir yaklaşımdır. KobrA bileşen modeli UML modelini karar modeli ile birleştirerek bileşen değişkenliğini tanımlamaktadır. Bir karar modeli en az bir hareket aksiyomu içeren formel bir sistemdir. KobrA'da genel değişkenlik modelinin gösterimi uygulamadan bağımsızdır ve en uygun geliştirme metodunun seçilebilmesi mümkün olmaktadır. Her bir değişkenlik noktası bir karara bağlıdır ve her bileşen yapısal, davranışsal ve fonksiyonel modelin dışında karar modeli ile ilişkilendirilmektedir. Burada tanımlanan değişkenlik yönetimi yazı tabanlıdır. Bu da değişkenlik yönetimini kimi zaman araç desteğine rağmen karmaşık hale getirebilmektedir.

4.3 Diğer Modellerde Değişkenlik

OSGi [13], bileşen-tabanlı geliştirim için tekrar kullanılabilirliği sağlayan dinamik ve modüler bir mimari sunan bir yaklaşımdır. Geliştirilmiş olan paket ve sınıfların zaman içinde değişimini ve yenilerinin sisteme dahil edilebilmesini kolaylıkla yönetebilmek için 'yazılım yığını' (bundle) adı verilen bir modül yapısı tanımlamıştır. Yazılım yığınları arayüzlerinde kendi özelliklerini, dışarıya verdiği servisleri, dışarıdan istediği ve bağımlı olduğu servisleri tanımlar. OSGi'nin güçlü yanı ise çalışan uygulama durdurulmadan dinamik olarak yazılım yığınlarının ekleme, çıkarma, yükleme ve kaldırma işlemlerine olanak sağlamasıdır. Ancak OSGi ile entegre çalışan ayrı bir değişkenlik modeli bulunmamaktadır [14].

Fractal [15], sistemlerin ve uygulamaların tasarımı, hayata geçirilmesi, dağıtımı ve yeniden yapılandırılmasını sağlayan modüler, genişletilebilir ve programlama dillerinden bağımsız bir bileşen modelidir. Fractal modeli bileşenlerin

dinamik olarak yeniden yapılandırılmasını destekler. Her bir Fractal bileşeni, bileşenin yeniden yapılandırılmasını kontrol etmek için 'membrane' adı verilen bir konteyner bulundurmaktadır. Ayrıca ROBOCOP [16], OpenCOM [17] ve SOFA 2.0 [18] bileşen modellerinde de farklı şekillerde dinamik yeniden yapılandırma desteklenmektedir.

Genel olarak endüstriyel otomasyon alanında kullanılan IEC 61499 [19] modelinde, düşük soyutlama seviyesinde fonksiyon bloklarının girdileri değiştirilerek yeniden yapılandırma sağlanabilmektedir. Ancak bu yeniden yapılandırma uygulama seviyesinde desteklenmemektedir [20].

Koala elektronik alanındaki gömülü sistemlerde kullanılmak üzere geliştirilen bir bileşen modeli ve mimari tanımlama dilidir [21]. Temel amacı yazılımların artan karmaşıklığını ve çeşitliliğini (diversity) yönetebilmektir. Koala'da bileşenlerin yapısal farklılığını yönetmek için özel bir anahtar bağlayıcı (switch connector) bulunmaktadır. Burada belirtilen anahtar bağlayıcı, girilen veriye bağlı olarak birden fazla bağlantıdan birini seçebilme özelliğine sahiptir. Bu özel mekanizma mod değişimini (mode switch) sağlamaktadır.

SaveCCM [22] araç kontrol sistemlerinde kullanılmak üzere geliştirilen bir bileşen modelidir. Koala'da kullanılan anahtar bağlayıcının aynısı bu modelde de kullanılmaktadır.

BlueArX bileşen modeli [23] donanım maliyeti nedeniyle sınırlı kaynaklara sahip otomotiv alanları için geliştirilmiştir. Bu model kendi yazılım bileşenleri tarafından üretilen çoklu mod (mode) uygulamalarını destekler. Mod, anlamsal içerik bilgisinin bir türü olarak kabul edilmektedir. Farklı modlar farklı kontrol stratejilerini ifade eder. Bu modelde çoklu mod sistemi tamamen inşa edilmeden BlueArX bileşeninin modu açıkça tanımlanmamaktadır [24].

Rubus [25] kara taşıtlarında bulunan gömülü kontrol sistemleri için geliştirilmiştir. Rubus'ta mod sistemin bir özelliği olarak kabul edilir. Bileşenlerin sabit yapılandırılması her bir mod için sistem genelinde olmaktadır. COMDES-II [26] modeli ise gerçek zamanlı kısıtlamalara sahip dağıtık gömülü kontrol sistemleri için bir bileşen tabanlı yazılım çerçevesi tanımlamaktadır. COMDES-II farklı modlarda bileşen yapılandırılmasını sağlamak için bir durum makinesi kullanmaktadır [27].

4.4 Gözlemler

Ele alınan bileşen modellerinde yazılım üretim hatlarında ulaşılmış bulunan olgunluk seviyesine yakın bir değişkenlik modellemesi bulunmamaktadır. Çoğu bileşen modelinde değişkenlik ele alınmamışken, bir kısmında bileşen tümleştirmesinin doğal bir adımı olarak ortaya çıkan bileşen değiştirme (tak/çıkar) adımının dolaylı olarak değişkenlik konusunu desteklediği görünmektedir. Ancak bu düzeydeki bir destek, ayrı bir değişkenlik modelinden olan beklentileri karşılamak için yeterli değildir. Çoğu bileşen modelinin soyut düzeylerden çok kod düzeyinde araçlar ve modelleme mekanizmaları ile desteklendiğini söylemek yanlış olmaz. Karmaşıklığı yönetmek üzere soyut düzeylerden başlayacak bir modeldense, kısmen tasarımının mevcut olduğu varsayılan bir sistemin kod parçalarının hızlı bir

şekilde bulunup birleştirilmesi öne çıkmış gibidir. Ulaşılmış olan sonuçları kısaca sunacak olursak:

1. Ayrık bir değişkenlik modeli ile desteklenme kabiliyeti yok denecek kadar azdır.
2. Değişkenlik daha çok tümleştirme düzeyinde ve minimal model soyutlaması ile ele alınmaktadır.
3. Değişkenlikler arası kısıt yayılımı çözümlemesi eksiktir.
4. Farklı modeller arası birlikte çalışabilirlik, modellerin kod düzeyine verdikleri önem sonucunda kaybedilen soyutlama ile birlikte zayıftır.
5. Alana yönelik modellerde değişkenlikler daha anlamlı olarak modellenebilecek temellere sahiptir.

5 Öneriler

Araştırmada bileşen modelleri için değişkenlik desteğinin yeteri kadar detaylı ve kavranabilir olmadığı sonucuna varılmıştır. Aşağıdaki problemler hala çözüm beklemektedir:

- Değişkenlik kolay yönetilebilirlik için işlerin ayrılması prensibine uygun olarak açık ve ayrı bir model olarak ele alınmalıdır [28]. Nitelik modelleri problem uzayıyla iç içedir ve çözüm uzayındaki gereksinimleri karşılamaktan uzaktır. Değişkenlik noktaları, değişkenler, bunların arasındaki ilişkiler ve bileşen tabanlı yazılım unsurlarıyla eşleştirilmeleri diğer işlerden ayrı yapılmalıdır.
- Değişkenlik geliştirimden analiz ve test düzeyine kadar tüm yazılım geliştirme süreçlerinde yönetilebilmelidir. Yazılım doğrulaması genellikle nitelik modeli seviyesinde uygulanmaktadır. Ancak bileşen modellerinin tutarlılığı tasarım aşamasında kontrol edilmelidir.
- Değişkenlik, sistemin gerçek davranışını belirleyecek olan bileşen tümleştirilmesine de uygulanabilir olmalıdır.
- Çalıştırılabilir bir sistem için bileşenlerin yanında bir süreç modeli de gerekir ve değişkenlik yönetimi bu iki farklı dünyayı (bileşenler ve süreçler) tutarlı bir şekilde desteklemelidir.
- Değişkenlik hiyerarşik olarak yukarıdan aşağıya olabildiğince otomatik bir şekilde çözümlenebilmelidir [6].
- Değişkenlik; modeldeki bileşen, bağlayıcı, arayüz ve birleşim unsurlarının tümüne uygulanabilmelidir.

Bu öneriler göz önünde bulundurularak aşağıdaki gibi bir çözüm teklif edilmektedir:

Önceki çalışmalarımızdan [29]'da COSEML dili (Component Oriented Software Engineering Modeling Language) değişkenlik yönetimi yaklaşımıyla genişletilmiştir. Oluşturulan yeni dilde (XCOSEML), COSEML'deki statik unsurların yanına; 'dinamik unsurlar', 'değişkenlik unsurları', statik ve dinamik unsurlarla

değişkenlik unsurlarını eşleyen 'eşleştirme unsurları' eklenmiştir. XCOSEML metamodeli, işlerin ayrılması prensibine sadık kalınarak 3 ana parçadan oluşturulmuştur. Bu parçalar; 'COSEML belirtimi' (COSEML Specification), 'eşleştirme belirtimi' (Mapping between COSEML and Varibility Specifications), ve 'değişkenlik belirtimidir' (Variability Specification). Şekil 1'de metamodele genel bir bakış gösterilmektedir. Şekil 2 metamodelin COSEML belirtimi parçasını gösterirken, Şekil 3'te eşleştirme ve değişkenlik belirtimleri beraber gösterilmiştir. Metamodelin detaylı anlatımı ve XCOSEML diliyle oluşturulan örnek sistemlere [29][30][31] çalışmalarından ulaşılabilir.

Şekil 1. XCOSEML metamodeline genel bakış.

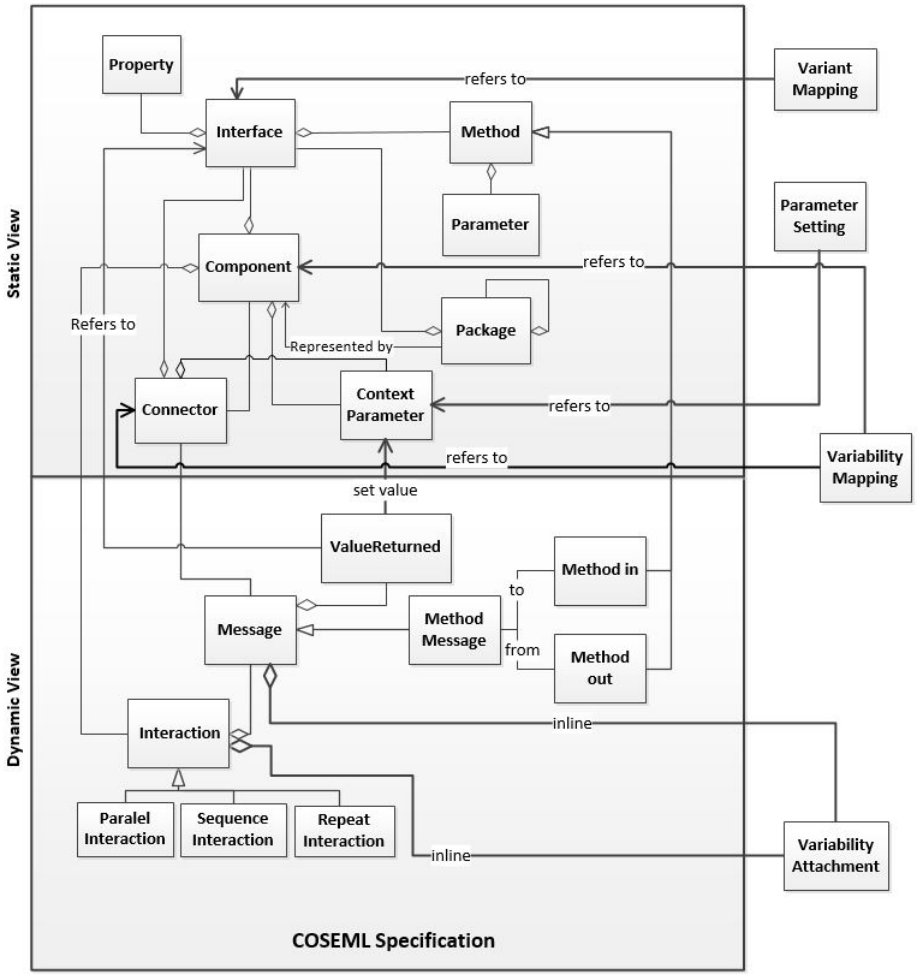
Yazılım doğrulaması prensibine uymak için ise metamodele dayanarak oluşturulan XCOSEML modelleri için doğrulama çalışması yapılmıştır. Modeller dil dönüşümüyle bir model denetleme aracının girdisine uygun hale getirilerek çalışabilirlikleri kontrol edilmiştir [30][31].

XCOSEML dilinde, öncüsü COSEML'de olduğu gibi bağlayıcılar (connector) birinci sınıf unsur olarak ele alınmış ve modellemeye dahil edilmiştir. Buna karşın bağlayıcı tanımlaması soyut seviyede kalmış ve değişkenlik unsurları bağlayıcılara uygulanmamıştır. Bu çalışmamızda XCOSEML dilindeki bağlayıcıların değişkenlik unsurlarıyla eşleştirilmesi metamodel seviyesinde gösterilmektedir. Bağlayıcı ve çevre unsurlarla olan bağlantıları Şekil 2'nin 'sabit bakış' (Static View) kısmında görülebilir.

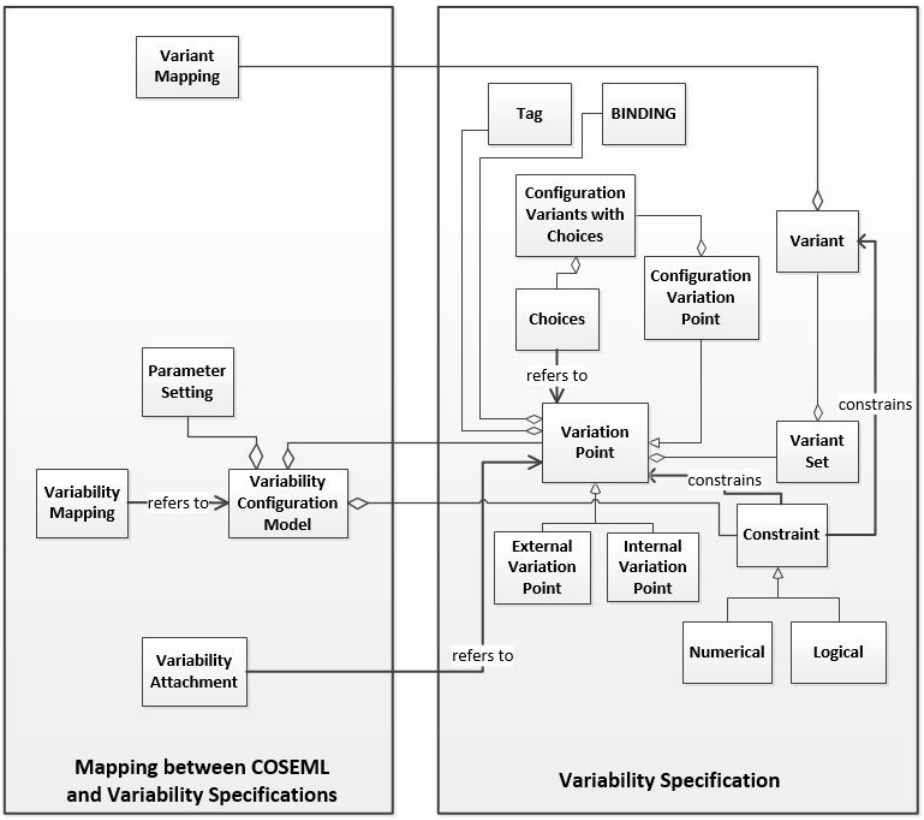
Bağlayıcılar; verimlilik, karmaşıklık, genişletilebilirlik gibi işlev-dışı özellikler başta olmak üzere sistemin karakteristiğini belirlemede önemli rol oynar. Bu nedenle yazılım sistemine en uygun bağlayıcının seçilmesi, ya da mevcut bağlayıcıların istenen şekilde düzenlenebilmesi önem kazanır. Ortam (context) özelliklerinin ve sistemde kullanılan bileşenlerin bağlayıcı seçimini ya da bağlayıcıların içeriğini kısıtlaması da söz konusudur. Bu konular göz önüne alınarak XCOSEML metamodelinde aşağıdaki genişletmeler yapılmıştır:

– XCOSEML'de içeriği tanımlanmamış olan bağlayıcı (Connector) kutusu, Mehta vd. tarafından yapılan bağlayıcı sınıflandırma çalışmasındaki [32] 4 temel bağlayıcı görevini (Bağlantı (Communication), Düzenleme (Coordination), Dönüştürme (Conversion), Kolaylaştırma (Facilitation)) tanımlamak üzere arayüzle (Interface) bağlanmıştır. Değişkenlik unsurlarıyla bağlı olan arayüz sayesinde bağlayıcının, sağladığı etkileşim hizmetleri için, farklı ara-

yüzlere sahip olması sağlanacak ya da bir arayüz aracılığıyla gösterilen hizmetlerinin geliştirici tarafından seçilmesine imkan tanınacaktır.
– Üst seviyede bağlanan bir değişkenlik noktasının bağlayıcı üzerinde olan etkilerinin gösterilmesi değişkenlik eşleştirmesi (Variability Mapping) kutusuyla olan bağlantıyla ifade edilmektedir.
– Ortamı ifade eden bazı değişkenlerin (örneğin ortam sıcaklığı, donanım özellikleri vb.) bağlayıcıların işlevini etkileme durumu söz konusudur. Bu durum bağlayıcı ile ortam parametreleri (Context Parameters) kutusunun bağlanmasıyla ifade edilmiştir.



Şekil 2. XCOSEML metamodel - COSEML belirtimi. COSEML belirtimi dışında kalan kutular, Eşleme Belirtimi kısmında yer almaktadırlar.



Şekil 3. XCOSEML metamodel - Eşleştirme ve değişkenlik belirtimleri ([29]'dan uyarlanmıştır).

6 Sonuç

Yapılmakta olan sistematik yazın taraması çalışmasının bir ara evresinde bileşenlerde değişkenlik modellemesi ile ilgili önemli gözlemlerde bulunulmuş ve bu konudaki yaklaşımlar kısmen iyileştirme önerileri ile desteklenmiştir. Yazarlar daha önce nitelik modelleri ve Servis Yönelimli Mimari platformlarında da, burada sunulan öneriler ile ilgili durum çalışmaları yapmış olup [33] benzeri çözümlerin bileşen modellerine de taşınabileceği umulmaktadır. Bir taraftan geleceğe yönelik olarak kapsamlı yazın taraması devam etmekte ve yakında daha ayrıntılı sonuçlara ulaşılacağı varsayılmaktadır. Bu çalışmanın sonrasında da ilk değerlendirme kümesinde bulunan bileşen modellerini aşarak benzeri bir çalışma, gelişmekte olan bileşen modellerine de genişletilebilir.

Kaynaklar

1. Kim S. D., Her J. S., Chang S. H.: A theoretical foundation of variability in component-based development. Information and Software Technology, Volume 47, Issue 10, 663–673 (2005)
2. Gurp J. V., Bosch J., Svahnberg M.: On the Notion of Variability in Software Product Lines. In: Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA '01). IEEE Computer Society, Washington, DC, USA (2001)
3. Crnkovic I., Sentilles S., Vulgarakis A., Chaudron M. R. V.: A Classification Framework for Software Component Models. IEEE Transactions on Software Engineering. 37, 593–615 (2011)
4. Kang K. C., Kim S., Lee J., Kim K., Shin E., Huh M.: FORM: A feature-oriented reuse method with domain-specific reference architectures. Ann. Softw. Eng. 5, 143–168 (1998)
5. Pohl, K., Böckle, G., van Der Linden, F. J.: Software product line engineering: foundations, principles and techniques. Springer Science and Business Media (2005)
6. Sinnema, M., Deelstra, S., Nijhuis, J., Bosch, J.: Covamof: A framework for modeling variability in software product families. In Software product lines, Springer, 197–213 (2004)
7. Szyperski, C.: Component software: beyond object-oriented programming. Harlow, Engl. Addison-Wesley (1995)
8. Dogru, A. H., Tanik, M. M.: A process model for component-oriented software engineering. IEEE Softw. 20, 34–41 (2003)
9. AUTOSAR (AUTomotive Open System ARchitecture), `https://www.autosar.org`
10. Atkinson, C.: Component-based product line engineering with UML. Pearson Education (2002)
11. Schulze, M., Weiland, J., Beuche, D.: Automotive model-driven development and the challenge of variability. In: Proceedings of the 16th International Software Product Line Conference-Volume 1, pp. 207–214 (2012)
12. pure-systems GmbH: pure::variants Eclipse Plugin User Guide, (2011)
13. OSGi Alliance, "OSGi Service Platform Core Specification, V4.1," (2007)
14. OSGi Alliance, https://www.osgi.org/
15. E. Bruneton, T. Coupaye, J. Stefani,: The Fractal Component Model Specification, The ObjectWeb Consortium, technical report, `http://fractal.objectweb.org/specification/index`. (2004)
16. Maaskant H.: A Robust Component Model for Consumer Electronic Products. vol. 3, Springer, 167–192 (2005)
17. Clarke M., Blair G., Coulson G., Parlavantzas N.: An Efficient Component Model for the Construction of Adaptive Middleware. In: Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms. (2001)
18. Bures T., Hnetynka P., Pla'sil F.: SOFA 2.0: Balancing Advanced Features in a Hierarchical Component Model. In: Proc. Int'l Conf. Software Eng. Research, Management and Applications, 40–48 (2006)
19. Vyatkin V., Instrument Society of America: IEC 61499 function blocks for embedded and distributed control systems design. ISA-Instrumentation, Systems, and Automation Society, (2007)
20. Froschauer R., Dhungana D., Gruenbacher P.: Managing the Life-cycle of Industrial Automation Systems with Product Line Variability Models. In: 34th Euromicro Conference Software Engineering and Advanced Applications, Parma, 35–42 (2008)

21. Van Ommering, R., Van Der Linden, F., Kramer, J. Magee, J.: The Koala component model for consumer electronics software. Computer (Long. Beach. Calif). 33, 78-85 (2000)
22. M. Akerholm, J. Carlson, J. Fredriksson, H. Hansson, J. Hakansson, A. Moller, P. Pettersson, and M. Tivoli: The SAVE Approach to Component-Based Development of Vehicular Systems. J. Systems and Software, vol. 80, no. 5, 655-667 (2007)
23. Kim J.E., Rogalla O., Kramer S., A. Haman: Extracting, Specifying and Predicting Software System Properties in Component Based Real-Time Embedded Software Development. In: Proc. 31st Int'l Conf. Software Eng. (2009)
24. Hang Y.: Introducing Mode Switch in Component-Based Software Development. (2015)
25. Hanninen K., Maki-Turja J., Nolin M., Lindberg M., Lundback J., Lundback K.: The Rubus Component Model for Resource Constrained Real-Time Systems. In: Proc. Int'l Symp. Industrial Embedded Systems. 177-183 (2008)
26. Ke X., Sierszecki K., Angelov C.: COMDES-II: A Component-Based Framework for Generative Development of Distributed Real-Time Control Systems. In: Proc. 13th IEEE Int'l Conf. Embedded and Real-Time Computing Systems and Applications. 199-208 (2007)
27. Hang Y., Hongwan Q., Jan C., Hans H.: Mode switch handling for the ProCom component model. In: Proceedings of the 16th International ACM Sigsoft symposium on Component-based software engineering. ACM, (2013)
28. Metzger A., Pohl K., Heymans P., Schobbens P. Y., Saval G.: Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis. In: 15th IEEE International Requirements Engineering Conference (RE 2007), Delhi. 243–253 (2007)
29. Kaya M. C., Suloglu S., Dogru A. H.: Variability Modeling in Component Oriented System Engineering. In: The 19th International Conference on Transformative Science and Engineering, Business and Social Innovation, Kuching, Sarawak, Malaysia. (2014)
30. Kaya. M. Ç., Modeling Variability in Component Oriented Software Engineering. MSc Thesis, Middle East Techical University. (2015)
31. Kaya M. C., Saeedi Nikoo M., Suloglu S., Dogru A. H.: Towards Verification of Component Compositions Incorporating Variability. In: The 20th International Conference on Transformative Science and Engineering, Business and Social Innovation, 202–207. (2015)
32. Mehta, N. R., Medvidovic, N., Phadke, S.: Towards a taxonomy of software connectors. In: Proceedings of the 22nd international conference on Software engineering. 178–187 (2000)
33. Suloglu, S., Tekinerdogan, B., Dogru, A. H.: XChor: Chreography Language For Integration of Variable Orchestration Languages. In: 3rd International Symposium on Business Modeling and Software Design, 2013, Noordwijkerhout, Netherlands. (2013)