Eşgüdümlü Bileşenler ile Birleştirme Yaklaşımı

Anıl Çetinkaya[1, 2], Alper Karamanlıoğlu[1], M. Çağrı Kaya[1], Ali H. Doğru[1]

[1]Orta Doğu Teknik Üniversitesi,
Ankara, Türkiye
[2]İskenderun Teknik Üniversitesi
Hatay, Türkiye
{cetinkaya,alperk,mckaya,dogru}@ceng.metu.edu.tr

Özet. Bu çalışmada bileşen yönelimli sistem geliştirme yaklaşımlarında süreç modeli kullanımını desteklemek üzere farklı kabiliyetler gerektiren bileşen yapıları önerilmektedir. Bu yapılar ile merkezi bir süreç denetimi sağlamak için sistemli bir yaklaşım ortaya konacaktır. Servis Odaklı Mimari'yi desteklemek üzere geliştirilen teknikler ve kabiliyetler, merkezi bir süreç modelini çalıştırılabilir bir sistemin ana unsuru olarak kullanmayı uygun hale getirmiştir. Bu mekanizma bileşen merkezli yaklaşımlar için de yararlı olacaktır. Bileşenlerin yayımlanan (published) ve gereken (required) arayüzleri bulunmaktadır ve bu arayüz tanımları ile hangi işlevleri verebileceklerinin yanı sıra hangi işlevlere ihtiyaç duyacakları belirtilebilmektedir. Bu mekanizmalar kullanılarak bileşenler arası işbirliği yapılabilmektedir. Ancak bir uygulama geliştirilirken yapılacak işlerin merkezi bir süreç güdümünde tetiklenmesini hedeflemekteyiz. Tümleştirme çabasını sistematik ve güvenilir bir şekilde yürütmek üzere bu tür bir yapı yararlı olacaktır. Bunun için de bileşenlerin ne zaman bir hizmet isteyecekleri ve bunun sonucunda diğer bir bileşenin ne zaman bir hizmeti vereceğinin merkezi bir süreç tarafından eşgüdüm açısından yönetilmesi gerekecektir. Bu çalışmada önerilen bileşen yapıları ile bileşenlerin gereken arayüzlerindeki işlevler dışarıdan etkinleştirilme kabiliyeti ile donatılmakta ve merkezi bir süreç denetimini destekler hale getirilmektedir. Karmaşık sistemlerin tümleştirme yolu ile çabuk geliştirilmesini hedefleyen yaklaşım, mesaj trafiği gibi verim düşürücü parametrelerin önemli olacağı durumları hedeflememektedir. Çalışma bir örnek üzerinden önerilen mimariyi sunmaktadır.

Anahtar Kelimeler: Bağlayıcı, Bileşen, Değişkenlik Yönetimi, Süreç Modeli

Integration Approach through Coordinated Components

Abstract. In this study, component structures that require different abilities to support the use of a process model in component oriented system development approaches are proposed. A systematic approach will be introduced with the use of these structures to provide a centralized process control. Techniques and capabilities developed to support Service Oriented Architecture have made it possible to use a centralized process model as the main constituent of an executable system. This mechanism will also be useful for component oriented approaches. Components have published and required interfaces, the functions provided by a component along with the functions required, are specified within these interface definitions. Collaboration between components can be conducted using these mechanisms. However, we aim to trigger the development of an application in a central and process driven manner. Proposed structure would be useful to carry out the effort of integration systematically and reliably. The time when one component requires a service that is offered by another component, should be managed by a central and process driven way. In this work, the proposed component structures and the functions of the required interfaces of the components are equipped with the abilities to be externally triggered and to support a central process control. With this approach, the aim is to contribute to the rapid development of complex systems through integration. Whereas, this approach does not aim the situations where efficiency reduction parameters such as message traffic. The approach is demonstrated through a proof of concept case study.

Keywords: Connector, Component, Variability Management, Process Model.

1 Giriş

Yeniden kullanım yazılım geliştirmede önemli bir kavramdır. Bileşen tabanlı yazılım mühendisliği (CBSE), önceden geliştirilmiş bağımsız bileşenlerin sistematik bir şekilde bir araya getirilerek yeni bir sistem oluşturulmasını amaçlar [1]. Yazılım sistemlerinin günümüzde giderek büyümesi ve daha karmaşık hale gelmesi ile yeniden kullanım prensibi, sistemlerin verimliliğinin artırılması ve genişletilebilme imkanı sunması açısından daha da önemli bir hale gelmiştir. Bileşen tabanlı sistemlerde yeniden kullanımın sağlıklı bir şekilde desteklenebilmesi için ortak ve değişken parçaların açık bir şekilde ifade edilmesi gerekmektedir [2].

Yazılım sistemlerinin giderek karmaşıklaşmasının sonucu olarak güncel sistemlerin daha dinamik ve uyarlanabilir olma gereksinimi ortaya çıkmıştır. Bu amaca ulaşmak için yazılım sistemlerinin değiştirilme, yapılandırma ve genişletme yeteneğine sahip olması gerekmektedir. Bu yetenekleri sağlamanın yollarından birisi de değişkenlik desteğidir [3]. Yazılım ürün hatları değişkenlik yönetimi ile yakından ilişkilidir. Yazılım ürün hatlarında değişkenliğin yönetilmesi için sistematik bir yaklaşım benimsenir. Yazılım varlıkları daha sonra kullanılmak üzere tanımlanmış bir yazılım mimarisine ve bir yazılım ürün ailesinin gereksinimlerine göre oluşturulmaktadır. Oluşturulmakta olan ürün ailesinin olgunlaşması ve değişkenliğin verimli bir şekilde yönetilmesi ile yeni ürünler oluşturmak için harcanan çaba, zaman ve maliyet azalır. Değişkenliğin etkin bir şekilde yönetilmesinde bağlanma zamanı önemli bir yere sahiptir [4].

Yazılım mimarisinde ve bileşen tabanlı yöntemlerde diğer bir önemli yapıtaşı da bağlayıcılardır (connector). Bileşenler yazılımın işlevselliğini temsil ederken, bağlayıcılar bileşenler arasındaki etkileşimlerden sorumludur. Bileşenlerin sistemin işlevselliği ve iletişiminden birlikte sorumlu olduğu geleneksel yaklaşımlarda, sistemi yeni bir bileşen ekleyerek güncellemenin veya mevcut bir bileşen üzerinde herhangi bir değişiklik yapmanın bileşenler arasındaki etkileşim üzerinde doğrudan etkisi vardır. Bu sebeple sistemdeki herhangi bir değişikliğin öngörülemeyen hatalara ve uyumsuzluklara yol açabilme olasılığı mevcuttur.

Bu çalışmada merkezi bir yapı olarak kullanılan süreç modeli üzerinden bileşenler arası iletişimin eşgüdümlü bir şekilde bağlayıcılar aracılığı ile sağlanması gösterilmektedir. Bileşenlere gerekli yeteneklerin kazandırılabilmesi için bileşen arayüzlerine dışarıdan tetiklenebilme kabiliyeti sağlanmakta ve merkezi bir süreç denetimi üzerinden kontrol edilebilir hale getirilmektedir. İşlerin merkezi bir yapı üzerinden yerine getirilmesi sayesinde daha anlaşılabilir ve kontrolü kolay bir sistem elde edilmesi, dolayısıyla ileride oluşması muhtemel hatalar ve uyumsuzluklardan arındırılmış daha güvenilir bir sistem hazırlanması planlanmaktadır.

Çalışmanın takip eden bölümünde yaklaşımın dayandığı temel konular hakkında bilgi verilmiştir. Üçüncü bölümde mevcut altyapı üzerine yapılan geliştirmelerden bahsedilerek önerilen yapının nasıl çalıştığı anlatılmıştır ve sistemin çalışması örnek bir çalışma üzerinden gösterilmiştir. Dördüncü bölümde ise önerilen yaklaşımın avantaj ve dezavantajları irdelenerek yazılım geliştirme üzerine potansiyel etkileri değerlendirilmiştir. Beşinci bölümde literatürdeki ilgili yaklaşımlar sunulurken altıncı bölümle çalışma sonlandırılmaktadır.

2 Mevcut Altyapı

Bu bölümde, önerilen mimari için gereken alt yapıdan bahsedilmiştir. Bileşenler, bağlayıcılar, değişkenlik yönetimi ve bileşen yönelimli modelleme dili XCOSEML bölüme dahil edilmiştir.

2.1 Bileşenler

Bir yazılım bileşeni, bir bileşen modeline uyan ve bir birleşim standardına göre değişim olmaksızın bağımsız olarak konuşlandırılarak oluşturulabilen bir yazılım öğesidir. Bileşenler, bağımsız kullanılabilmelerinin yanı sıra daha büyük ve karmaşık sistemleri üretebilmek için diğer bileşenler ile tümleşik olarak da kullanılabilirler. Sağlamış oldukları yeniden kullanım teknolojisi sayesinde yazılım sistemlerini sıfırdan oluşturmak yerine yeniden kullanım ile daha kolay geliştirme imkanı sunmaktadırlar.

Bileşenlerin standart bir bileşen modeline uygun olmaları gerekir. Günümüzde otomotiv yazılımı ve tüketici elektroniğinde kullanılan gömülü sistemlerden telekomünikasyon, finans, sağlık ve ulaşım gibi iş alanlarına kadar uzanan birçok uygulama alanını hedefleyen bileşen modelleri bulunmaktadır. Bazı bileşen modelleri ise doğrudan teknoloji platformlarına dayalıdırlar.

Bileşen modellerinde bileşenler mimari birimlerdir. Her bir bileşen, bileşenin sağladığı hizmetleri gösteren bir arayüze ve doğru bir şekilde çalışması için gereken servislere sahip olmalıdır. Böylece işlevler ve davranışlar ortaya konulabilirken uygulama ayrıntıları gizlenebilmektedir.

2.2 Bağlayıcılar

Günümüzde modern yazılım sistemleri pek çok karmaşık bileşenden oluşmaktadır. Bu bileşenler arasındaki etkileşimlerin doğru ve verimli bir şekilde sağlanması sistemin kararlılığının ve genişletilebilirliğinin sağlanması açısından büyük önem arz etmektedir. Bileşenlere sistemin işlevselliğini sağlamanın yanında iletişim sorumluluğunu da yüklemek sistemin kontrolünü zorlaştırmaktadır.

Bağlayıcının tanımı “Bileşenler arasında etkileşimi sağlamak ve düzenlemek için görevlendirilen mimari öğe” olarak verilmiştir [5]. Bu tanıma göre bağlayıcılar bileşenler arasındaki etkileşimden sorumlu mimari seviyedeki soyutlamalardır. Basit bir ifadeyle, bileşenler arasındaki kontrol ve verilerin aktarılmasından sorumludurlar. Bağlayıcılar bir uygulamanın birleşiminde belirli roller üstlenirler. Bileşenler ise genellikle uygulama veya bağlamdan bağımsızdırlar. Bileşen ve bağlayıcıların bu yapıları yazılım mühendisliğindeki ilgilerin ayrılması prensibine hitap etmektedir. Bileşenler arasındaki iletişime göre farklı görevler üstlenebilmeleri ve bileşenler üzerinde yapılabilecek değişiklerden sonra dahi tekrar kullanılabilir olmaları bileşen tabanlı geliştirmede bağlayıcıların işlevselliklerini artırmaktadır.

Bağlayıcılar, sundukları hizmetlere göre Mehta vd. [6] tarafından dört genel sınıfa ayrılmışlardır. Bunlar İletişim (Communication), Eşgüdüm (Coordination), Dönüşüm (Conversion) ve Kolaylaştırmadır (Facilitation).

İletişim bağlayıcıları bileşen etkileşimlerinde en önemli yapı taşlarından birisi olan veri aktarımında kullanılır. Eşgüdüm bağlayıcıları bileşenler arasında kontrol

aktarımından sorumludur. Dönüşüm bağlayıcılarının amacı heterojen sistemlerde bileşenlerin birbiri ile etkileşime girmesine olanak sağlamaktır. Kolaylaştırma bağlayıcıları ise bileşenler arasındaki etkileşimlere arabuluculuk etmek ve bunları düzene koymak için kullanılır.

Bağlayıcıların sağladığı bu servislere karşılık Prosedür Çağrısı (Procedure Call), Olay (Event), Veri Erişimi (Data Access), Bağlantı (Linkage), Akış (Stream), Sıra Düzenleyici (Arbitrator), Uyarlayıcı (Adaptor) ve Dağıtıcı (Distributor) olmak üzere sekiz adet bağlayıcı türü tanımlanmıştır. Bu bağlayıcı türleri ve sağladıkları servisler Tablo 1'de verilmiştir.

Tablo 1. Bağlayıcı türleri ve karşılık gelen servisleri ([6]'dan uyarlanmıştır).

		Verilen Servisler			
		İletişim (Communication)	Eşgüdüm (Coordination)	Dönüşüm (Conversion)	Kolaylaştırma (Facilitation)
Bağlayıcı Tipleri	Prosedür Çağrısı (Procedure Call)	✓	✓		
	Olay (Event)	✓	✓		
	Veri Erişimi (Data Access)	✓		✓	
	Bağlantı (Linkage)				✓
	Akış (Stream)	✓			
	Sıra Düzenleyici (Arbitrator)		✓		✓
	Uyarlayıcı (Adaptor)			✓	
	Dağıtıcı (Distributor)				✓

Mehta vd. tarafından belirlenen bu sınıflandırmalar XCOSEML'deki bağlayıcı tanımına eklenmiştir. Dildeki bağlayıcı yapısında bulunan bağlayıcı tipini temsil eden "ConnectorType" ve servis tipini temsil eden "ServiceType" belirleyicileri, geliştiricilere ilgili bağlayıcının amacı ve yapısı hakkında bilgi vermek için kullanılmaktadır. Bu sınıflandırmalar değişkenliğin yönetilmesinde önem arz etmektedir.

2.3 Değişkenlik Yönetimi

Değişkenlik yönetimi, yaşam döngüsü boyunca yazılım eserlerindeki (artifacts) değişkenliği açıkça ifade etme, farklı değişkenlikler arasındaki bağımlılıkları yönetme ve bu değişkenliklerin örneklerini destekleme faaliyetlerini kapsar. Değişkenliği yönetmenin karmaşıklığı, daha sistematik yöntemlerin kullanılmasını gerektirmektedir.

Bu doğrultuda, değişkenliğin gereksinimlerden kaynak koduna yazılım geliştirmenin her seviyesinde yönetilebilmesini sağlayan çeşitli değişkenlik modeli yaklaşımları önerilmiştir. Bu modeller arasında en yaygın olarak kullanılanları OVM [7], Covamof [8] ve Nitelik Modelleridir [9]. Değişkenlik modelleri, bileşen modelleri ile birlikte kullanılarak bu modellere değişkenlik yönetimi kabiliyeti kazandırabilmektedirler.

Değişkenlik, değişkenlik noktaları (variation points) ve bu noktalar üzerinde sunulan seçenekler (variant) aracılığıyla gerçekleştirilir. Bileşen üzerinde tanımlanabilmesinin dışında bağlayıcı, arayüz ya da süreç üzerinde de değişkenlik tanımlanabilmektedir. Arayüz değişkenliği, işlevlerin ve parametrelerin nasıl ve ne zaman değiştirileceğini belirten bir yapılandırma mekanizmasını gerektirir. Bağlayıcı değişkenliği, iki bileşen arasında ne zaman ve hangi bağlayıcının kullanıldığını belirtmek için bir ilişki mekanizmasına ihtiyaç duyar. Süreç değişkenliği ise hizmetlerin birbirleri ile hangi sırayla ve nasıl etkileşim kurduğunu tanımlamak için bir uyarlama (tailoring) mekanizmasını gerektirmektedir.

2.4 XCOSEML

XCOSEML [10] bileşen yönelimli yazılım geliştirme için değişkenlik desteğiyle donatılmış metin tabanlı bir dildir. Dilin metamodelinde dil belirtimleri ve değişkenlik belirtimleri ayrı ayrı ele alınmakta ve eşleştirme belirtimleriyle aralarındaki bağlantı kurulmaktadır. Böylece ilgilerin ayrılması sağlanır ve modellerin yönetimi kolaylaşır.

XCOSEML'in öncüsü COSEML [11] bileşen yönelimli yaklaşımı desteklemek üzere geliştirilmiştir. Bu yaklaşımda tümleştirme hedefi öne çıkmış ve kod yazmayı hedefleyen çoğu yaklaşımın aksine nesne gibi bileşen dışındaki kavramlara yer verilmemiştir. Bileşenlerin soyut ve somut düzeylerde ele alınması ile olgunlaşmış uygulama sahalarında (domain) çabuk geliştirmeler hedeflenmiştir. Ancak bu öncü dil daha çok sistemlerin statik modellemesine yer vermiş ve bağlayıcıları yalnızca bağlama noktaları arasında yer alan bir isim beyanı düzeyinde içermiştir. Dinamik modelleme kabiliyeti XCOSEML ile tanımlanmıştır. Bu tanımlamada süreç belirtimi (composition specification) doğrudan bileşenler arasındaki mesajlaşmaları arayüzleri aracılığıyla göstermektedir. Dile bağlayıcı yeteneği kazandırılmasıyla beraber [12] mesajlaşmalar bağlayıcıların içine taşınmış ve süreç belirtiminde bağlayıcılara yer verilmiştir. Bu çalışmada ise bağlayıcıların mesajlaşma sıralaması için kullanımına ve bileşen eşgüdümü modellemesine yeni bir yaklaşım getirilmektedir.

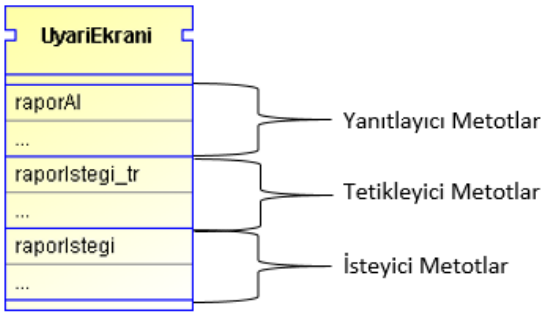
XCOSEML, değişkenlik içeren saha modelinden (domain model) çalışan bir ürün elde edilmeden önce yazılım doğrulaması amacıyla model kontrolüne (model checking) imkan vermektedir. Bunun için yarı otonom bir araç desteği [13]'te önerilmiştir. Model kontrolü için SNIP [14] aracı kullanılmıştır. SNIP aracı FTS (Featured Transition System) [15] yaklaşımıyla çalışır ve kendine özgü metin dilleriyle doğrulanmak istenen sistemin süreç modelini ve değişkenlik modelini girdi olarak alır. XCOSEML modellerinin doğrulanması için SNIP aracının bu girdi dillerine dönüşüm yapılması gerekir. Diller arasındaki anlamsal farklılıklardan dolayı bu dönüşüm tam olarak araç desteğiyle yapılamamakta ve bazı noktalarda kullanıcı müdahalesi gerekmektedir.

3 Önerilen Yapı

Önerilen sistemde bileşen yapılarına eklemelerde bulunulmuştur. Bileşenlerin dışarıdan tetiklenmesi amacı ile kullanılan tetikleyici (trigger) metotlar, dışarıdan istekte bulunmakta kullanılan isteyici metotları (method out) harekete geçirmek için eklenmiştir. Bu kabiliyetler COSEML ve XCOSEML dillerine eklenmiştir. Ancak örnekler sunulurken yer kısıtı ve fazla yarar getirmeyeceği değerlendirmeleri sonucunda bu dillerdeki modeller yansıtılmamıştır. Ayrıca önerilen katkı üst düzey model sunumuna da fazlaca karşı düşmemektedir.

Yazılım dünyasında en yaygınca kullanılan "method" terimi, yayınlanmış (published) işlevlere karşı düşmektedir ve COSEML'de "method-in" (yanıtlayıcı görevini üstlenir) olarak tanımlanır. Dışarıdan çağrılır ve bir hizmet verir. COSEML'deki "method-out" tanımı ise gereken (required) işlevlere karşılık gelir ve isteyici görevini üstlenir. Bu kavram hem bileşen modellerinde hem de UML 2.0'da [16] mevcuttur ve dışarıdan çağrılacak bir işlevi tanımlar. Bu çalışmada isteyici metotlar, çalıştırılabilen kod parçalarıdır ve tetikleyici metotlar tarafından başlatılır.

Tetikleyici metodun bir isteyici metodun görevini dışarıdan bazı varlıklar tarafından çalıştırılmak üzere üstlenmesi bile mümkün olmaktadır. Ancak, daha çok tam çalışma anında ne tür parametre değerlerinin sağlanacağını istekte bulunacak bileşenden başka bir varlık bilemez. İşlevin adı, hatta parametre listesi bile biliniyor olsa da gerçek çağrıyı, isteyici metoda sahip olan bileşenin yapması doğru olur. Bu açıdan tetikleme mekanizması geliştirilmiştir. Önerilen sisteme uygun olarak tasarlanan bir bileşen yapısı Şekil 1'de gösterilmektedir. Bu bileşen yapısı bahsedilen metotları içermektedir.

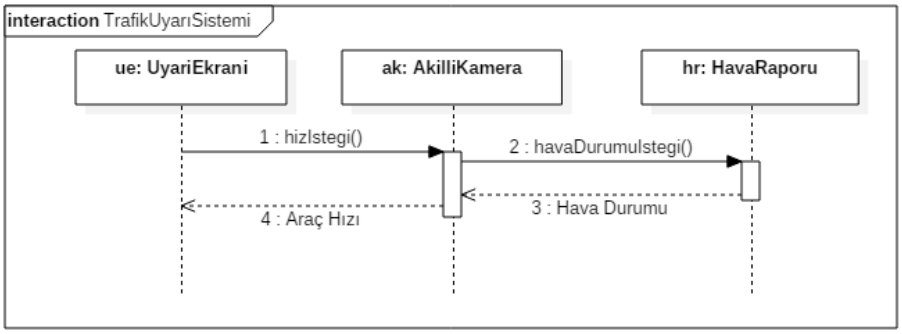


Şekil 1. Önerilen bileşen yapısı.

Önerilen mimariyi sunmak için gerçek zamanlı çalışan bir sistem olarak düşünülen Trafik Uyarı Sistemi (TUS) kullanılacaktır. Bu örnek, yer kısıtı açısından büyükçe tutulmamış, temel mekanizmanın çalışmasını aktaracak boyutta verilmiştir. Ayrıca, önerilen yapılar, mesajların bağlayıcılar tarafından tekrarlanması ve alt düzeyde mesaj sayısını bir iki misline çıkarması açısından katı gerçek zamanlı sistemler için verimsiz olabilecektir. Bu örnekteki düzeyi ile gerçek zamanlı kullanım için bir sorun görünmemektedir. Ayrıca, bağlayıcı yapıları yalnızca modelleme aşamasında kullanılacaksa ve model güdümlü (model driven) yöntemlerle serbestçe kod üretilebilecekse, gerçek zamanlı sistemler için böyle bir uyarlama da düşünülebilir.

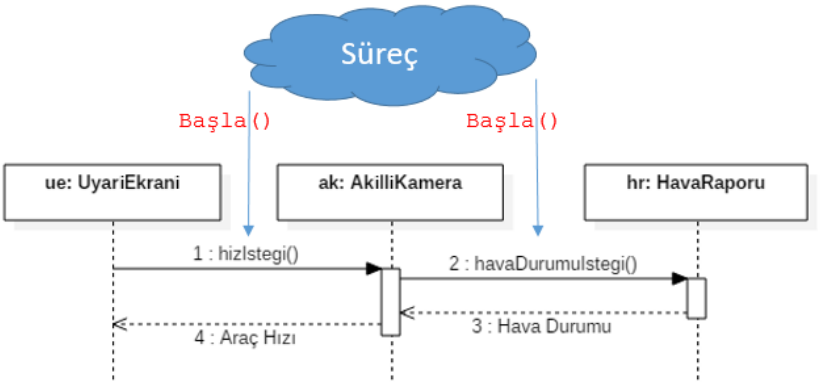
UML modellerinde içerilen kapı (port) yapılarını bu şekilde model düzeyinde içerip kod düzeyinde bunları kaldırıp mesaj sayısını azaltan çalışmalardan [17] bu yönde uyarlama yapılabilir. TUS'ta sürücülere anlık bilgilendirme sağlamak amacı ile kullanılan "UyariEkrani", hava koşullarına göre araçların hız/takip mesafesini termal-fotoğrafik yöntem veya radar ile hesaplamak amacıyla kullanılan "AkilliKamera" ve anlık hava durumu bilgisi sunmaya yarayan "HavaRaporu" bileşenleri bulunmaktadır.

Sistemin gerçekleştirimi nesne yönelimli programlama kullanılarak yapılmıştır. Bileşenlerin birbiri ile olan iletişimini göstermek amacıyla bileşenler birer nesne olarak ele alınarak Şekil 2'deki akış diyagramı verilmiştir.



Şekil 2. Geleneksel yöntemlerle bileşenler arası mesajlaşma.

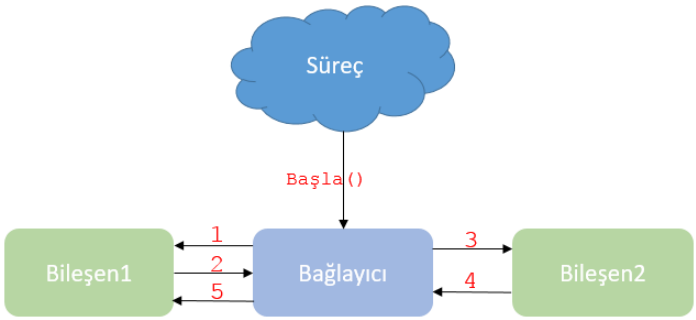
Şekil 2'de verilen mesajların çalışacağı göreceli sıranın belli olmasına rağmen hangi mesajların tam olarak ne zaman çalışacağının bilinmesi mümkün değildir. Bu durum özellikle gerçek zamanlı çalışan sistemlerde sistemin anlaşılabilirliği ve güvenirliği konusunda sorunlara sebep olabilmektedir. Bileşenler arasındaki etkileşimlerin merkezi bir yapı olarak bir süreç modeli üzerinden sağlanması ile kontrolü daha rahat ve güvenilir bir sistem elde edilebilecektir. İlgili sistemin merkezi bir yapı üzerinden kontrol edilen karşılığı Şekil 3'te gösterilmiştir.



Şekil 3. Önerilen sistemde süreç modeli üzerinden etkileşimin tetiklenmesi.

Şekil 3'te gösterildiği üzere önerilen yapıda iletişim, süreç modeli tarafından "Başla()" komutu ile tetiklenmektedir ve bu sayede hangi mesajın tam olarak ne zaman gönderileceği süreç tarafından kontrol edilir duruma gelmiştir. Bu durum Şekil 2'de verilen, mesajların zamanlarının tam olarak bilinemediği sisteme göre önemli avantajlar sağlamaktadır.

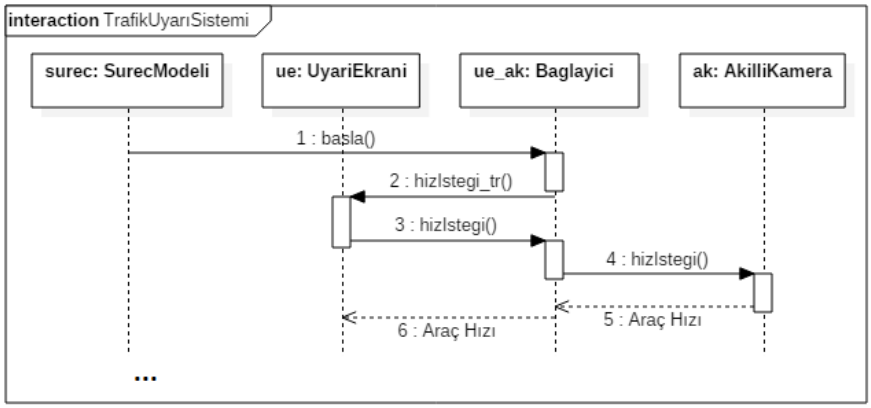
Süreç modeli tarafından bileşenler arası etkileşimi başlatmak üzere kullanılan "Başla()" metodu ile etkileşimde bulunması istenen iki bileşen arasındaki bağlayıcı tetiklenir. Bu tetiklenme ve akabinde gerçekleşen mesajlaşmalar Şekil 4'te gösterilmiştir. Bu süreç aşağıda sıralandırılmış mesajlar şeklinde özetlenmektedir:



Şekil 4. Önerilen sistemde süreç modeli üzerinden tetiklenen bileşenler arası mesajlaşma.

1. Bağlayıcı tarafından ilgili isteyici bileşen (requester) üzerinde bulunan aynı isimdeki metodu tetikleyen bir tetikleyici metot çağrılır.
2. İsteyici bileşendeki ilgili isteyici, isteğini bağlayıcıya iletir.
3. Bağlayıcı ilgili yanıtlayıcı bileşen (responder) üzerindeki yanıtlayıcı metodu çağırır.
4. Talebi alan yanıtlayıcı bileşen (responder) cevabı bağlayıcıya yanıtlayıcı metodu üzerinden iletir ve
5. bağlayıcı da cevabı ilgili bileşene göndererek etkileşimi sonlandırır.

Örnek olarak verilen TUS üzerinden, süreç modeli tarafından tetiklenen ve bileşenlerin bağlayıcılar aracılığı ile haberleşmesini gösteren akış diyagramı Şekil 5'te verilmiştir. Sistemdeki etkileşimleri gösterebilmek amacı ile süreç modeli, bileşenler ve bağlayıcılar birer nesne olarak ele alınmıştır. Birinci mesajda süreç modeli üzerinden "Başla()" metodu ile ilgili bağlayıcı tetiklenmektedir. İkinci mesajda bağlayıcı ilgili bileşenin tetikleyici metodunu (hizIstegi_tr()) çağırmakta, ardından tetikleyici metodu uyarılan bileşen (UyariEkrani) ilgili isteyici metodu ile bağlayıcı üzerinden istekte bulunmaktadır. Akabinde gerçekleşen mesajlaşmalar sırası ile verilmiştir.



Şekil 5. Süreç modeli üzerinden tetiklenen bileşenler arası mesajlaşmayı gösteren akış diyagramı.

4 İzlenimler

Çalışmayı özendiren fikirler, değişik uygulama alanlarındaki zorlukları yenmek üzere bileşen teknolojilerinin kullanımını kolaylaştırmak için değişkenlik yönetimi ve bağlayıcı kullanımı değerlendirildiğinde ortaya çıkmıştır. Bu uygulamalara örnek olarak nesnelerin interneti (IoT) sahasındaki çeşitlilik [18] verilebilir. Bu zorluklar ele alındığında ayrıca tümleştirme ile birlikte sıralama denetiminin karmaşıklığı göze çarpmaktadır. Mesajlar arasındaki sıralandırmanın denetim kabiliyetinin, gerekli soyutlamaların ima edilerek değil, doğrudan tanımlanarak geliştiricilere sunulması yönüne gidilmiştir. Böylece toplam zorluğu azaltıcı yönde bir adım atılmıştır. Sıralandırma tasarımı için öncelikle, uygulamanın (ya da bir kısmının) merkezi bir kontrol ile mi yoksa dağıtık bir algoritma tasarımı ile mi gerçekleştirileceği seçimi yapılmalıdır.

Bağımsız karakteri olan bileşenler ile bir tümleştirme yapıldığında ve merkezi bir uygulamanın da varlığı söz konusu olduğunda, temel "kontrol geçişi" yapıları olarak yoklama ya da kesme (polling/interrupt) mekanizmaları değerlendirilmelidir. Bu gibi mekanizmalar ve merkezi/dağıtık gibi üst düzey yaklaşımlar bir arada değerlendirildiğinde, birçok bileşimi gündeme getirmektedir. Temelde, bir sistem geliştirme ortamı farklı kullanımlara açık kapı bırakmalıdır, ancak özellikle bazı çözüm kategorilerinde daha verimli olmak üzere bir yönelimi de destekleyebilir. Bu çalışmada da değişik sıra denetimi mekanizmaları engellenmemekte ancak merkezi bir denetim ve yoklama mekanizmasının öne çıkarılacağı çözüm kategorileri hedeflenmektedir.

Çalışmada bağlayıcının üstlendiği görevleri yapabilmesi için aralarında bağlantı kurduğu bileşenlerin yerleri ve hangi metotlarını kullanarak iletişim sağlayacağını bilmesi gerekmektedir. Bunlara imkan tanıyan kod parçaları ilgilerin ayrılması prensibine uygun olarak bağlayıcının içine yerleştirilmiştir. Bunun alternatifi olarak bu iletişim süreç tarafından yapılacak olursa kod parçalarının sürecin içinde yer alması

gerekecektir. Bu da çalışan bir sistemi mümkün olduğu kadar kolay geliştirmek için basit tutulmaya çalışılan süreç modellerini daha karmaşık hale getirecektir.

Bileşenler arası iletişim, arakatman (middleware) kullanarak da sağlanabilir. Bileşenler arası veri aktarımının yanı sıra veri üzerinde yapılacak tür dönüşümü gibi işlemlerle bileşenlerin eşgüdümlü çalışması ve veri güvenliği gibi işlerin de arakatman tarafından sağlanması beklenebilir. Arakatman içinde sağlanan bu işlevler genellikle kullanıcı müdahalesine kolaylıkla imkan vermeyen ya da arakatman yapısını tam anlamıyla öğrenerek değişikliğe el veren niteliktedir. Bu nedenlerle ve zaten eşgüdüm denetimini öne çıkarmak için bağlayıcılar önerildiğinden bileşenler arasındaki eşgüdüm ve iletilen veri üzerinde yapılacak işlemler de arakatmanın dışına çekilerek, bağlayıcılarla yapılmıştır. Bir arakatman ile çalışıldığında bile, bileşenlerin arakatman ile uyumlu çalışması amacıyla yazılması gerekecek kodlar için de en uygun yapı bağlayıcılardır.

Bu makalede önerilen yöntemin bir zorluğu, mevcut bileşen modellerinde değişiklik önermesidir. Bu nedenle, kritik ihtiyaca neden olacak uygulama alanları oluştuğunda bu tür bileşenlerin yaygın kullanımı ancak söz konusu olabilecektir. Dolayısı ile, yapılmakta olan çalışmalarda denetim kolaylığı ile birlikte bileşen modelinde gerekecek yenilikler, bir ödünleşme konusu olarak ele alınmaktadır. Genelde diğer yaklaşım kategorileri için mevcut bileşen modellerini olduğu gibi kullanan öneriler öne çıkmaktadır. Ancak bu ve gelecek çalışmaların ortak yönü, yeni bağlayıcı tanımlarıdır. Yaygın kullanımda bileşenler gibi protokollere ve modellere sahip olmayan bağlayıcılar için bu yönlerde tanımlamalar yapılmaktadır.

5 İlgili Çalışmalar

Bağlayıcılara sözü geçen kabiliyetlerin yüklenmesi yönünde çalışmalar daha önceden de bulunmaktaydı. Ancak değişkenlik yönetiminin de eklenmesi ve eşgüdümün pratikleştirilmesi katkıları ile yazarların çalışmaları, bağlayıcılara bileşenlerin elde ettiği düzeyde bir yeniden kullanılabilirlik kazandırılmasına doğru gitmektedir. Bu bölümde, bu tür bir hedefle doğrudan ilişkili olarak daha çok yazarların çalışmalarına atıfta bulunulmaktadır. Değişkenlik yönetimi bileşen yönelimli modellemeye uyarlanmış [19] daha sonra bağlayıcılara da taşınmış ve bu unsurlara eşgüdüm boyutunun pratik araçları olma yönünde bir yapı kazandırılmıştır [20]. Bu öncü çalışmalar sonucunda tümleştirme kolaylaşmışsa da karşılığında bileşen modellerine müdahale edilmesi gereği ödünü verilmiştir. Daha yaygın kullanıma açık olunması gözetilerek mevcut bileşen yapıları ile uyumlu çalışacak mimarilerin öne çıktığı bir çalışma ise [21] yakında sunulacaktır.

6 Sonuç

Merkezi bir denetim yaklaşımı için bileşen tümleştirerek sistem geliştirme işlemini desteklemek üzere yeni bağlayıcı yapıları ve bileşen yapılarında değişiklikler tanımlanmıştır. Hedefi gerçekleştirmek üzere özellikle örnek çalışmalarda sunulan yeniliklerin geçerli olabileceği gözlemlenmiştir. Burada sunulan küçük örneğe ek

olarak başka çalışmalar da yapılmıştır. Yapılan çalışmalar, gerçek bileşenlerle değil, onların benzetimi olarak Java kodu ile yazılmış varlıklarla yapılmıştır. Çalışmanın ana zorluğu, mevcut bileşenlerin doğrudan kullanılamayıp bileşen protokollerinde değişiklik gerektiriyor olmasıdır. Verim gibi kısıtların zorlanmadığı ve merkezi denetimin önemli olduğu (örneğin bazı IoT tabanlı uygulamalar) durumlarda önerilen modelin yararlı olacağı tahmin edilmektedir. Gelecek çalışmalarda ise yaklaşımın yaygın olarak kullanılan diğer bileşen modellerine uyarlanması ve farklı arakatmanlarda kullanılarak testlerin yapılması hedeflenmektedir.

Kaynaklar

1. Dogru, A. H., Tanik, M. M.: A process model for component-oriented software engineering. IEEE software, 20(2), 34–41 (2003).
2. 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).
3. Hallsteinsen, S., Hinchey, M., Park, S., Schmid, K.: Dynamic software product lines. Computer, 41(4) (2008).
4. 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)
5. Taylor, R. N., Medvidovic, N., Dashofy, E. M.: Software architecture: foundations, theory, and practice. Wiley Publishing (2009).
6. Mehta, N. R., Medvidovic, N., Phadke S.: Towards a taxonomy of software connectors. In: Proceedings of the 22nd international conference on Software engineering, pages 178–187. ACM (2000).
7. Pohl, K., Böckle, G., van der Linden, F. J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer (2005).
8. Sinnema, M., Deelstra, S., Nijhuis, J., Bosch J,: Covamof: A framework for modeling variability in software product families. In: RobertL. Nord, editor, Software Product Lines, volume 3154 of Lecture Notes in Computer Science, pages 197–213. Springer Berlin Heidelberg (2004).
9. 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).
10. Kaya, M. Ç., Suloglu, S., Dogru, A. H.: Variability modeling in component oriented software engineering. In: Proceedings of the 2014 Society for Design and Process Science, (2014).
11. Dogru, A.H.: Component oriented software engineering modeling language: COSEML. Computer Engineering Department, Middle East Technical University, Turkey, TR 99-3 : (1999).
12. Cetinkaya, A., Kaya, M. Ç., Dogru, A. H.: Enhancing xcoseml with connector variability for component oriented development. In: Proceedings of the 2016 Society for Design and Process Science, Orlando, USA (2016).
13. Kaya M. C., Saeedi Nikoo M., Suloglu S., Dogru A. H.: Towards Verification of Component Compositions Incorporating Variability, In: Proc SDPS the 20th International Conference on Transformative Science and Engineering, Business and Social Innovation, Fort Worth Texas USA (2015).

14. Classen, A., Cordy, M., Heymans, P., Legay, A., Schobbens P. Y.: Model checking software product lines with SNIP, International Journal on Software Tools for Technology Transfer (STTT) (2012).
15. Classen, A., Cordy, M., Schobbens, P. Y., Heymans, P., Legay, A., Raskin, J. F.: Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and Their Application to LTL Model Checking, IEEE Transactions on Software Engineering, (2013).
16. The Object Management Group, UML 2.0, http://www.omg.org/spec/UML/2.0/, last accessed 2017/06/20.
17. Kocataş, A. T., Can, M., Doğru, A. H.: Lightweight Realization of UML Ports for Safety-Critical Real-Time Embedded Software, In: 4th International Conference on Model Driven Engineering and Software Development (Modelsward 2016), Rome, Italy (2016).
18. Kaya, M. C., Saeedi Nikoo, M., Suloglu, S., Tekinerdogan, B., Dogru, A. H.: Managing Heterogeneous Communication Challenges in Internet of Things using Connector Variability, Connected Environments for the IoT: Challenges and Solutions, Springer, Yayınlanmak üzere kabul edilmiştir, (2017).
19. Kaya, M. Ç.: Modeling Variability in Component Oriented Software Engineering. MSc Thesis, Middle East Techical University, (2015).
20. Çetinkaya, A.: Variable Connectors in Component Oriented Development. MSc Thesis, Middle East Techical University, (2017).
21. Kaya, M. Ç., Cetinkaya, A., Dogru A. H.: Composition Capability of Component-Oriented Development, In: 5th International Symposium on Innovative Technologies in Engineering and Science (ISITES), Sözlü sunum için kabul edilmiştir, Baku-Azerbaijan (2017).