Bileşen Yönelimli
Yazılım Geliştirme İçin Süreç Modeli

Vedat BAYAR

Havelsan A.Ş. Eskişehir yolu 7.km ANKARA

vbayar@havelsan.com.tr

Özet

Bileşen Yönelimli Yazılım Mühendisliği (BYYM) yaklaşımı için bir süreç modeli geliştirildi. Geliştirilen bu süreç modeli, bileşen tabanlı yazılım geliştirme için bir yol gösterici niteliği taşımaktadır. Süreç, sistem spesifikasyonlarının ayrıştırımı ile başlayıp, ayrıştırılan sistemin yazılım bileşenlerine dönüştürülmesi ile devam edip, bileşenlerin entegre edilip hedeflenen sistemin oluşturulması ile sonuçlanmaktadır.

Abstract

A process model is developed for Component Oriented Software Engineering (COSE) paradigm. The model presents a guide for component-based software development. The process starts with the decomposition of a system specification and continues with represention of the decomposed system parts with software components and ends up with the system by the integration of these software components.

1. Giriş

Yazılım sistemlerinin zaman içerisinde daha büyük ve daha kompleks hale gelmeleri ile yazılım sistemlerinin geliştirilmesinde yazılım bileşen teknolojisinin önemi de artmıştır. Yazılım bileşenlerinin neredeyse yazılım geliştirme sürecinin başından beri biliniyor olmasına rağmen pratik kullanımı ve gelişimi zaman içerisinde olmuştur [1].

Yazılım teknolojisinin başlangıcından itibaren, yazılım geliştiren kuruluşlar, tasarım tekniklerini iyileştirerek, daha uygun süreç modelleri ve metodolojileri ve mevcut sistemlerin kullanılabilir parçacıklarını kullanarak, yazılım geliştirme süreçlerini iyileştirme çabası içerisinde olmuşlardır. Günümüzde, yazılım sistemlerinin, mevcut yazılım bileşenlerinin uygun bir yazılım mühendisliği yaklaşımı ile entegre edilerek geliştirilmesi ihtiyacı, sıfırdan geliştirilmesine göre kaçınılmaz bir hal almıştır.Yazılımın yeniden kullanımı (reusability) sadece yazılım geliştirme sürecini hızlandırmakla kalmaz aynı zamanda yazılım kalitesini ve üretkenliği de artırır [2]. Fakat yazılımın yeniden kullanımı çok kolay değildir. Yeniden kullanılabilecek bileşenlerin çok iyi tanımlanmış ve bağımsız bir şekilde yaygınlaştırılabilir (deploy) olmaları gerekir. Yazılım bileşenleri ile ilgili çok farklı tanımlamalar mevcuttur, fakat tüm bu tanımlamaların üzerinde mutabakata vardığı nokta, yazılım bileşenlerinin yeniden kullanılabilir olması noktasıdır. Yazılım bileşeni, bir standarda uygun, yeniden kullanılabilir ve tanımlı bir mimari çerçevesinde geliştirilmiş, işlevsel ve değişitirilebilir bir

yazılım parçasıdır. Önceden geliştirilmiş, test edilmiş ve onaylanmış yazılım bileşenlerinin yazılım geliştirmede kullanılması geliştirilecek sistemin kalitesini ve güvenirliliğini artırır.

Bileşen tabanlı yazılım geliştirme, geliştirme sürecini, sıfırdan kod yazma işleminden bileşenlerin entegrasyonu işlemine dönüştürmüştür. Hedeflenen sistemin spesifikasyonu belirlenir, bu spesifikasyonlara göre sistem alt bileşenlerine ayrıştırılır. Bu işlemden sonra uygun bileşenlerin araştırılması ve uyarlanması işlemi gerçekleştirilir. İhtiyaç duyulan bileşenlerin sağlanmasından sonra bileşenler entegre edilerek hedeflenen sistem oluşturulur [3].

Yapısal (structural) bir bileşen yönelimli yazılım geliştirme yapabilmek için bir süreç modeli geliştirildi. Bu süreç modeli BYYM Süreç Modeli olarak isimlendirildi [4]. BYYM Süreç Modeli adım-adım bir sistemin hiyerarşik olarak nasıl yapıtaşlarına ayrıştırılacağını, bu yapıtaşlarının bileşenlere nasıl denk düşürüleceğini ve bu bileşenlerin nasıl entegre edilip hedeflenen sistemin oluşturulacağını tanımlar.

2. Bileşen Yönelimli Yazılım Geliştirme Süreç Modeli

Geleneksel metodolojiler büyük oranda sistemin fonksiyonel ayrıştırımı üzerinde yoğunlaşmışlardır. Nesne yönelimli metodolojilerde ise sistemin 'veri' boyutu ön plana çıkar. Öte yandan bileşen yönelimli yazılım mühendisliği, BYYM, yapı-yönelimli bir metodolojiyi benimserdir. BYYM Süreç Modeli, sistemin yapı taşlarını ortaya koymak için sistemin yukarıdan-aşağı bir ayrıştırımı ile başlar. Süreç ilerledikçe modüller arasındaki arayüzler tanımlanmış olur. Modüllerin bileşenlerle eşleştirilebileceği düşünüldüğü bir sevieyede geçici bir aşağıdan-yukarıya yaklaşıma geçilir. BYYM süreç modeli dört ana aşamadan ve sistem test evresinden oluşur:

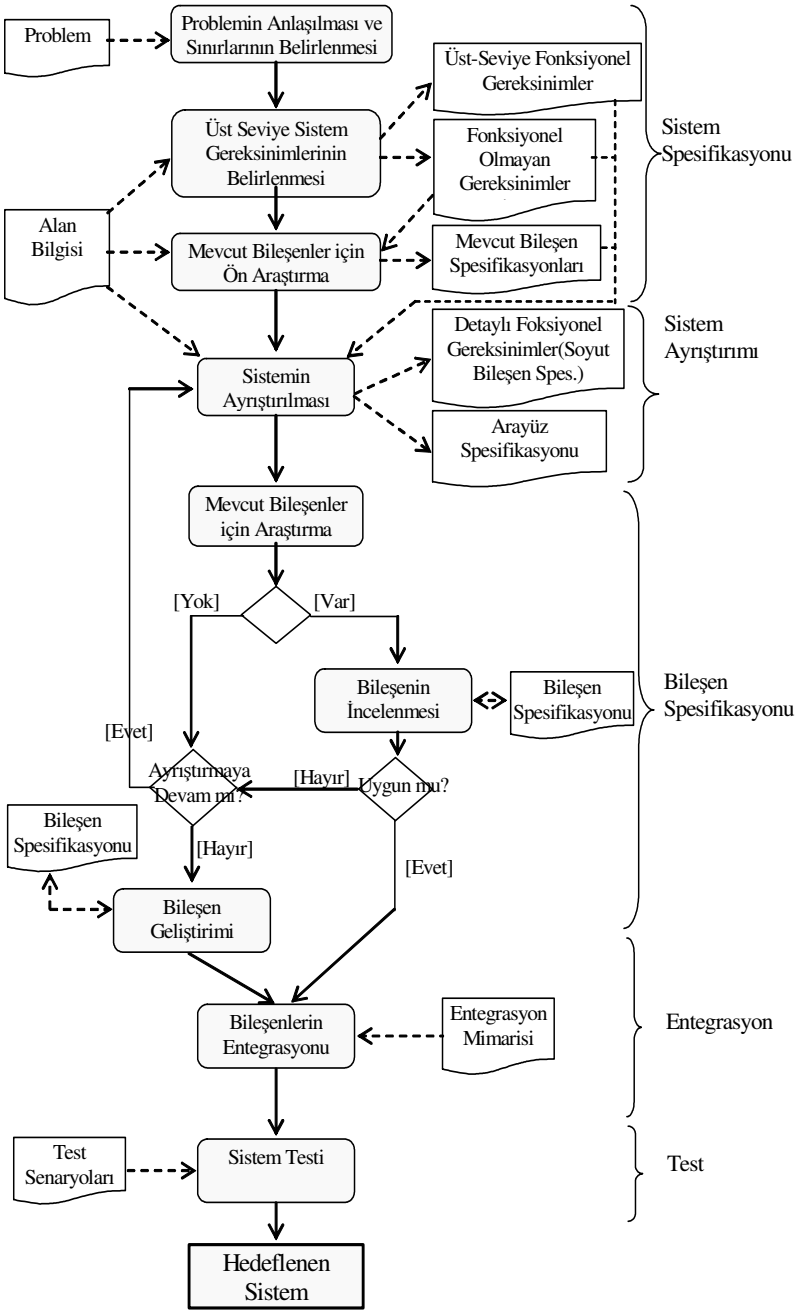
- Sistem spesifikasyonu,
- Sistem ayrıştırımı,
- Bileşen spesifikasyonu, ve
- Entegrasyon.

BYYM süreç modelinin süreç akışı Şekil 1'de gösterildiği gibidir. Oval dikdörtgenler belirtilen evredeki üst seviye süreçleri gösterir. 'Ok' yönleri süreç ve veri akışı yönlerini gösterir. Süreçlere kesikli ok çizgileri ile iliştirilmiş dokümanlar bilgi akışını gösterir. Eşkenar dörtgenler karar durumlarını gösterir. Süreç modelinde kullanılan ve oluşturulan dokümanların özet tanımları aşağıdaki gibidir:

Problem dokümanı: Hedeflenen sistemin tanımını içerir, müşteri tarafından sağlanır.
Alan Bilgisi: Alan bilgisini içeren iyi hazırlanmış bir doküman ve/veya alan uzmanları tarafından sağlanan alan bilgisi.
Üst-Seviye fonksiyonel gereksinimler dokümanı: Metin veya use-case model olarak sistemin fonksiyonel gereksinimleri dokümanı.
Fonksiyonel olmayan gereksinimler dokümanı: Fonksiyonel olmayan fakat sistemin çalışmasını etkileyen kısıtlar; güvenlik, performans, vb.
Mevcut bileşen spesifikasyonları: Bileşenler ve bileşenlerin arayüzleri ile ilgi dokümante edilmiş bilgiler.
Detaylı fonksiyonel gereksinimler (Soyut Bileşen Spesifikasyonları): Sistem ayrıştırımı safhasında oluşturulan gereksinimler dokümanı.
Bileşen Spesifikasyonları: Tanımlanmış bileşen arayüzleri dokümanı. Bileşenler arasındaki bağlantı mekanizmalarını açıklar.
Entegrasyon Mimarisi: Bileşenlerin içinde hayat bulacağı birbirleri ile iletişim kuracağı bileşen mimarisi spesifikasyonu.

Test Senaryoları: Sistem fonksiyonlarının faaliyet serileri bazında açıklandığı doküman.

BYYM Süreç Modeli sistem spesifikasyonu fazı ile başlar. Geliştirilecek olan sistem belirlenir ve anlaşılır, sistemin üst seviye gereksinimleri belirlenir, sistemin gerçekleştiriminde kullanılabilecek bileşenler için ön araştırma yapılır. Bu faza 'problem' ve 'alan bilgisi' bilgileri girdi olur. Sistemin üst seviye fonksiyonel ve fonksiyonel olmayan gereksinimleri ve alan ile ilgili mevcut bileşen spesifikasyonları dokümanları bu fazın çıktılarını oluşturur.



Şekil 1. Süreç akışı

Sistemin analizi ve alt parçalarına ayrıştırımı, sistem ayrıştırımı fazında gerçekleştirilir. Bu fazda fonksiyonel gereksinimler detaylandırılır ve gerekli bileşenler ve bu bileşenlerin spesifikasyonları ortaya konur.

Sistemi gerçekleyecek bileşenlerin belirlenmesi ve geliştirilmesi işlemleri bileşen spesifikasyonları fazında belirlenir. Sistem ayrıştırımı ve bileşen spesifikasyonu fazları birbirinden tamamıyla ayrık fazlar değildirler. Sistem ayrıştırımı belli bir olgunluğa geldiğinde bileşen spesifikasyonu fazı başlar. Bileşen spesifikasyonu fazı, mevcut bileşenlerin araştırılması ile başlar ve bulunan bileşenlerin değerlendirilmesi ile devam eder. Eğer ayrıştırım seviyesi, gerekli bileşenlerin bulunması ve değerlendirilmesi için yeterli değil ise daha anlamlı bir ayrıştırım olmayacağı görülünceye kadar sistem ayrıştırılmaya devam edilir. Sistem ayrıştırımı ve bileşen spesifikasyonu fazlarının çıktıları, bileşenler ve bu bileşenlerin spesifikasyonlarıdır.

BYYM Süreç Modeli, bileşenlerin entegrasyonu ve tüm sistemin test edilmesi ile devam eder. Entegrasyon mimari dokümanı entegrasyon fazına girdi teşkil eder. Test senaryoları da test fazına girdi olurlar.

2.1 Sistem Spesifikasyonu

Yazılım geliştirme perspektifi altında, geliştirilecek sistemin belirlendiği ve ortaya konduğu aşama sistem spesifikasyonu aşamasıdır. Bu aşama sonucunda ortaya çıkan çıktılar müşteri ile sistemi geliştirecek üretici arasında kontrat teşkil eder. Bu fazda aşağıdaki faaliyetler gerkeleştirilir:

- Problemin, yazılım konseptine uygun olarak analiz edilerek modellenmesi,
- Hedeflenen sistemin sınırlarının ve içeriğinin belirlenmesi,
- Üst-seviye sistem fonkiyonalitesinin belirlenmesi,
- Gerçekleştirilecek sistemle ilgili olabilecek mevcut aday bileşenler için ön araştırmanın yapılması.

Bu fazın en önemli zorluğu problemden analiz modele geçiştir. Yazılım geliştiriciler ile müşteri arasındaki yanlış anlaşılmalar kaçınılmazdır. Müşterilerin büyük bir bölümü yazılım konseptlerine yabancı oldukları için ihtiyaçlarını tam olarak ifade edemeyebilirler ya da yazılım geliştiricilerin problemi yanlış yorumlamalarına neden olabilirler. Bu yanlış anlaşılmaları ortadan kaldıracak ya da en aza indirecek bir yönteme ihitiyaç vardır. Use case modelleme bu amaç için kullanılabilecek en etkili ve en basit yöntem olarak görülmektedir [5]. Basit ve etkili bir yöntem olduğu için geliştiriciler ile müşterinin ortak bir dili konuşmasına olanak sağlar.

Sistemin genel bir görünümü, sistem sınırları ve kullanıcıları, müşteri ile birlikte çalışılarak use case modelleme yöntemi ile modellenir. Üst-seviye sistem fonksiyonaliteleri, use case ve sistem kullanıcıları (actor) ile ifade edilir. Fonksiyonel olmayan gereksinimler, use case modellerine ek olarak belirlenir ve dokümante edilir. Sistemin üst-seviye gereksinimleri ve sınırları belirlendikten sonra aday bileşenlerin bulunması için ön bir araştırma yapılır. Bileşenlerin etkili kullanımı için bileşenlerin bulunması, onları geliştirmekten daha kolay olmalıdır. Uygun bileşenlerin bulunması demek bire bir tüm ihtiyaçları karşılayacak bileşenlerin bulunması anlamına gelmemektedir. Sistem detaylı bir şekilde analiz edilmediği için bu aşamada aday bileşenlerin belirlenmesi yeterli olacaktır. Yapılan bu ön araştırma, geliştiricilerin alan ile ilgili bileşenler hakkında fikir sahibi olmalarını sağlayacaktır. Etkili bir bileşen yönelimli geliştirim için bir bileşen kütüphanesinin olması faydalı olur. Aday bileşenlerin seçiminde göz önünde bulundurulacak kriter, bileşenlerin kavramsal olarak

sistemden beklenen servisleri sağlayabilecek nitelikte olmalarıdır. Bu çalışmanın sonucunda aday bileşenler ve bu bileşenlerin sepsifikasyonları belirlenmiş ve anlaşılmış olur.

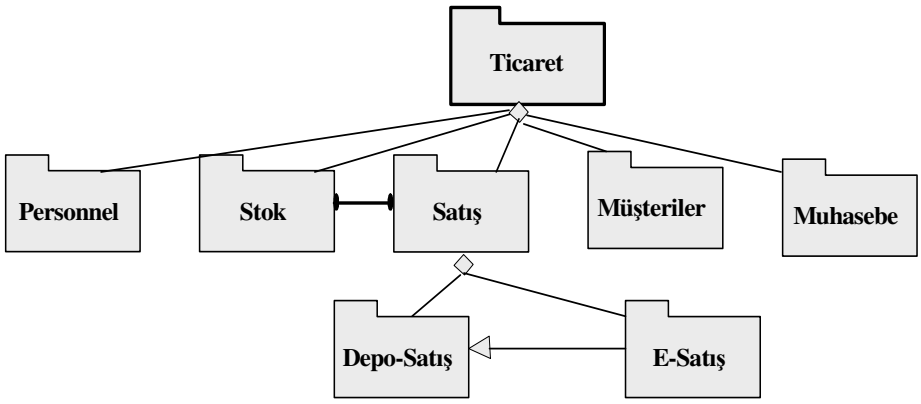
2.2 Sistem Ayrıştırımı

Sistemin ayrıştırımı fazı, alan bilgisine sahip uzmanlar ile geliştiricilerin ortak çalışması ile gerçekleştirilir. Alan uzmanları, geliştirilecek sistemin alan bilgisine sahip insanlardır. Bu insanlar geliştiricilere alan bilgilerini aktararak, geliştiricilerin sistemi alt parçalara ayrıştırmalarında ve detaylı gereksinimleri ortaya koymalarında yardımcı olurlar. Sistem ayrıştırımı safhasının amacı, sistemi yeniden kullanılabilir bileşenlere denk düşecek sekilde daha atomik alt parçacıklara bölmektir. Bu fazda aşağıdaki faaliyetler gerçekleştirilir:

- Sistemin paketlere, fonksiyonlara ve veri soyutlamalarına ayrıştırılması[1],
- Paketler arasındaki arayüzlerin belirlenmesi,
- Paket elemanlarının birbirleri ile olan iletişimlerinin belirlenmesi,
- Ayrıştırımın alan ile ilgili senaryolara göre değerlendirilmesi.

Sistem ayrıştırım fazında, sistem, soyut bir paket olarak ele alınır ve ayrıştırım bu paketin alt paketlere bölünmesi ile başlar. Ayrıştırım yukarıdan-aşağı yapılır. Sistem, kabiliyetlerine göre üst-seviye paketlere bölünür. Sistemin ana kabiliyetleri soyut olarak bu üst-seviye paketlerle ifade edilir. Üst-seviye sistem kabiliyetleri, fonksiyonlara veya alt seviye paketlere bölünür. Bu işlem anlamlı bir bölünmenin yapılamayacağı seviyeye kadar devam eder. Şekil 2, personel, stok, satış, müşteriler ve muhasebeden oluşmuş küçük ölçekli bir işin ayrıştırımını göstermektedir. E-Satış, satış işlemi altında olup Depo-Satışın özelleşmiş bir halidir.

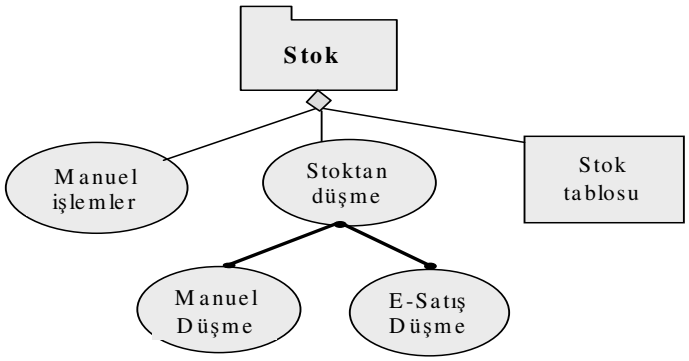
Sistem spesifikasyonu aşamasında, modellenen use caseler paketlere dönüştürülür. Her bir use case bir veya birden fazla pakete karşı gelebilir. Paketler arasındaki arayüzler, use caselerin birbirleri ile olan etkileşimlerinden çıkarılabilir. Paketler arasındaki arayüzlerin hepsi, use case modellerinde açıkça görülmeyebilir, bu tür arayüzler bu aşamada tanımlanabilir. Arayüzler paketlerin birbirleri ile iletişimlerinde uyulması gereken kontratları belirtirler. Her bir paket bu kontrata uymak durumundadır. Bu şekilde herbir paket paralel bir şekilde daha alt paket, fonksiyon veya veri soyutlamalarına ayrıştırılır. Her bir ayrıştırım seviyesinden sonra paketler arasındaki arayüzler gözden geçirilir ve mutabakata varılır.



Şekil 2. Sistemin ana kabiliyetleri

[1] Makale genelinde kullanılan kavram ve sembollerin detaylı açıklamaları 4 nolu kaynakçada verilmiştir.

Paket detayları fonksiyon ve veri soyutlamaları ile ifade edilir. Başlangıçta her bir use case fonksiyon soyutlamalarına denk düşürülür. Use case diagramlarındaki notlar ve ek bilgiler içeriklerine göre veri veya fonksiyon soyutlamalarına denk gelebilir. Veri yapıları gibi ek veri ve fonksiyon soyutlamaları tanımlanabilir. Paketlerin fonksiyon ve veri soyutlamalarına ayrıştırılması ile paketin iç işleyişi ve arayüzleri açıkça anlaşılır duruma getirilir. Şekil 3, Stok paketinin fonksiyon ve veri soyutlamalarına ayrıştırımını göstermektedir.



Şekil 3. Stok paketinin fonksiyon ve veri elemanlarına ayrıştırımı.

Fonksiyon ayrıştırmaları sistem seviyesi fonksiyonaliteye denk düşmektedir. Eğer bir soyutlama paket olacak kadar kompleks bir soyutlama değil ise fonksiyon soyutlaması ile gösterilir. Fonksiyonlar, veri erişimini de kullanarak veri erişiminden fazlasını yerine getirirler. Şekil 3'te görüldüğü gibi fonksiyon soyutlamaları alt fonksiyon soyutlamalarına sahip olabilirler. Veri erişim metodları veri soyutlamalarına dahil edilirler. Kendilerine has metodları ve alt veri soyutlamalarına sahip olabilirler. Eğer bir soyutlama veri veya depolama ile ilgili ise onun veri soyutlaması ile ifade edilmesi anlamlıdır.

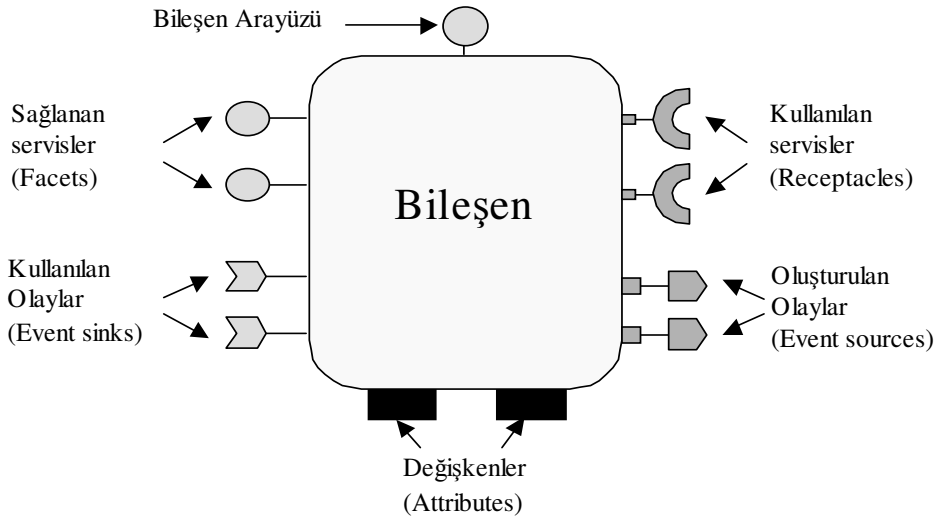
Ayrıştırım işlemi tamamlanınca, parça-bütün hiyerarşisine göre bir gözden geçirme işlemi yapılır. Paketler arasındaki mesaj tarfiğini azaltmak ve paketlerin birbirlerine olan bağımlılıklarını en aza indirmek için eğer gerekiyorsa bazı foksiyonlar ilgili paketlere taşınabilir. Ortak yeni fonksiyon ve veri soyutlamaları yaratılabilir. Daha sonra mesajlaşma mekanizması, belirlenmiş senaryolara göre gözden geçirilir. Bu senaryolar sistemin işleyişini gösterir. Paketler arası iletişimi baz alan senaryolar paketlerin arayüzlerinin oturmasını sağlar. Paketin iç işleyişine yönelik senaryolar ise paketin ayrıştırımının ve paket elemanları arasındaki ilişkilerin değerlendirilmesinde kullanılır. Bu senaryolar UML in akış modelleri kullanılarak görsel olarak modellenebilir.

2.3 Bileşen Spesifikasyonu

Ayrıştırılmış sistemin gereksinimlerine uygun bileşen araştırması, bileşenlerin modifikasyonu ve geliştirilmesi işlemleri bu fazda yapılır. Mantıksal olarak ayrıştırılmış sistem fiziksel elemanlara yani bileşenlere bu aşamada denk düşürülür. Bu fazda aşağıdaki faaliyetler gerçekleştirilir:

- Ayrıştırılmış sistemin ihtiyaçlarına göre, bileşen araştırmasının yapılması,
- Bulunan bileşenlerin eğer gerekiyorsa değiştirilmesi,
- İstenen özeliklerdeki bileşen bulunamadıysa bilşenin geliştirilmesi.

Bileşen araştırmasına geçilmeden önce ihtiyaç duyulacak bileşenlerin arayüzlerinin ve spesifikasyonlarının ayrıştırılan sistemin ihtiyaçlarına göre belirlenmesi gerekir. Her bir paket, fonksiyon ve veri soyutlamaları aday bileşenlerdir. Bu soyutlamalar arasındaki ilişkiler bileşenlerin arayüzlerini belirler. Her bir paketin bir bileşen olarak ele alınması iyi bir başlangıç olabilir. Bir bileşen birden fazla bileşeni içerebilir. Eğer fonksiyon ve veri soyutlamaları bileşenlerin karakteristik özelliklerini taşıyorsa bunlarında paket içerisinde bileşen olarak ele alınmaları anlamlıdır. Soyut bir öğeden fiziksel bileşenlere geçiş için kesin adımlar tanımlayamıyoruz. Bu yetenek tasarımcıların deneyim ve etkinliğine bağlı olarak gelişecektir. Genelde tasarım fazla yönlendirilemeyen insana bağlı bir süreçtir. CORBA bileşen modeli yazılım bileşen mantığını en iyi şekilde ifade eden bir model olup bileşenlerin oluşturulmasında bu model referans alınacaktır [6]. CORBA bileşen modelinde bileşenin görsel olarak nasıl ifade edildiği Şekil 4'te gösterilmiştir[2]. Günümüzde web servisleri popüler olarak kullanılmaktadır. Soyut düzeyde bu tür servisler birer bileşen olarak değerlendirildiğinde önerdiğimiz metodoloji açısından önemli bir fark oluşmamaktadır.



Şekil 4. CORBA Bileşen Modeli (CORBA Component Model- CCM)

Bileşen olmaları anlamlı olmayan fonksiyon ve veri soyutlamaları oldukları gibi kalabilirler. Bu tür fonksiyonlar, bileşenlerin iç işleyişini oluştururlar, veri soyutlamaları ise veri yapılarını temsil edecek şekilde kalırlar. Şekil 5, Stok paketinin bileşen ve bileşen arayüzlerini gösterir.

Bu işlemler tamamlandığında taslak bir bileşen spesifikasyonu oluşturulur ve mevcut bileşenlerin araştırılması yapılır. Bulunan bileşenler detaylı bir şekilde değerlendirilir. Gerekiyorsa bulunan bileşenler değiştirilir fakat ideal olanı bileşenleri değitirmeden kullanabilmektir. Mevcut bileşenlerin değiştirilmesi işlemi değişik formasyonlar da yapılabilir:

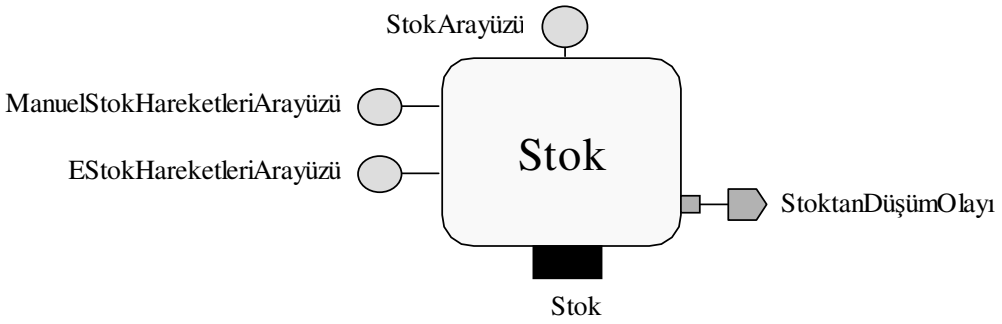
Yeni fonksiyonalite eklemek: Bileşen geliştirilirken sonradan ortaya çıkmış gereksinimleri yerine getirmiyor olabilir veya bu gereksinimleri gözardı etmiş olabilir. Eksik olan bu fonksiyonalitelerin eklenmesi bileşeni yeniden yazmaktan daha anlamlı olabilir.

Fonksiyonalitenin çıkarılması: Bileşenler gerekenden daha fazla fonksiyonalite içeriyor olabilir. Gerekmeyen bu tür fonksiyonalitelerin çıkarılması etkinlik için gerekli olabilir.

Genelleştirme: Bileşenler istenildiği ölçüde genel olmayabilir. Bu durumda bileşenin daha genel olacak şekilde değiştirilmesi gerekebilir.

[2] CORBA bileşen modeli ile ilgili detaylı bilgi 6 nolu kaynakçada verilmiştir.

Bileşen araştırması ve bulunan bileşenlerin değişitirilmesi sonucunda hala karşılanamayan sistem fonksiyonaliteleri varsa bu fonksiyonaliteleri yerine getirecek bileşenlerin belirlenmesi ve sıfırdan geliştirilmesi işlemi yerine getirilir.



Şekil 5. Stok bileşeni ve arayüzleri

2.4 Entegrasyon

Bu fazda, sistemi gerçekleyen bileşenlerin entegre edilme işlemi yerine getirilir. Bileşenlerin entegrasyonu seçilen bileşen mimarisine göre yapılır. Bileşen mimarisi, servis sağlayıcılar ile istemciler arasındaki iletişimi birbirlerinin detaylarını bilmek durumunda kalmadan sağlar. Nesne-yönelimli modellemelerde geniş çapta kullanılan UML akış modelleri bileşen yönelimli modellemeye de uyarlanabilir. Bu modellerdeki nesneler bileşenlerle ve nesne metod çağırmaları da bileşen arayüzleriyle değiştirilerek, bileşenlerin entegrasyonu işleminde kullanılabilir.

3. Sistem Testi

Bileşenler test edilip onaylandıkları için entegrasyonun ve sistemin bütün olarak testi bu aşamada yapılır. Bütün entegrasyon noktaları ayrı ayrı test edilir. Entegrasyon testinde iç işleyişi değilde bileşenlerin istenilen şekilde birbirleri ile entegre edilip edilmediği test edilir. Entegrasyon noktalarını içerecek şekilde oluşturulmuş entegrasyon senaryolarına göre entegrasyon testi yapılır. Sistemin bütün olarak hedeflenen fonksiyonaliteyi yerine getirip getirmediğini gösterecek olan senaryolar ile sistem testi gerçekleştirilir.

4. Sonuç

Mevcut bileşenlerin entegrasyonu ile yazılım geliştirme, sıfırdan kod yazımı ile yazılım geliştirmeye göre önemli bir değişimdir. Bu nedenle mevcut süreç modellerinden bileşen yönelimli yazılım geliştirme için yeni bir süreç modeline geçiş kaçınılmazdır. Bu makale ile henüz yeni olan BYYM süreç modeli tanıtıldı. BYYM süreç modeli bileşen yönelimli bir süreç modelidir. Bu model sistemi nerdeyse bağımsız alt parçalara bölerek yazılım geliştiriminin yönetilebilirliğini kolaylaştırır ve paralelliğini artırır.

BYYM süreç modelinin köşe taşı entegrasyon ile geliştirimdir. Modelin yapıtaşlarını bileşenler oluşturur. Model, sistemin nasıl daha atomik alt parçalara bölüneceğini ve bu alt parçaların yazılım bileşenleri ile nasıl ifade edileceğini belirtir. Dolayısı ile yazılım sektöründeki yazılım bileşenlerinin belli bir olgunluk seviyesine ulaşmaları ve kolay erişilebilir olmaları modelin etkinliğini artıracaktır.

Kaynakça

[1] Jon Hopkins, "Component Primer," *Communications of the ACM*, cilt 43, no. 10, sayfa. 27-30, Ekim. 2000.

[2] H. Mili, F. Mili ve A. Mili, "Reusing Software: Issues and Research Directions," *IEEE Trans. Software Eng.*, cilt 21, no. 6, sayfa. 528-562, Haziran 1995.

[3] Ali H. Dogru, Leon K. Jololian, Franz Kurfess ve Murat M. Tanık, "Component based Technology for the Engineering of Virtual Enterprises and Software", *TR-98-7,* Bilgisayar Mühendisliği Bölümü, Orta Doğu Teknik Universitesi, Şubat 1998.

[4] Vedat Bayar, "A process model for component oriented software development", *Master Tezi*, Bilgisayar Mühendisliği Bölümü, Orta Doğu Teknik Universitesi, Kasım 2001.

[5] Kulak Darly, Guiney Eamonn, *Use Cases Requirements in Context*, Addison-Wesley, New York, 2000.

[6] "CORBA Components version 3.0", *http://www.omg.org*, OMG, Haziran 2002.