SERVİS KALİTESİ DESTEKLİ
OTOMATİK WEB SERVİSLERİ YÜRÜTÜCÜSÜ

Ömer MESCİGİL

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Bölümü

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Haziran 2007

ANKARA

Fen Bilimleri Enstitü onayı

_____

Prof. Dr. Yücel ERCAN

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Ali YAZICI

Anabilim Dalı Başkanı

Ömer MESCİGİL tarafından hazırlanan OTOMATİK WEB SERVİSLERİ YÜRÜTÜCÜSÜ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Erdoğan DOĞDU

Tez Danışmanı

Tez Jüri Üyeleri

Başkan : Doç. Dr. Erdoğan DOĞDU _____

Üye : Yrd. Doç. Dr. Osman ABUL _____

Üye : Yrd. Doç. Dr. Kadir ERTOĞRAL _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

…….

Ömer MESCİGİL

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Doç. Dr. Erdoğan Doğdu**
Tez Türü ve Tarihi : **Yüksek Lisans – Haziran 2007**

Ömer MESCİGİL

Servis Kalitesi Destekli Otomatik Web Servisleri Yürütücüsü

ÖZET

BPEL birleşik web servislerde kullanılan bir endüstri standardıdır. BPEL ile web servislerinden kurulu bir iş akışı oluşturulabilir ve bir BPEL yürütücüsü üzerinde işletilebilir. Standart BPEL ile web servislerinin dinamik çalıştırılması gerçekleştirilebilir; ancak otomatik servis seçim desteği sağlanamamaktadır. Bu tez kapsamında servislerin gerçek zamanlı seçimini sağlamak üzere BPEL diline servis kalitesi eklentisi yapılmıştır. Ayrıca kullanıcının tanımladığı servis kalitesi parametrelerini kullanan bazı basit servis seçim algoritmaları önerilmiştir. Böylelikle yürütücünün iş çıkarma yeteneği arttırılmıştır. Önerilen çözüm servis kalitesini destekleyecek şekilde açık kaynaklı bir BPEL yürütücüsü, ActiveBPEL, üzerinde gerçekleştirilmiştir. Ayrıca servis seçim algoritmalarının başarımını ölçmek üzere kapsamlı bir test ortamı geliştirilmiştir. Yapılan testler ile Servis Kalitesi Destekli BPEL Yürütücüsü'nün BPEL süreçlerine ilişkin ortalama cevaplama sürelerini düşürerek sistemin genel başarımını arttırdığını gösteren sonuçlar sunulmaktadır.

Anahtar Kelimeler: Web servisleri, dinamik servis seçimi, QoS, BPEL, XML.

University : **TOBB University of Economics and Technology**
Institute : **Institute of Natural and Applied Sciences**
Science Programme : **Computer Engineering**
Supervisor : **Associate Professor Dr. Erdoğan DOĞDU**
Degree Awarded and Date : **M.Sc. – June 2007**

Ömer MESCİGİL

Automatic Web Services Execution Engine with QoS Support

ABSTRACT

Business Process Execution Language for Web Services (BPEL) is an industry standard language for web services composition. BPEL allows users to compose and execute web services-based workflows utilizing distributed web services. Standard BPEL allows dynamic execution of web services, but automatic service selection is not supported. We propose to extend WS-BPEL to allow users to specify Quality of Services (QoS) parameters that will guide the BPEL execution engine to select appropriate services during run-time that will improve the engine performance towards higher system throughput. For this we propose to use some simple service selection algorithms that utilize user-specified QoS parameters. We implemented our proposal by extending an open-source BPEL engine, ActiveBPEL, to support QoS parameters. We also developed an extensive test environment to test the performance of the algorithms for service selection. We present the results showing that QoS-supported BPEL execution improves the overall system throughput by lowering the average execution times of BPEL processes.

Keywords: Web services, service selection, QoS, BPEL, XML.

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren hocam Doç. Dr. Erdoğan Doğdu'ya ve kıymetli tecrübelerinden faydalandığım TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyelerine, yüksek lisans programındaki arkadaşlarıma, bana verdikleri manevi destekten dolayı ailem ve arkadaşlarıma teşekkürü bir borç bilirim.

Bu tez Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından desteklenen "Otomatik Web Servisleri Yürütücüsü Projesi" (Proje No: 105E025) çalışmasının bir parçası olarak yapılmıştır. TÜBİTAK'a verdiği destekten dolayı teşekkür ederiz.

İÇİNDEKİLER

Sayfa

ÇİZELGELER LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

KISALTMALAR

Kısaltmalar	Açıklama
BPEL	İş Akışı Oluşturma Dili
KİİSSA	Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması
KİİSSA(G)	Güvenilirlik Etkili Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması
KİİSSA(Y)	Yük Etkili Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması
MİSSA	Minimum İşleme Süresine Göre Seçim Algoritması
MYSA	Minimum Yüke Göre Seçim Algoritması
RSA	Rastgele Servis Seçim Algoritması
SOA	Servis Tabanlı Mimari
SOAP	Yalın Nesne Erişim Protokolü
UDDI	Genel Tanımlama, Keşif ve Bütünleştirme
WSDL	Web Servisleri Tanımlama Dili
XML	Genişletilebilir İşaretleme Dili

SİMGE LİSTESİ

Bu çalışmada kullanılmış olan simgeler açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
μ	Ortalama Değer, Beklenen Değer
σ^2	Varyans Değeri
$X \sim N(\mu, \sigma^2)$	Normal Dağılım
σ	Standart sapma
f(t)	Negatif Üstel Dağılım Olasılık Yoğunluk Fonksiyonu
$f(x;\mu,\sigma)$	Normal Dağılımın Olasılık Yoğunluk Fonksiyonu
$\varphi(x)$	Normal Dağılımın Yoğunluk Fonksiyonu

BÖLÜM 1

1.GİRİŞ

Günümüzde internet ve web teknolojilerinden üzerinde yoğun bir şekilde araştırmaların yapıldığı ve uygulamaların geliştirildiği teknoloji "web servisleri" (web services) ve ilgili teknolojilerdir. Web servisleri teknolojisi web-tabanlı dağıtık uygulamalar geliştirme ve dağıtık uygulamaları bütünleştirmek için geliştirilmiş bir teknolojidir. Kısa zamanda yaygın kabul gören bu teknolojinin en büyük avantajı eski teknolojilere oranla kullanımının daha kolay olması ve standardizasyon sağlamasıdır. Bu tez çalışması kapsamında bu teknolojilerle ilgili son gelişmeler incelenmiş, yenilik getirecek öneriler yapılmış, bu öneriler bir uygulamaya dönüştürülüp gerçekleştirilerek başarıları gözlemlenmiş ve sonuçlar değerlendirilmiştir.

1.1. Çalışmanın Amacı

Bu tez kapsamında web servisleri, birleşik web servisleri, iş akışı oluşturmak için kullanılan diller, mevcut dinamik web servis seçim algoritmaları ve başarımları, web servislerin fonksiyonel ve fonksiyonel olmayan özellikleri ile bunların tanımlanması konularında araştırmalar yapılmıştır. Yapılan araştırmalar neticesinde çoğu önerinin belli bir sorunu çözmeye odaklı olduğu genel başarımı arttırmada yetersiz kaldıkları ayrıca önerilen çözümler ile standartları büyük oranda sağlamanın yanı sıra yeni standartların da tanımlamasını zorunlu kılan bazı gelişmeler ortaya çıkmaktadır. Bu araştırmalardan dersler ve yol gösterimi elde edilerek mevcut sistemin başarımını arttırmaya yönelik bazı geliştirmeler yapılmıştır. Öncelikle önerilen tasarımın gerçekleştirilebileceği açık kaynak kodlu bir iş akışı yürütücüsü araştırılmıştır. Yeteri uzunlukta bir süre yürütücünün kaynak kodları incelenmiş, basit değişikliklerle oluşan çıktılar gözlemlenmiş, değiştirilen kaynak kodlarının nasıl derleneceği, derlenmiş kodların nasıl çalıştırılacağı araştırılmış ve öğrenilmiştir. Bir yandan da web servis yapıları, web servislerin kurulumu, yüklenmesi ve çalıştırılmasına yönelik aşamaların nasıl gerçekleştirildiği bilgileri birçok kaynak taranarak edinilmiştir. Servislerin kullanıldığı mimarilerden olan servis odaklı mimari ile ilgili birçok kaynaktan bilgiler elde edilerek bunların bileştirilmesi ile konu ile ilgili

derinlemesine bilgi sahibi olunmuştur. Web servislerine ilişkin fonksiyonel olmayan özellikler olan servis kalitesi parametrelerinin en sık kullanılanları incelenmiş, değerlerinin nasıl ve hangi araçlarla elde edilebileceği araştırılmıştır. Kullanımı çok yaygın bazı servis kalitesi parametrelerin kullanımına karar verilmiş ve birtakım mevcut çalışmalara yenilik teşkil edebilecek parametreler önerilmiştir. Servislerin birlikte işler ve birleştirilebilme özellikleri kullanılarak oluşturulabilecek iş akışlarına ilişkin bilgiler edinilmiş ve bazı örnek uygulamalar incelenilerek bilgi sahibi olunmuştur. Elde edilen tüm birikim ve oluşturulan tasarım açık kaynak kodlu yürütücüye yapılan ekler sayesinde işler hale getirilmiştir. Ayrıca önerilen sistemi test etmek üzere bir test yazılımı oluşturulmuş, test ortamı tasarlanmış ve birtakım testler yapılarak sonuçlar ortaya konmuştur. Bütün bahsedilen araştırma, tasarım, gerçekleme ve test aşamaları "Servis Kalitesi Destekli Otomatik Web Servisleri Yürütücüsü" Projesi ve bu tez kapsamındadır.

"Servis Kalitesi Destekli Otomatik Web Servisleri Yürütücüsü" Projesi ile ortaya konulan özelliklerin bir araya getirilmesi ve işler hale sokulması tez projesi kapsamında değerlendirilmektedir. Çalışma ile ortaya konan en önemli amaçlardan birisi mevcut teknolojiye yenilik getirmenin yanı sıra, mevcut altyapının da en elverişli şekilde kullanılmasını sağlamaktır. Böylelikle de sonuç olarak endüstride ve akademik ortamda yaygın olarak kullanılan açık kaynak kodlu bir sisteme yeni özelliklere kazandırıp sistemin başarımın arttırılması sağlanmaktadır. Başarımı arttırmak için servislerin fonksiyonel olmayan özelliklerinden servis kalitesi parametrelerine ilişkin değerler ölçülmekte, ölçülen değerler karşılaştırılmakta ve servis seçiminde kullanılmaktadır.

İkinci bölümde; web servisleri ve birleşik web servislerine ilişkin bilgiler aktarılmıştır. Üçüncü bölümde; şu ana kadar servis kalitesi parametrelerine ilişkin yapılan çalışmalar incelenmiş ve özellikle birleşik web servislerinde kullanımları araştırılmıştır. Ayrıca BPEL diline yapılan servis kalitesi ekleri, gerçek zamanlı servis seçiminin hangi algoritmalar kullanılarak sağlandığı ve algoritmalarla ilgili bilgiler anlatılmıştır. Bu bölümü takiben açık kaynak kodlu yürütücünün mevcut mimarisi ve yürütücüye önerilen eklere ilişkin tasarımın nasıl gerçekleştirildiği anlatılmıştır. Beşinci bölümde; oluşturulan test ortamı ile ilgili bilgiler aktarılmakta, performans ölçümlerinin nasıl yapıldığı anlatılmakta ve test sonuçları ile bu

sonuçların yorumlarına yer verilmektedir. Son bölümde ise genel sonuçlar açıklanarak geleceğe yönelik neler yapılabileceği irdelenmektedir.

BÖLÜM 2

2. WEB SERVİSLERİ, BİRLEŞİK SERVİSLER

Bu tez kapsamında web servisleri, birleşik web servisleri, dinamik servis seçimi ve servis kalitesi parametreleri ön plana çıkmaktadır. Bu bölümde öncelikle web servislerinde kullanılan protokoller açıklanmakta sonrasında birleşik web servislerine değinilmektedir.

2.1. Web Servisleri

Web servisleri bilgisayar ağları üzerinden makineden makineye etkileşimi birlikte işlerliği (interoperability) destekleyerek sağlayan yazılım uygulamalarıdır. Bir anlamda Servis Tabanlı Mimari'deki servisler olarak düşünülebilir; ancak bu mimarideki her servisin web servis olma zorunluluğu olmadığı unutulmamalıdır [7-9]. Örneğin; Oberleitner, Rosenberg ve Dustdar' ın çalışmasında tüm servislerin web servis olmadığı, COM, CORBA ve .NET objeleri biçiminde elektronik servislerin olabildiği ve melez birleşik elektronik servis yapısının standartlar kullanılarak nasıl oluşturulabileceğine değinilmektedir [11]. Bu çalışmada standart olarak sistemlere ilişkin UML yapıları kullanılmaktadır.

2.2. Web Servisleri Protokolleri: XML, SOAP, WSDL, UDDI

Web Servisler dağıtık uygulama bileşenleridir ve web servislere özel bazı standartlar-teknolojiler mevcuttur. Bu teknolojilerin önde gelenleri XML, SOAP ve WSDL'dir.

XML (extensible markup language) veri saklama ve veri alışverişinde kullanılan bir işaretleme dilidir. Günümüzde birçok yazılım diğer yazılımlarla veri alışverişini XML formatı üzerinden yapmaktadır. Ayrıca XML' de dilin bileşenleri kullanıcı tarafından belirtildiği için XML diğer işaretleme dillerinin temelini teşkil etmektedir.

Böyle diller XML tabanlı diller olarak nitelendirilmektedir. Aşağıdaki örnekte de görüldüğü gibi ad ve soy ad verilerinden oluşan bir XML dosyası farklı kullanıcılar tarafından değişik biçimlerde oluşturulabilir. Bu durum da esnekliği beraberinde getirmektedir.

```
<kullanicilar>
    <kullanici id="1">
        <ad>A</ad>
        <soyad>B</soyad>
    <kullanici>
    <kullanici id="2">
        <ad>C</ad>
        <soyad>D</soyad>
    </kullanici>
</kullanicilar>
```

Şekil 2.1. XML belgesi örneği

SOAP (simple object access protocol) merkezi olmayan ortamlarda yapısal bilgi alışverişini sağlayan bir protokoldür ve XML tabanlıdır. Protokol çatısı herhangi bir programlama dilinden bağımsızdır, bu özelliği standart olarak kullanılmasında en büyük etkendir [7]. Protokolün mesaj yapısında;

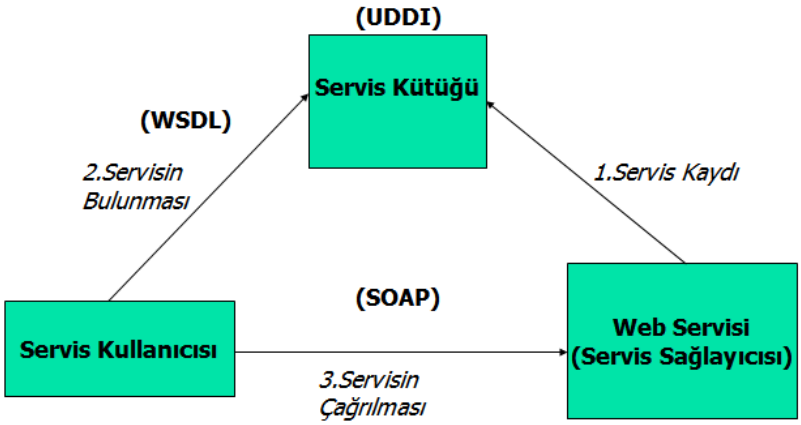
- Zarf
  - Başlık
  - Ana kısım belirtilmektedir.

WSDL de XML tabanlı bir dildir. Web Servislerini tanımlamak için kullanılır. Soyut bağlanma ve somut bağlanma olmak üzere iki temel alanı bulunmaktadır. Soyut kısmında mesaj ve operasyon tanımları; somut kısmında ise ağ protokolü bildirimi yer almaktadır. Bu ayrımın sebebi ise ilerde ağ protokollerinin değişme ihtimaline karşı soyut kısımların aynı şekilde kullanılabilirliğini sağlamaktır [9].

Web Servis Mimarisine bakılacak olursa ana bileşenler olarak şunlar sayılabilir;

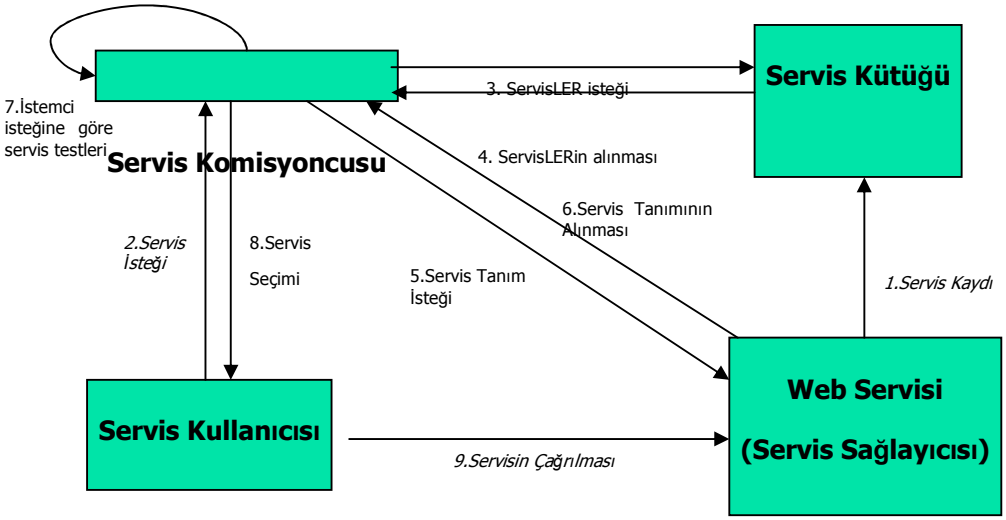
- Web Servisi (Servis Sağlayıcısı)
- Servis Kütüğü
- Servis Kullanıcısı

Yapılan ana işlemlere bakıldığında ise yayımlama, bulma ve bağlanma işlemlerinin yer aldığı gözükmektedir [10]. Temel web servis mimarisine ilişkin yapı Şekil 2.2' den görülebilir.



Şekil 2.2. Web Servis Mimarisi

Temel web servis mimarisinin genişletilmiş halinde servis komisyoncusu bileşeni de mimaride yer almaktadır. Bu bileşen servis kalitesi parametrelerini kullanarak servis seçiminde yer alır ve servislerin gerçek zamanlı servis kalitesi değerlerini takip eder. Böyle bir mimari ile Tian, Gramm, Ritter ve Schiller'in çalışmasında karşılaşılmaktadır [12]. Bu mimariye göre gerçekleşen servis isteğinin yapılması adımından servisin çağrılmasına kadar gerçekleşen adımlar Şekil 1.2 'den takip edilebilir.



Şekil 2.3. Web Servis Mimarisi (Genişletilmiş)

UDDI, servis bilgilerinin tutulduğu özel bir veritabanı protokolü olarak düşünülebilir. Temel Web Servis Mimarisi' ndeki yayımlama ve bulma adımlarında kullanılmaktadır. Web servislere ilişkin kaydı tutulan bilgiler iki ana kısımda incelenebilir. Bunlar şu şekilde belirtilebilir;

- Servis tip tanımları (WSDL'lerle ilgili meta veriler)
- İş tanımları
  - Beyaz sayfalar (adres ve erişim bilgileri)
  - Yeşil Sayfalar (servislerle ilgili teknik bilgi)
  - Sarı Sayfalar (endüstri kategorisi)

Kamuya açık UDDI kullanımında bazı sıkıntılar göze çarpmaktadır. Bunların başında servislere ilişkin tutulan bilgilerin güncel olmaması ve doğruluklarının düşük olması sayılabilir. Bu sorunların aşılması amacıyla çeşitli çalışmalar yapılmaktadır. Örnek olarak; Du Huai, Liu' nin "Ad-UDDI: An Active and Distributed Service Registry" çalışmaları sayılabilir [15]. Bu çalışmada dağıtık yapıda bir UDDI tasarımı ile özel ve kamuya açık UDDI'larda bilgi paylaşımı ve servislerin gerçek zamanlı takibi ile kullanıcılara daha doğru ve güncel servis

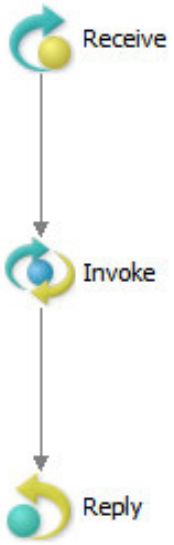
bilgilerinin sunulması amaçlanmaktadır. Önerilen protokol test edilmiş ve mevcut yapıya göre daha başarılı olduğu testler neticesinde gösterilmiştir.

2.3. Birleşik Servisler

Web servislerin belli bir işlevi yerine getiren, tekrar kullanılabilir, diğer bileşenlerden bağımsız bir yapıda olduğundan bahsedilmişti. Bunların dışında web servislerin daha doğrusu servis odaklı mimarideki servislerin en önemli özelliği birleştirilip belli bir iş akışını gerçekleştirebilir olmalarıdır. Servislerle kurulan iş akışlarında yaygın olarak BPEL dili kullanılmaktadır [13].

BPEL, birleşik web servislerin iş akışında tanımlanmasında kullanılan XML tabanlı en popüler dillerden biridir. Tanımlanan aktiviteler ile iş akışı oluşturulmaktadır.

BPEL dili ile tanımlanmış örnek bir iş akışı şekil 2.4 ve şekil 2.5'ten görülebilir.



Şekil 2.4. BPEL Dili ile Oluşturulmuş Bir İş Akışı (Grafik Gösterim)

```
<process>
    <flow>
            <receive createInstance="yes"
                  name="…..”operation=”…..”
                  partnerLink=”…..” portType=”….”
                  variable=”…..”>
            </receive>
            <invoke inputVariable=”…..”
            name=”…..” operation=”…..”
            outputVariable=”…..” partnerLink=”…..”
            portType=”…..”>
            </invoke>
            <reply name=”…..”  operation=”…..”
                partnerLink=”…..” portType=”….”
                variable=”…..”>
            </reply>
   </flow>
</process>
```

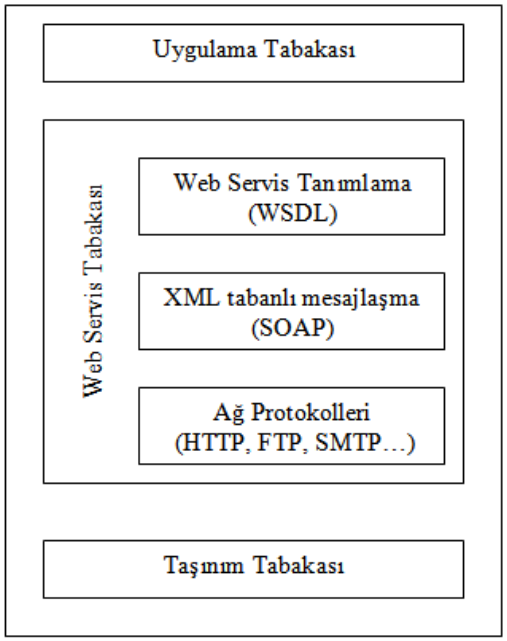
Şekil 2.5. BPEL Dili ile Oluşturulmuş Bir İş Akışı (Kaynak Kodu)

Servislerin birleştirilerek oluşturduğu bir iş akışında en önemli işlevlerden biri herhangi bir zaman anında iş akışında hangi noktada olunacağını tahmin edebilmektir. Böylelikle istenmeden sisteme dâhil edilen hataların önüne geçilebilme şansı elde edilmiş olur. Ayrıca iş akışının en elverişli şekilde oluşturulabilmesi için ipuçları elde edilebilir. Bu özelliği sağlamak üzere yapılan çalışmalardan biri Biswas ve Vidyasankar'ın çalışmalarıdır [14]. Bu çalışma ile iş akışına kazandırılan özellik istenen herhangi bir zaman anında hiyerarşik bir iş akışının durumunu saptayabilmektir. Devamında iş akışında oluşabilecek potansiyel ölümcül kilitlenme durumlarının önüne geçilebilecek çalışmalar ortaya konulmaktadır.

BÖLÜM 3

3. BİRLEŞİK SERVİSLER İÇİN SERVİS KALİTESİ DESTEĞİ VE SERVİS SEÇME ALGORİTMALARI

İnternet Başvuru Modeli'ne göre beş adet katman bulunmaktadır. Bunlar; fiziksel katman, veri bağı katmanı, ağ katmanı, taşınım katmanı (transport layer) ve uygulama katmanı (application layer) olarak belirtilmektedir. Son zamanlarda doğan ihtiyaçlar ve teknolojik gelişmeyle ilintili olarak bazı ara katmanlar belirtilebilmektedir. Tian, Voigt, Naumowicz, Ritter, Schiller çalışmalarında İnternet Başvuru Modeli'ni temel alarak web servis tabakasını taşınım katmanı ile uygulama katmanı arasına yerleştirmişlerdir [16]. Web servis katmanı, WSDL, SOAP, FTP ve SMTP gibi birçok standart internet protokolünden oluşmaktadır. Web Servisleri katmanında yer alan protokol yığıtına ilişkin yapılanma Şekil 2.1'de aşağıdaki gibidir [16].

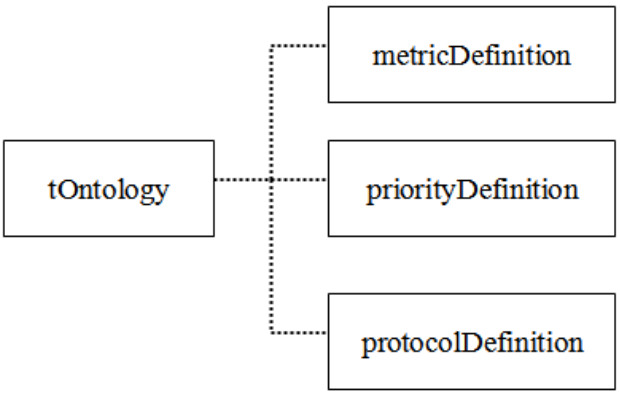


Şekil 3.1. Web Servis Protokol Yığıtı

Servis Kalitesi (Quality of Service - QoS) ilk olarak ağ tabakasına yönelik bir terim olarak karşımıza çıkmaktadır. Genel bir bakış açısıyla, servis kalitesi kullanımı ile ağ trafiği ve bant genişliğine bağlı ağ performansı artışı hedeflenmektedir. Yapılan çalışmalarda mevcut uygulamalara servis kalitesi özelliğinin kazandırılmasına yönelik öneriler yer almaktadır. Örnek olarak web üzerinden erişilebilen multimedya uygulamalarına servis kalitesi kazandırmaya yönelik bir çalışma olan Gu, Nahrstedt, Yuan, Wichadakul'nın çalışmaları verilebilir [17]. Son zamanlarda servis kalitesi, web servisleri katmanında bazı ağ tabakasının performansını belirtebilecek parametreler ile ifade edilerek ilk kullanımından farklı bir bakış açısı ile ele alınabilmektedir. Mani ve Nagaraja'nın tanımlarına göre; servis kalitesi, servis sağlayıcısının ağ kaynaklarının varlığı ile servis isteğinde bulunan tarafın ihtiyaçlarının eşleştirilmesini kapsamaktadır [18]. Ayrıca web servislerine ilişkin servis kalitesi denilince performans, güvenilirlik, bulunulabilirlik ve güvenlik gibi web servislerinin işlevsel olmayan özelliklerinden bahsedilmektedir. Tian, Gram, Naumowicz, Ritter ve Schiller'in çalışmasında istemciler ve sunucular tarafından tanımlanmış olan iki çeşit servis kalitesi parametresi üzerinde durmaktadırlar, bunlar; web servisleri katmanı, ağ tabakası servis kalitesi desteği olarak belirtilmektedirler [19]. Bu çalışmaya göre istemci ve sunucu tarafından tanımlanan web servis katmanına yönelik servis kalitesi parametreleri: işleme süresi, cevaplama süresi, servis kullanım ücreti, bulunabilirlik, güvenilirlik, güvenlik protokolleri, istek oranı, işlem-bilgi (transaction) olarak belirtilmektedir. Ağ tabakasına yönelik parametreleri ise ağ gecikmesi, bant genişliği, seğirme (jitter) ve paket kaybı olarak ele alınmıştır. Bu çalışmada servis kalitesine ayrıca bir ontoloji yapısı tanımlanmış ve servis kalitesi parametreleri bu yapıda belirtilmektedir. Servis kalitesi tanımlanmasında ontolojinin kullanımı ile önceden tanımlanmış parametrelere alternatif olarak kullanıcının sonradan parametre tanımı yapabileceğine değinilmiştir.

WS – QoS Ontolojisi olarak adlandırılan ontoloji yapısında; kullanıcı tanımlı metrikler, öncelikler ve protokol ifadelerinin hepsi ontoloji olarak öznitelik değerine sahiptir. Böylelikle referans tiplerin tanımlı olduğu WS-QoS Ontolojisini içeren dosyaya referans gösterilmiş olur. Müşterilerin başka bir deyişle kullanıcıların

tanımlamış olduğu "priority definition" elemanında insan tarafından okunabilir (human readable) bir ifadeyle hangi metrik önceliğinin kullanıldığı ve verilen farklı isim belirtilir. “metric definition” kısmında yine farklı bir ad ve insan tarafından okunabilir bir ifadeyle neyin ölçüldüğü yer alır ayrıca standart bir birim belirtimi yapılır ve değerlerin nasıl ölçüleceğine yönelik yol gösterimi vardır. “protocol definition” elemanında ise protokolün tanımlanmış olduğu özet dokümanın URL’ı belirtilmelidir. WS-QoS Ontoloji yapısı Şekil 2.7’den incelenebilir [19].



Şekil 3.2. Kullanıcı Tanımlı Servis Kalitesi, WS – QoS Ontolojisi

Web servisleri protokolleri incelendiğinde web servislerinin tanımlarında servis kalitesine ilişkin özelliklerin belirtilmediği görülmektedir. Servis kalitesi parametrelerini tanımlamak üzere geliştirilmiş standartlar henüz mevcut değildir. Ancak servis tanımlarına ek olarak servis kalitesi tanımlarını yapmak üzere ontoloji benzeri yapılar önerilmiştir.

Papaioannou, Tsesmetzis, Roussaki ve Anagnostou’ın yapmış olduğu çalışmada web servislerinin servis kalitesi destekli hale getirilmesi için tasarlanan “Web Servisleri için Servis Kalitesi Ontoloji Dili” önerilmiştir [20]. Ayrıca çalışmada geliştirilen ontoloji dili ile istenilen servis kalitesi parametrelerini tanımlamak için esnek ve birlikte işler bir şema sağlanmaktadır.

Birleşik web servislerinin kullanıldığı bir iş akışı düşünüldüğünde servis kalitesinin kullanımı başka bir açıdan ele alınabilmektedir. Bu bakış açısına göre iş akışındaki her servisin ayrı ayrı servis kalitesi özellikleri parametrelerle belirtilebilir veya birleşik servislerin doğası gereği iş akışının tümü bir servis olarak ele alınıp iş akışına ilişkin servis kalitesi parametreleri tanımlanabilir. Böyle bir durumda parametreler yerel ve genel olmak üzere iki ana başlık altında toplanabilmektedir. Brandic, Benkner, Engelbrecht ve Schmidt'in çalışmalarında servis kalitesi açısından ele alınan bu iki farklı başlığa değinmişler ve kullanmışlardır [21]. Yapılan önemli bir çalışma kullanıcının belirttiği parametre değerlerine göre tabandan tepeye (bottom up) veya sezgisel-analitik yaklaşımlarla yerel (local) ve genel (global) değerler arası dönüşümü sağlayabilmeleridir. Örneğin; kullanıcı tarafından yerel değerler belirtilmiş ise tabandan tepeye hesaplama yöntemiyle iş akışı için genel servis kalitesi değeri veya kullanıcı tarafından genel değerler verilmiş ise sezgisel-analitik yöntemle yerel değerler elde edilebilir. Diğer bir durum ise bu iki durumu da kapsayan yerel değerlerden ve genel değerlerden bazılarının kullanıcı tarafından verildiği karışık (melez) durumdur.

Web servislerine ilişkin servis sağlayıcı tarafından tanımlanan servis kalitesi parametrelerini elde etmeye yönelik veya sağlayıcının tanımlamadığı ama yürütücü tarafında servise ilişkin parametrelerin elde edilebileceği bazı yöntemler mevcuttur. Bu yöntemler ışığında servis kalitesi parametrelerinin tanımlanması (ontolojik tanım veya servis derecelerine ilişkin tanım), elde edilmesi ve karşılaştırılmasına ilişkin birçok çalışma yapılmıştır, yapılmaktadır [22-25].

Brandic, Benkner, Engelbrecht ve Schmidt çalışmalarında servis kalitesi destekli bir dil tanımlanmıştır. Tanımlanan dilin, BPEL dilinin bir uzantısı olduğu belirtilmektedir. Burada tanımlanan iş akışı soyut bir iş akışını göstermektedir. Soyut iş akışının somut iş akışından farkı pazarlıktan önce belirtilmiş olmasıdır. Ayrıca soyut iş akışında potansiyel servislerin bulunduğu sistem kütüklerine referans gösterilmektedir. Bu dile ilişkin örnek aşağıda verilmektedir [21].

```
...
<invoke name="start" portType="appex"
    operation="start" inputVar="startRequest">
    <qos-constraints reqDescVar="startReqDesc">
        <candidate-registry inputVar="queryRequest"
            ...
            wsdl="http://kim:9357/registry/reg?wsdl"/>

            <qos-constraint name="beginTime"weight="0.3"
              value="18-08-2005 12:00:00,0 MET" />
            <qos-constraint name="endTime" weight="0.2"
                value="18-08-2005 14:00:00,0 MET" />
            <qos-constraint name="price" weight="0.5"
              value="20.00" />
    </qos-constraints>
</invoke>
...
```

Şekil 3.3. BPEL ile Tanımlanmış Servis Kalitesi Destekli Soyut Bir İş Akışı

Servis kalitesi parametreleri ve bu parametrelerin tanımlarına ilişkin çalışmalardan sonra bahsedilebilecek diğer bir konu servis kalitesine göre servis seçim algoritmalarıdır. Bu tip algoritmalar ile temelde yapılan iş, servisin sahip olduğu değer ile kullanıcının istekte bulunduğu değerlerin karşılaştırılmasıdır. Rifaieh, Chukmol ve Benharkat çalışmalarında değişik standartların kullandığı EDI (Electronic Data Interchange) mesajlarının yarı otomatik anlamsal karşılaştırıldığı algoritmalar anlatılmaktadır [26]. Bu çalışmada XML şemaları pivot format olarak kullanılmaktadır. Önerilen algoritma, iki EDI mesajını girdi olarak alır ve metin yapılarını ve veri yapılarını karşılaştırarak elemanlar arasındaki temel benzerlikleri hesaplar.

Servis seçimi ile aynı fonksiyonel özelliklerle donatılmış farklı servislerin, fonksiyonel olmayan özelliklerinin karşılaştırılarak en uygun servisin seçilebilmesi sağlanabilmektedir. Servislerin fonksiyonel olmayan özelliklerinden servis kalitesi özellikleri kullanılarak servislerin seçimi gerçekleştirilebilmektedir. Seçim esnasında tek bir servis seçimi yapılabildiği gibi servislerin birleştirilmesinden oluşan bir iş akışında da servislerin önceden seçilip iş akışının yapılması yoluna gidilebilmektedir.

Buna yönelik yapılan önceki çalışmalardan ikisi Esmaeilsabzali ve Larson'ın çalışmaları ile [27]; Yu ve Lin'in yaptıkları çalışmalardır [28]. Esmaeilsabzali ve Larson'ın çalışmasında oyun kuramının esasları ön plana çıkarılmakta ve seçim esnasında kullanılmaktadır. Yu ve Lin'in çalışmasında ise web servis mimarisinde servis kalitesi parametrelerinin elde edilmesi ve servis seçiminden sorumlu servis komisyoncusu adlı bir yazılım ara katmanı kullanılmaktadır. Yu ve Lin'in diğer bir çalışmalarında ise birden çok servis kalitesi parametresinin olduğu durumda servis seçiminin nasıl yapılabileceğine yönelik ayrıntılı bir öneri sunulmaktadır [29].

3.1. Servis Kalitesi Kısıtları ve Birleşik Servislerde Kullanımı

Bu bölümde web servislerine ilişkin önerilen servis kalitesi parametrelerinin hangileri olduğu ve özellikleri belirtilmektedir. Ayrıca BPEL diline yapılan servis kalitesi eklentileri açıklanmakta ve servis seçim algoritmaları anlatılmaktadır.

3.1.1. Servis Kalitesi Kısıtları (QoS Constraints)

Bu çalışmada önceki yapılan çalışmalardan yararlanılarak BPEL diline servis kalitesi gereksinimlerini tanımlamak üzere aşağıdaki ekleri önerdik;

- İşleme süresi
- Ücret (yüksek değer, düşük değer)
- Bulunabilirlik
- Güvenilirlik

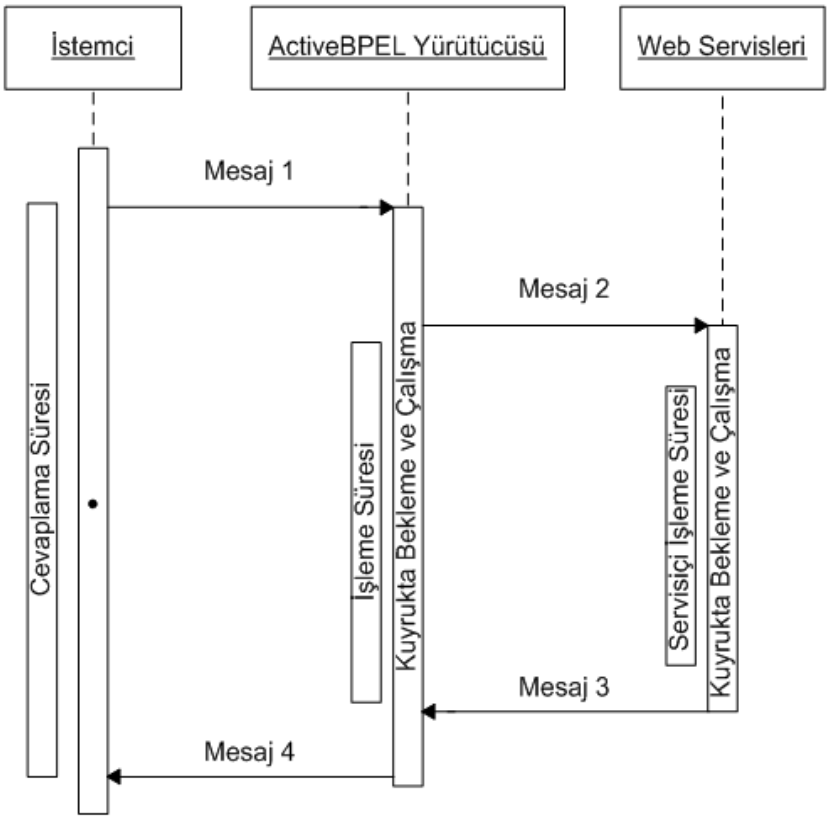
Diğer çalışmalardan farklı olarak güvenilirlik kullanıcının isteğine göre geçmişe yönelik başarı oranını belirtmektedir. Ayrıca ücret iki sınır içine alınarak en düşük değerin de kullanıcı tarafından belirtilebilmesine olanak sağlanmıştır.

Önerilen parametrelere daha detaylı değinmeden ve tanımlamalarını vermeden önce bazı temel tanımları belirtmekte yarar vardır. Zaman açısından değerlendirildiğinde

üç değişik terim ortaya çıkmaktadır. Bunlar; cevaplama süresi, işleme süresi ve servis içi işleme süreleridir. Belirtilen sürelerin tanımları yapılırken üç katılımcı taraf dikkate alınmaktadır. Taraflar genel haliyle istemci ve sunucu olarak belirtilebilir; ancak sunucunun ikiye ayrılması ile yürütücünün işletildiği sunucu, web servislerin işletildiği sunucu ve istemci olmak üzere üç katılımcı elde edilmektedir. Cevaplama süresi; istemcinin istekte bulunma anı ile isteğe cevap verilme anı arasında geçen süre olarak tanımlanmaktadır. İşleme süresi ise; yürütücünün iş akışındaki web servisi çağırma anı ile web servis cevabının yürütücüye ulaştığı an arasında geçen süre olarak belirtilebilir. Bu süreler arasında tanımı gereği en küçük olan ise servis içi işleme süresidir. Servis içi işleme süresi ile ilgili olarak; web servisin uzaktan çağrılan metodunun kod bloğunun işletilmesi için gereken süre tanımlaması yapılmaktadır. Bu süreler büyüklükleri açısından karşılaştırıldığında aşağıdaki gibi bir ilişki olduğu gözlemlenmektedir;

Cevaplama Süresi > İşleme Süresi > Servis İçi İşleme Süresi

Süre tanımlarının daha iyi anlaşılabilmesi için Şekil 3.4 incelenebilir.



Şekil 3.4. Cevaplama Süresi, İşleme Süresi ve Servis-İçi İşleme Süresi

Tez kapsamında gerçekleştirilen projede kullanılan servis kalitesi parametreleri ve bunların tanımları aşağıda incelenmektedir.

- İşleme süresi
    - ActiveBPEL yürütücüsünün web servisi çağırma anı ile web servisten yürütücüye cevap gelmesi arasında geçen süredir.
    - Zaman Damgası (time stamp) #A;
        - Web servisten gönderilen cevabın yürütücünün ilgili modülüne vardığı andır.
    - Zaman Damgası #B;
        - Prosesten web servise çağrının gönderildiği andır.
    - Bu tanımlamalara göre işleme süresi şu şekilde formülize edilebilir;
        - İşleme Süresi = Zaman Damgası #A – Zaman Damgası #B

- Ücret
    - Gerçek dünyadaki çoğu servisin işletilme maliyeti vardır. Ücret servis kalitesi parametresi de her bir servisin işletilme maliyetini belirtmektedir.
    - Alt sınır ve üst sınır olarak iki bileşeni vardır. Bunlar ayrı parametreler olarak değerlendirilmektedir. Yaygın olarak kullanılan üst sınırdır. Diğer bir deyişle bir servisin işletilmesi sonucu elde edilebilecek en yüksek maliyettir. Genel kullanımdan farklı olarak proje kapsamında önerdiğimiz ücrete ilişkin alt sınırın belirtildiği parametredir. Bu tanımlamanın varlık sebebi ise ücretin bazı kullanıcılara göre kalitenin bileşenlerinden biri olarak ortaya çıkabilmesi ve belli bir ücretin altına düşecek servisin kullanıcı tarafından seçilmemek istenebileceği durumun da değerlendirilmesini sağlamaktır.

- Bulunabilirlik
    - Web servisin gerçek zamanlı olarak kullanıma hazır olup olmadığını belirten bir servis kalitesi parametresidir.

- Güvenilirlik
    - Güvenilirlik parametresinin iki farklı şekilde kullanılabileceği hesaba katılmaktadır. İki farklı kullanıma ilişkin tanımlar şu şekilde yapılabilir;
        - Geçmişe yönelik bulunabilirliğin yüzde olarak belirtilmesinde kullanılması tasarlanan servis kalitesi parametresidir. Diğer parametrelere göre daha geçmişe dönük olması sebebi ile web servislerle ilgili istatistikî bir bilgi olarak değerlendirilebilir.
        - Kullanıcının istekte bulunduğu işleme süresinin altında bir işleme süresi ortalamasına sahip olan servis başarılı olarak nitelendirilmektedir. Aksi durumda ise servis başarısız olarak değerlendirilir. Bir servisin, servise yapılan tüm çağrılara göre

başarılı olma yüzdesi güvenilirliği ifade etmektedir. Projenin gerçeklenmesi sonrası yapılan testlerde güvenilirlik parametresi bu şekilde kullanılmaktadır. Test ortamına ilişkin bilgilerin aktarıldığı 4. bölümde bu konu ile ilgili daha detaylı bilgi aktarılacaktır.

3.1.2. BPEL Diline Yapılan Servis Kalitesi Ekleri

Proje kapsamında oluşturulan servis kalitesi destekli BPEL dilinin yapılan araştırmalarda görülen dillerden en büyük farkı daha esnek bir yapıya sahip olmasıdır. Mevcut dillerde servis kalitesi parametrelerinin değerleri iş akışının oluşturulması aşamasında belirtilmektedir. Hâlbuki bizim oluşturduğumuz dilde BPEL'in kendi fonksiyonları kullanılarak kullanıcının gönderdiği parametrelerin çalışma zamanında alınması ve ilgili değişkenlere atanması sağlanmaktadır. Örnek; getVariableData fonksiyonunun kullanımıdır. Dilin bazı karakteristiklerine değinmek gerekirse; servis çağrısının yapıldığı invoke aktivitesinin altında tanımlanmış qos aktiviteleri ile servis kalite parametreleri tanımlanır. Qos aktivitesinin `paramName` ve `value` (`price` için `maxValue`, `minValue` olmak üzere iki değer) özellikleri tanımlanmıştır. Aşağıdaki örnekte de görüldüğü gibi bir tanesi iki değere sahip olmak üzere 5 adet servis kalitesi parametresi tanımlanmıştır.

```
<invoke inputVariable="sumOfNumsRequest"
                name="InvokeWsType1" operation="sumOfNums"
                outputVariable="sumOfNumsResponse"
                partnerLink="webServiceType1" portType="ns1:TestWs">
                <qos paramName="price"
                        maxValue="bpws:getVariableData('qosMethodRequest',
                        'qosPriceMax')"
                        minValue="bpws:getVariableData('qosMethodRequest',
                        'qosPriceMin')"/>
                <qos paramName="processTime"
                        value="bpws:getVariableData('qosMethodRequest',
                        'qosProcTime')"/>
                <qos paramName="availability"
                        value="bpws:getVariableData('qosMethodRequest',
                        'qosAvail')"/>
                <qos paramName="reliability"
                        value="bpws:getVariableData('qosMethodRequest',
                        'qosReliabil')"/>
                <target linkName="assignAddress-to-invokeWsType1"/>
        <source linkName="invokeWsType1-to-assignRetParams"/>
</invoke>
```

Şekil 3.5. Servis Kalitesi Destekli BPEL Dili

Oluşturulan eklenti standart BPEL 1.1 diline yapılmış bir eklentidir. Bu dili yorumlamak ve anlaşılmasını sağlamak için ActiveBPEL Yürütücüsünde de bazı değişiklikler yapılmıştır. Bu kısma sonraki bölümde değinilecektir.

3.2. Servis Seçim Algoritmaları

Servis kalitesi ile ilgili yapılan çalışmalara bakıldığında bu alanın üç ana başlık altında toplanabildiği görülmektedir. Bunlardan ilki servis kalitesi parametrelerine ilişkin kullanıcının tanımladığı değerleri ilgili servis seçim modüllerine iletmek olarak tanımlanabilir. Bir diğeri servislere ilişkin gerçek zamanlı ya da istenilen zamanda parametre değerlerini elde etmek olarak belirtilmektedir. Sonuncusu ise belirtilen iki tarafa yönelik servis kalitesi parametrelerinin değerlerin kıyaslanması ve bu kıyaslama neticesinde en uygun servisin seçiminin gerçekleştirilmesidir. Bu kapsamda tez projesi dâhilinde çeşitli servis seçim algoritmaları kullanılmıştır. Kullanılan servis seçim algoritmaları şu şekilde belirtilmektedir;

1. Minimum İşleme Süresine Göre Seçim Algoritması (MİİSSA):

Her servis çağrısının cevabı verildiğinde yürütücü tarafında servislere ilişkin işleme süreleri hesaplanmaktadır ve yürütücü içindeki veri yapısına kaydedilmektedir. Aynı servise ilişkin birden çok cevabın yaratıldığı durumda işleme sürelerinin aritmetik ortalaması aşağıdaki 3.1 formülü ile hesaplanarak yine ilgili veri yapısına kaydı yapılmaktadır.

$$OrtalamaİşlemeSüresi = \frac{\sum_{x}^{İstekSayısı} İşlemeSüresi_{x}}{İstekSayısı} \qquad (3.1)$$

Servis seçim adımında servislerin işleme süreleri ortalamalarının en küçük değeri sahip olanı seçilmekte ve kullanıcının isteği bu servise yönlendirilmektedir.

2. Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması (KİİSSA):

Servislerin işleme süreleri minimum işleme süresine göre seçim algoritmasına benzer şekilde yapılmaktadır. Bu algoritmadaki farklılık servis seçiminde gerçekleşmektedir. Seçimde kullanıcının gönderdiği işleme süresi değeri referans alınarak bu değerin altındaki en yüksek işleme süresine sahip servis, yoksa bu değerin üstündeki en küçük işleme süresi değeri seçilmektedir.

3. Minimum Yüke Göre Seçim Algoritması (MYSA):

Her bir servis çağrısında sunucuda bulunan servis üzerinde bir istek yürütülmektedir. Her yürütülen ve henüz cevabı iletilmemiş olan her bir istek servis üzerinde yük oluşturmaktadır. Minimum yüke göre servis seçim algoritmasında her bir istekte bulunulan servisin yükü arttırılmaktadır. Seçim aşamasında en az yüke sahip olan servisin seçilmesi sağlanmakta ve servisten cevap veya cevaplar geldiği anda ilgili servisin yükü cevap sayısı kadar azaltılmaktadır.

$$Yük = ServisteYürütülenİstekSayısı + KuyruktaBekleyenİstekSayısı \quad (3.2)$$

4. Rastgele Servis Seçim Algoritması (RSA):

Bu algoritma ile sisteme kayıtlı servisler arasından rastgele bir servis seçilmekte ve istek seçilen servise yönlendirilmektedir. Seçim aşamasında servislerin karakteristiklerini öğrenmek ve davranışlarını yorumlamak için algoritmanın çalışması için gerekli olmasa da servislerin yükleri, işlem süreleri ve güvenilirliklerine ilişkin bilgiler toplanmaktadır.

5. Güvenilirlik Etkili Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması (Güvenilirlik Etkili KİİSSA)

Daha önce anlatılan kullanıcı isteğine en yakın işleme süresine göre seçim algoritmasına bir eklenti yapılarak güvenilirlik etkisi de servis seçimi aşamasında hesaba katılmaktadır. KİİSSA'da servis seçiminde önce toplana verilere ek olarak servislere ilişkin güvenilirlik değerleri veri yapısına kaydedilmektedir. Bu değer servisin ilgili zaman aralığında toplam çağrı üzerinden kaç çağrıya başarılı yanıt verdiğinin yüzde olarak ifadesidir. Güvenilirlik ve güvenilirlik yüzdesinin hesaplanması formül 3.3 ve 3.4'te şu şekilde belirtilmektedir.

$$Güvenilirlik = \% \frac{BaşarılıİstekSayısı}{ToplamİstekSayısı} \quad (3.3)$$

$$GüvenilirlikYüzdesi = \frac{\sum_{n=1}^{İstekSayısı} EskiGüvenilirlikYüzdesi}{İstekSayısı} \quad (3.4)$$

Servis seçim adımında ise servisin sahip olduğu ortalama işleme süresine kendisinin güvenilirliğe bölümü ile elde edilen değer ilave edilerek yeni bir ortalama işleme süresi değeri elde edilmektedir. Bu değer kalıcı değildir sadece seçim aşamasında kullanılmaktadır ve her bir seçim adımında yeniden hesaplanmaktadır. Yeni ortalama işleme süresinin hesaplanmasına ilişkin matematiksel ifade formül 3.5'te gözükmektedir.

$$YeniOrtİşlSür = OrtİşlSür + OrtİşlSür * 100 / GüvenilirlikYüzdesi \quad (3.5)$$

Formülden de görüldüğü gibi güvenilirlik etkisinin hesaba katılması ile güvenilirliği fazla olan servislerin seçilme şansı arttırılmakta tersi durumdaki servislerin ise seçilme şansı azaltılmaktadır.

6. Yük Etkili Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması

Yük etkili algoritma bir önceki algoritmaya benzer şekilde oluşturulmuştur. Bir öncekinde farklı olarak güvenilirlik etkisi yerine yük etkisi kullanılmaktadır. Yeni ortalama işleme süresi hesaplanırken işleme süresine servis üzerindeki yük sayısı kadar ortalama işleme süresi ilave edilir. Bunun sebebi her bir isteğin yüzeysel bir hesaplamayla ortalama işleme süresi kadar zaman alacağının düşünülmesidir. Hesaplamada paralel çalışma ile beraber daha derinlemesine işleme süresi hesabı yapılmamasının sebebi karmaşıklığın artmasının istenmemesidir.

$$YeniOrtİşlSür = OrtİşlSür + OrtİşlSür * Yük \quad (3.6)$$

Algoritmada kullanılan bu basit formül ile üzerinde fazla yük olan servislerin seçilme şansını azaltarak yüklerinin bir kısmının diğer servislere dağıtılması sağlanmaktadır.

Algoritmalarda ilk göze çarpan nokta bazı seçim algoritmalarının kullanıcının gönderdiği değerlere ihtiyaç duymuyor oluşudur. Bu tip algoritmalar hem test

ortamında kıyaslama amaçlı oluşturmuştur hem de gerçek dünyada kullanıcının isteği ile çelişmek pahasına en iyi performansı vermeyi amaçlayan yaklaşımlar olarak değerlendirilebilir. Algoritmaların gerçekleştirilmeleri ile ilgili bilgilere ilerleyen aşamalarda değinilecektir.

BÖLÜM 4

4. SERVİS KALİTESİ EKLERİNİN BİR SERVİS YÜRÜTÜCÜ ÜZERİNDE GERÇEKLEŞTİRİMİ

Bu bölümde BPEL'de servis kalitesi parametrelerinin gerçekleştirimi, servis seçimi algoritmalarının gerçekleştirimi sunulacaktır.

Önerilen servis kalitesi ekleri, açık yazılım bir servis yürütücü olan ActiveBPEL üzerinde gerçekleştirilmiştir [30]. Yürütücü (engine) akademik ortamda araştırma için ve ticari alanda, belli sürümler ile ve bazı lisans şartlarını sağlamak kaydıyla, yaygın olarak kullanılmaktadır. Yürütücü açık kaynak kodludur ve Java ile gerçekleştirilmiştir.

İlerleyen kısımlarda yürütücünün mimarisi olduğu şekliyle ve eklenti yapıldığı haliyle anlatılacaktır. Yürütücünün derlenmesi için yapılaması gerekenler, derlendiği geliştirme ortamı ve derlenen yürütücünün çalışır hale getirilmesine kadar yapılması gerekenlerden bahsedilecektir. Ayrıca mevcut yapıda yapılan değişiklikler, sisteme dâhil edilen yeni kısımlar bir liste içersinde sunulacaktır. Son olarak servis seçim algoritmaların nasıl gerçekleştirildiği sözde kodlar yardımıyla açıklanacaktır.

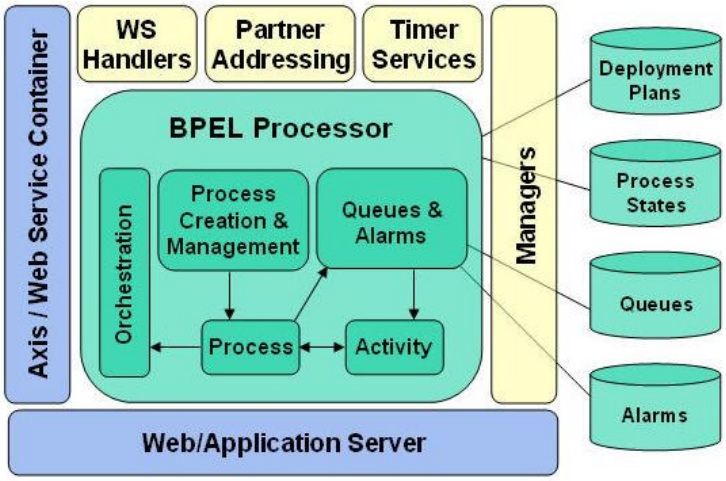
4.1 ActiveBPEL Yürütücüsü

ActiveBPEL web sayfasında yürütücünün belli başlı özellikleri ve mimarisi açıklanmaktadır [30]. Yürütücünün çalışmaya başlaması "Engine" sınıfı türünden bir nesnenin yaratılması ile sağlanır, bu aşamada bazı servisler de başlatılmaktadır. Başlatılan servisler arasında tüm çalışma süreci boyunca bazı görevlerin dağıtılması gerçekleştirilmiş olur.

Belli başlı servisleri şu şekilde sıralayabilmek mümkündür;

- Kuyruk Yönetimi (birden çok sürecin kuyruğa eklenmesi, sırayla veya diğer kriterlere göre işletilmeleri)
- Alarm ve Zamanlama Servisleri
- İş Yükleme Servisi

Yürütücünün tüm modüllerini gösteren ve mimarisine ilişkin yapıyı aktaran şekil aşağıda görülmektedir [30].

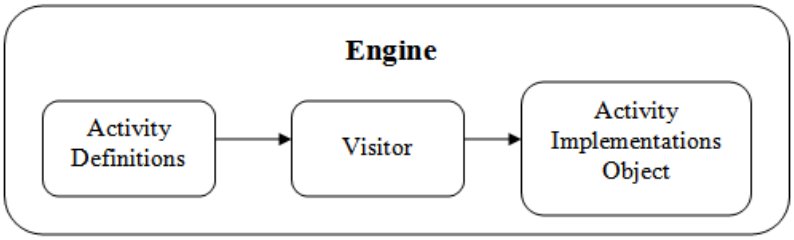


Şekil 4.1 ActiveBPEL Yürütücüsünün Mimarisi

Şekil 4.1'de görülen veritabanı bileşenleri kalıcı hafızayı oluşturmak üzere kullanılmaktadır. Yürütücünün kalıcı hafızaya ilişkin yönetici bir yazılımı vardır böylelikle yapılan işlemlerle ilgili belli başlı adımlar hafızada tutulabilmektedir [30].

BPEL süreçleri (process) bazı aktiviteler içerir. Yürütücünün süreç ile ilgili yaptığı ilk iş BPEL süreç tanımını okuyarak buna ilişkin Aktivite Tanım Nesneleri oluşturmaktır. Daha sonra ziyaretçi tasarım örüntüsü yardımıyla tanım nesnelerine ilişkin gerçekleme nesneleri yaratılır.

Bu sürecin daha iyi anlatılabilmesi için aşağıdaki şekle başvurulabilir (şekil 3.2). Buna göre süreçler yürütücüye ilk yüklendiğinde tanım nesneleri oluşturulur. Servislerin çağrılması adımında ise ziyaretçi tasarım örüntüsü ara süreç olarak kullanılır ve aktivitelere ilişkin gerçekleme nesneleri yaratılır [30].



Şekil 4.2. Aktivite Tanımlarından Aktivite Gerçekleme Nesnelerine

Aktivite sınıflarının sahip oldukları ortak metotlar şu şekildedir:

- isReadyToExecute(): Eğer aktivite süreci işletmeye hazır ise mantıksal doğru değeri döndürülür.
- execute(): Sürece ilişkin asıl görevin gerçekleştirildiği metoddur.
- objectCompleted(): Süreci tamamlayan aktivite tarafından çağrılır ve adımların başarıyla tamamlandığı belirtilir.
- terminate(): Ebeveyn süreç tarafından mevcut sürecin durdurulması amacıyla çağrılır. Hata durumunda ya da sürecin kesilmesinin gerektiği durumlarda bu metot çağrılır.

ActiveBPEL yürütücüsünün kullandığı belirli dosya yapıları vardır. Bunlardan bazıları standart olmakla beraber bazıları da bu yürütücüye özgüdür. Dosyalar ve dosya yapıları şu şekildedir:

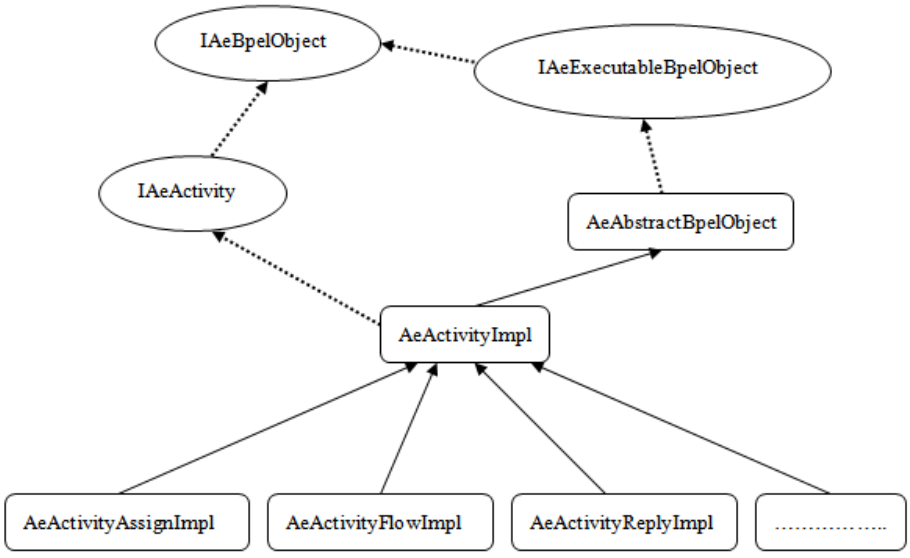
- aeEngineConfig.xml: ActiveBPEL yürütücüsünün her başlamasında bu dosyada belirtilen ayarlamalar kayıt altına alınır. Örneğin hata kaydı tutma özelliği açılıp kapatılabilir.

- *.bpr: WAR dosyası gibi bir arşiv dosyasıdır. İçeriğinde bulunanlar;
  - BPEL dosyası
  - Process Deployment Description dosyası
  - WSDL
  - Partner dosyası veya dosyaları

- *.pdd: “Process Deployment Description” Dosyası
    - ActiveBPEL yürütücüsüne özgü bir dosya formatıdır.
    - Partner bağlantılarını belirten ve BPEL sürecinin ihtiyaç duyacağı WSDL dosyalarını belirten bir ara yüz tanımlar.

- wsdlCatalog.xml: *.bpr dosyaları içindeki WSDL dosyalarının konumlarını tanımlar. META-INF klasörü altında bulunur

Tüm aktivitelere durumlar arasındaki geçişi sağlamak üzere ve olay ateşlemelerin (event firing) gerçekleştirmek üzere ortak bir metot kümesine sahiptir. Ortak metotların dışında her aktivitenin kendine has özellikleri sağlamak üzere sahip olduğu metotlar bulunmaktadır. Aktivite sınıflarına ilişkin hiyerarşi Şekil 4.3’te görülebilir. Aktivite sınıfları arasındaki hiyerarşide AeActivityImpl tüm gerçekleme sınıfları için taban sınıftır. IAeBPELObjectBPEL nesnesi gerçekleme için taban ara yüzdür. Diğer sınıflar aktivite tanım ve aktivite gerçekleştirme sınıflarıdır[30].



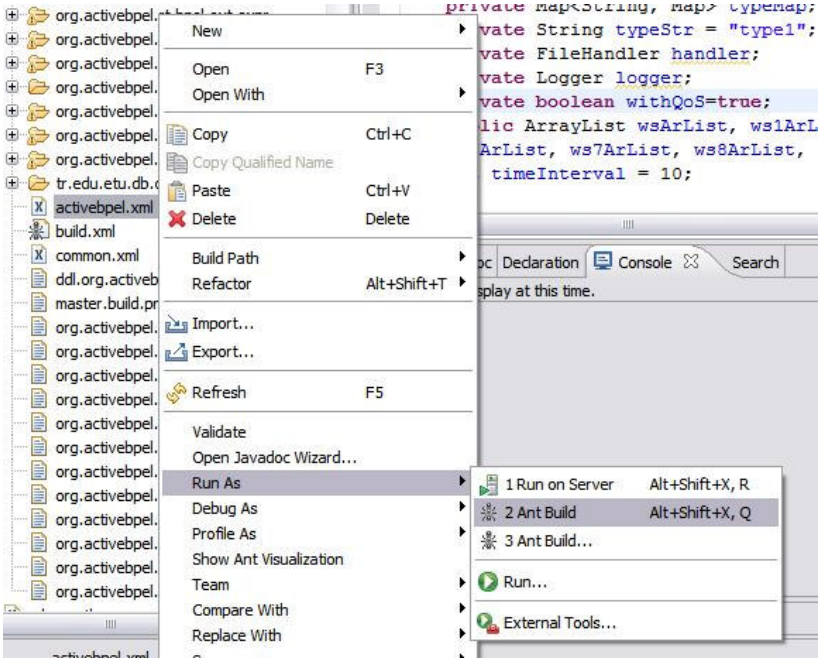
Şekil 4.3. Aktivite Sınıfları Arasındaki Hiyerarşi

4.2. Açık Kaynak Kodlu Yürütücünün Derlenmesi ve Çalıştırılması

ActiveBPEL Engine açık kaynak kodlu bir BPEL yürütücüsüdür. Kurulum dosyaları, kaynak kodları ve tüm yardımcı dokümantasyon yürütücünün web sayfasından indirilebilir [30]. Kodlarda değişiklik yapıldıktan sonra derleme aşamasının gerçekleştirilmesi ve ilgili *.jar dosyalarının oluşturulması "…\activeBPEL-2.0\doc\compile_engine.txt" dokümanında iki yolla açıklanmıştır;

1. Belirtilen birinci yolda konsol komut penceresi açılır ve "activeBPEL/projects" dizini altına gelindikten sonra "ant -f activeBPEL.xml compile.activeBPEL" komutu işletilir. Derleme adımında oluşturulan JAR ve WAR dosyaları "activeBPEL/dist" dizinine kendiliğinden eklenebilmektedir.

2. Derlemeye ilişkin belirtilen ikinci aşamada Eclipse bütünleşik geliştirme ortamı kullanarak derlemenin nasıl yapılabildiği açıklanmaktadır. Eclipse'de standart Java projesi yaratılır ve kaynak kodları olarak yürütücünün kaynak kodları referans gösterilir. İlgili kütüphaneler proje yoluna eklenir. Proje oluşturulduktan sonra "...\activeBPEL-2.0\projects\activeBPEL.xml" dosyası Eclipse'deki "Run As->Ant Build" seçenekleri kullanılarak çalıştırılır ve 30 saniye ile 90 saniye arasında değişen bir süre zarfında tüm kodların derlenmesi gerçekleştirilmiş olur.



Şekil 4.4. Derleme aşaması, activeBPEL.xml dosyası

Proje boyunca kullanılan derleme yöntemi olarak ikinci metot benimsenmektedir. Derlenen projenin çalıştırılması aşamasından hemen önce ilgili dosyalar uygulama sunucusuna yüklenmelidir. Bu adımı sağlamak için "...\activeBPEL-2.0\install.bat" dosyası çalıştırılmalıdır. Yükleme gerçekleştirildiğinde CATALINA_HOME altında bpr adlı bir dizin oluşturulduğu ve ilgili dosyaların buraya ve diğer ilgili dizinlere kopyalandığı görülmektedir.

Uygulama sunucusu olarak 5.5.17 sürüm numaralı Tomcat kullanılmaktadır [31]. Tomcat sunucu uygulaması ve JSP desteği olan bir web sunucusudur. Uygulama sunucusu yüklendikten sonra CATALINA_HOME adı altında Tomcat 5.5'in yüklendiği yol ortam değişkenlerine eklenmelidir. Bu değişken hem yürütücünün uygulama sunucusuna yüklenebilmesi hem de uygulama sunucusunun sağlıklı bir şekilde çalıştırılabilmesi ve diğer programlarla etkileşebilmesi için yaratılması gerekli olan bir değişkendir.

Bir önceki adımda bahsedilen derleme aşamasından sonra yürütücü dosyasının yüklenmesi için "...\activeBPEL-2.0\install.bat" dosyası çalıştırılmalıdır. Yükleme gerçekleştirildiğinde ortam değişkeninde CATALINA_HOME dizini olarak belirtilen dizinin altında bpr adlı bir klasör oluşturulduğu, ilgili dosyaların buraya ve diğer ilgili klasörlere kopyalanma işlemleri gerçekleştirilmiş olur.

4.3. Uzaktan Hata Ayıklamanın Etkinleştirilmesi ve İlgili Ayarlar

Yazılım geliştirirken en faydalı ve zaman tasarrufu sağlayan araçlardan biri hata ayıklama aracıdır (debug). Hata ayıklama araçları ile kodu çalışma esnasında adım adım izleyebilmek, değişkenlerin her aşamada aldığı değerleri görebilmek ve iş akışını etkin bir şekilde takip edebilmek mümkün olmaktadır. Günümüz yazılım geliştirme araçlarının büyük kısmında hata ayıklama araçları bulunmaktadır. Projeyi geliştirirken kullandığımız araçlardan biri olan Eclipse ortamında da hata ayıklama aracı bulunmaktadır ve bu projede sıklıkla kullanılmıştır [37].

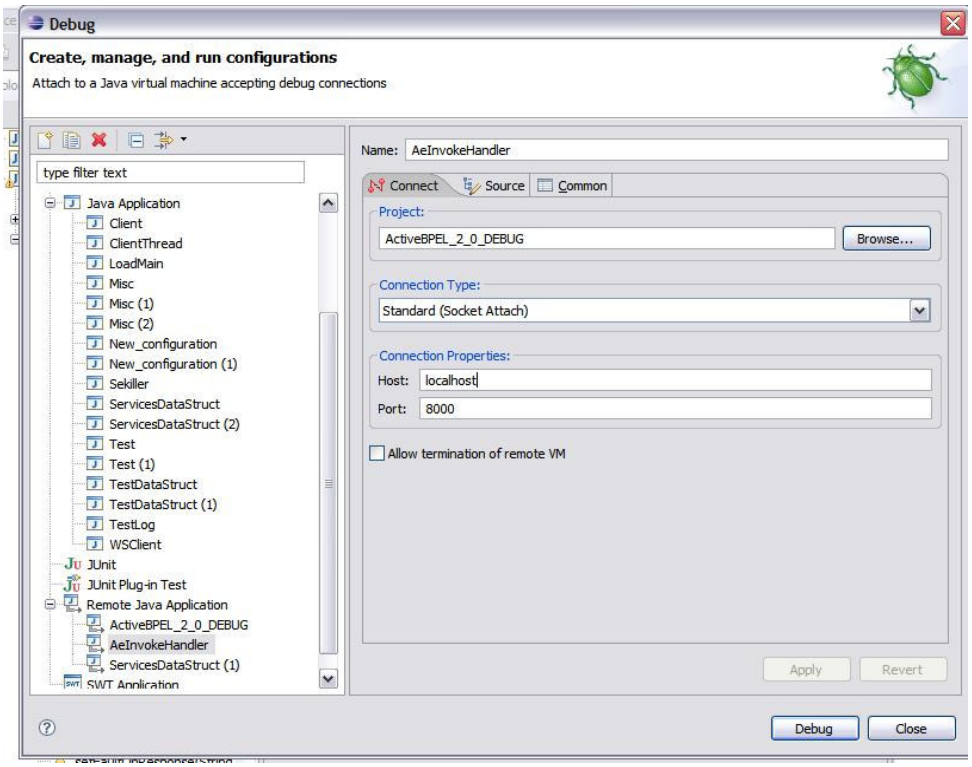
Projede gerçekleştirilen hata ayıklama metodu ile klasik hata ayıklama arasındaki en önemli fark derlenip paket haline getirilip uygulama sunucusuna yüklenen arşiv dosyaları ile yürütücünün kaynak kodlarını çalışma zamanında eşleştirebilmektir. Bu adımı geliştirme ortamının hata ayıklama aracı otomatik olarak gerçekleştirmektedir; ancak bazı ayarlamaların yapılması gereklidir.

Uygulama sunucusunun uzaktan hata ayıklama (remote debugging) desteğini sağlamak üzere ortam değişkenlerine iki adet değişken eklenmelidir. Bunlar;

- JPDA _ADRESS adında, "8000" değerine sahip ortam değişkeni
- JPDA_TRANSPORT adında, "dt_socket" değerine sahip ortam değişkeni

Tomcat'i hata ayıklama modunda başlatabilmek için komut penceresi açılmalı ve CATALINA_HOME\bin dizinine gelinmelidir ve "catalina jpda start" komutu işletilmesi sağlanmalıdır.

Eclipse'de ise "Debug->Remote Java Application" seçeneği ile yeni bir değişken yaratılıp hata ayıklama işlemi başlatılabilir. Hata ayıklamayı açıklamak üzere anlatılan adımlara ilişkin ekran görüntüsü aşağıdaki şekilde verilmektedir.



Şekil 4.5. Eclipse'de Hata Ayıklama (debug mode)

4.4. Yürütücünün Kaynak Kodunda Yapılan Değişiklikler ve Eklentiler

Tasarlanan sistemin gerçekleştirilmesi ActiveBPEL Yürütücüsü üzerinden yapılmaktadır. İstenen özellikleri gerçekleştirmek üzere yürütücünün kaynak koduna bazı değişiklikler ve eklentiler yapılmıştır; temelde iki adımdan oluşmaktadır. Bunlar ilerleyen bölümlerde bahsedilmektedir.

4.4.1. Servislerin Yüklenmesi Aşamasında Yapılan Değişiklikler ve Eklentiler

Servislerin yürütücü üzerinden sisteme yüklenmesi aşamasında yapılan değişiklikler ve eklentiler ayrıntıları ile EK A'da belirtilmektedir. Servisler yürütücüye yüklenirken BPEL dosyasında belirtilen sürece ilişkin veriler ve bilgiler yürütücü içersindeki JAVA tabanlı veri yapılarına alınmaktadır. Bu proje kapsamında BPEL diline yapılan servis kalitesi eklentisinin yürütücü içersinde JAVA yapılarına aktarılabilmesi için servis kalitesi parametrelerine ilişkin sınıf tanımlanmış ve mevcut yapıya bu parametreleri işletmeyi sağlamaya yönelik ekler yapılmıştır.

4.4.2. Servislerin İşletilmesi Aşamasında Yapılan Değişiklikler ve Eklentiler

Sisteme yüklenen servislerin işletilmesi aşamasına ilişkin kaynak kodunda yapılan değişiklikler Ek B'de belirtilmektedir. Kullanıcının servisleri çağırdığı adımda gerçek zamanlı servis seçimi yapılmaktadır. İsteğin yürütücüye iletilip servis seçiminin yapıldığı aşamaya kadar ziyaretçi tasarım örüntüsü yardımıyla tanım nesneleri kullanılarak gerçekleme nesneleri yaratılmaktadır. Bu nesneler vasıtasıyla kullanıcının istekte bulunduğu servis kalitesi parametreleri işlenmekte, JAVA veri yapılarına alınmakta ve servis seçim algoritmalarına giriş parametresi olarak gönderilmektedir.

4.4.3. Web Servisleri Seçim Algoritmalarının Gerçekleştirilmesi

Bu bölümde web servisleri seçim algoritmalarının yürütücü kaynak koduna bir eklenti olarak nasıl gerçeklendiklerine yönelik bilgiler aktarılmaktadır.

RSA hariç diğer algoritmalarda iki temel kısım dikkati çekmektedir. İlk kısımda servisin seçilmesi işlemi gerçekleştirilir ki RSA' da bu kısım bulunmaktadır. İkinci kısımda ise servise ilişkin gerçek zamanlı servis kalitesi değerinin güncellenmesi bulunmaktadır, bu kısmın gerçeklenmesine RSA için ihtiyaç olmamaktadır. Aşağıdaki sözde kod algoritması verilmektedir (Şekil 4.6).

```
// Gerçek Zamanlı Servis Kalitesi Verilerini Tutan Veri Yapısı
servisKalVY;

// Kullanıcı Tanımlı Servis Kalitesi Verilerini Tutan Veri Yapısı
kullaTanSerKalVY;

/* Kullanılabilir Servis Listesi */
servisListesi;

/* SERVİS SEÇİM KISIMLARI */
switch(algoritma)
{
    case: en_düşük_işleme_süresi_alg
    seçilenAdres = missa(servisKalVY, servisListesi);

    case: en_düşük_yük_alg
    {
       seçilenAdres = mysa( servisKalVY, servisListesi);
       servisYükünüArttır(seçilen Adres);
    }

    case: kullanıcı_isteğine_en_yakın_alg
    {
       seçilenAdres = kiissa( servisKalVY,
                      servisListesi, kullaTanSerKalVY);
    }

    case: güv_etkili_kullanıcı_isteği_alg
    {
       seçilenAdres = güv_et_kiissa( servisKalVY,
                      servisListesi, kullaTanSerKalVY);
    }


   case: yük_etkili_kullanıcı_isteği_alg
   {
      seçilenAdres = yük_et_kiissa(servisKalVY,
                     servisListesi, kullaTanSerKalVY);
   }
}
```

Şekil 4.6. Seçim Algoritmalarının Servis Seçimi Adımının Sözde Kodları

Servis seçimi adımı gerçekleştirildikten sonra servislerin gerçek zamanlı elde edilen servis kalitesi değerleri güncelleştirilir. Örneğin; MİSSA, KİİSSA, Yük Etkili KİİSSA ve Güvenilirlik Etkili KİİSSA' da servislerin işleme süresi, güvenilirlik değerleri kaydı tutulan ortalama değerine katılıp yeni ortalama işleme süresi hesaplanır ve bu değer kayıt edilir. MYSA' da ise ilgili servis başarı ile isteği gerçekleştirdiği için servisin yük değeri 1 azaltılır. Bu kısma ilişkin sözde kod ifadesi tablodaki gibidir.

```
/* Servis Kalitesi Listesi */
servisKalListesi;

/* Servisin Adresi */
servisAdresi;

/* Servisin Gerçek Zamanlı İşleme Süresi */
servisİşlSüresi;

/* SERVİS KALİTESİ GÜNCELLEŞTİRME KISIMLARI */
/* MİSSA, KİİSSA, Yük Etkili KİİSSA, Güvenilirlik Etkili
KİİSSA Servis İşleme Süresi Güncelleştirmeleri */
işlemeSüresiniGüncelle (servisAdresi, servisİşlSüresi,
servisKalListesi);

// MYSA ve Yük Etkili KİİSSA için Servis Yük Değerini
Azaltma Adımı
yükAzalt (servisAdresi, servisKalListesi);

/* Güvenilirlik Etkili KİİSSA için Güvenilirlik Değerinin
Hesaplanması ve Güncelleştirilmesi */
güvenilirlikGüncelle (servisAdresi, servisİşlSüresi,
servisKalListesi);
```

Şekil 4.7. Seçim Algoritmalarının Servis Kalitesi Güncelleştirme Sözde Kodları

BÖLÜM 5

5. PERFORMANS DEĞERLENDİRMESİ

5.1. Test Yazılımının Tasarımı ve Mimarisi

Bir projedeki en önemli aşamalardan bir tanesi test aşaması olarak değerlendirilmektedir. Bunun sebebi sisteme ilişkin başarımın ortaya konduğu veya eğer varsa hataların ortaya çıkarıldığı, düzeltildiği bir proje adımı olmasıdır. Projelerin testleri yapılırken öncelikli hedef kararlı bir test ortamının hazırlanmasıdır. Test ortamları genel bir bakış açısıyla üçe ayrılmaktadır, bunlar; gerçek dünyadaki olayların bir benzetim ortamında gerçekleştirilmesi ile oluşan ortamlar, benzetime ihtiyaç duymaksızın her bileşenin gerçek dünyada çalışır durumda tasarlandığı ortamlar veya her ikisinin de kullanıldığı ortamlar olarak ön plana çıkmaktadır. Proje kapsamında ağırlıklı olarak gerçek dünyadan bileşenler kullanılmakla birlikte gerçek dünyaya yakın olarak tasarlanan bazı bileşenler, örneğin eş zamanlı istemci isteklerinin üretilmesi, benzetimden faydalanılarak ortaya konulmuştur. Proje için oluşturulan test ortamında beş adet bileşen bulunmaktadır. Bunlar; istemci, BPEL yürütücüsü ve üç adet Java tabanlı web servisleridir.

- Tasarlanan istemci bileşeni; eş zamanlı BPEL istekleri yaratmak üzere eş zamanlı izlek yapısını kullanmaktadır. İstemci çalıştırıldığında bir yapılanış dosyası okumaktadır. Bu dosyada istemcinin yaratacağı isteklerin hangi zaman aralıklarında, nasıl bir zaman aralığı artımında gerçekleşeceği ve her bir zaman aralığı için toplam kaç isteğin üretileceği belirtilmektedir.

    - Testlerde kullanılan zaman aralıkları başlangıç 10 milisaniye, bitiş 100 milisaniye olarak belirlenmiştir. Ardışık zaman aralıkları arasındaki fark 10 milisaniye olarak hesaplanmıştır. Her zaman aralığında ise 3000 çağrı

yapılmaktadır. Bu ayarlamalar, yapılanış dosyasının içeriği uygun biçimde değiştirilerek yeniden düzenlenebilmektedir.

- Herhangi iki istek arasındaki zaman aralığının hesaplanmasında Java tabanlı simJava [32] benzetim paketindeki negatif üstel dağılım fonksiyonu kullanılmaktadır. Benzetim paketindeki bu fonksiyonun girdi değeri bir önceki maddede değinilmiş olan ve yapılanış dosyasından okunan ortalama zaman aralığı değerleridir.

- Ortamda bulunabilecek aktif eş zamanlı en fazla izlek sayısı sınırlandırılabilmektedir, böyle bir yöntem izlenmesinin sebebi tüm testlerin aynı şartlar altında gerçekleştirilebilmesi ve fazla isteğin olduğu durumda sistemin dar boğaza takılıp cevap veremez hale gelmesinin önüne geçilmesidir. Çalışma anında en fazla izlek sayısına ulaşıldığı taktirde isteklerin cevaplanması beklenir, cevaplanan her bir izlek yerine yeni bir izlek ortama girebilir. Proje testinde 5 ve 50 olmak üzere iki adet maksimum izlek sayısı kullanılmıştır.

- İstemcinin diğer bir özelliği test verilerini toplayıp dosyalara yazabilmesidir. Toplanan veriler servislerin cevaplama süreleri, hangi servisin kaç kere çağrıldığı ve servislerin isteklere ürettikleri cevaplar gibi verilerdir.

İstemci yazılımı test verilerini toplayan ve sistemin kararlı çalışmasında önemli etkisi olan bir sistem bileşenidir. Test ortamının tasarlanırken 4 adet Java sınıfı ortaya çıkmıştır. Bu sınıflar; Client, ClientThread, CallingThread, Results sınıflarıdır. Bu sınıfların görev ve işlevleri aşağıdaki gibi özetlenebilir;

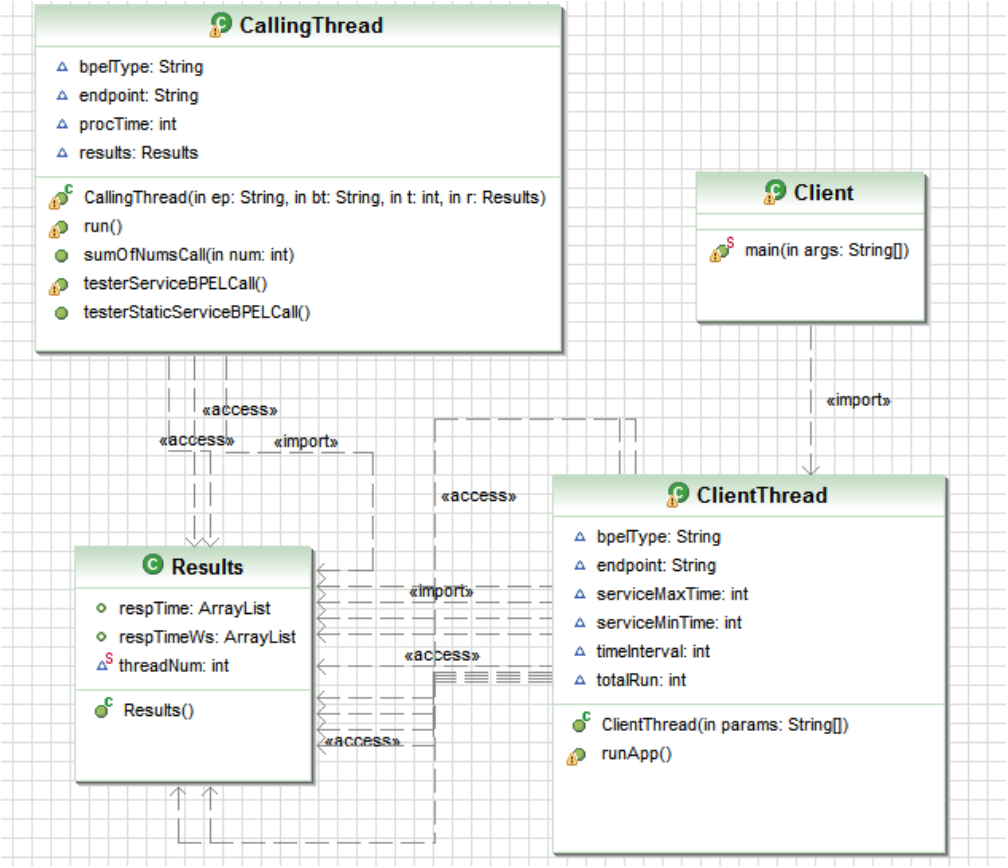
- Client sınıfı konfigürasyon dosyasını okumakta ve ilgili parametreleri ClientThread sınıfı türünden nesneye göndermektedir. Başlangıç-bitiş-artış zaman aralığı parametrelerine göre bir döngü içersinde yeni ClientThread tipinden nesneler yaratılıp parametreler bu nesnelere iletilmektedir.

- ClientThread aldığı zaman aralığı parametresine göre negatif üstel dağılıma göre bir bekleme süresi elde etmektedir. Bu süre istemcilerin isteklerine başlamaları arasında geçen süreyi belirtmektedir. Bulunulan zaman aralığında kaç çağrı yapılacağını belirten parametreye göre döngü içersinde bu adımlar tekrarlanır.

- CallingThread sınıfı isteğin yapıldığı ve servis çağrısının gerçekleştirildiği sınıftır.

- Results sınıfı servislerden gelen cevap verilerinin ve gerçek zamanlı olarak aktif olan istemci sayısını kaydeden sınıftır.

Bu sınıflara ilişkin UML diyagramları ve sınıfların birbirleri ile bağımlılık ilişkileri aşağıdaki şekilde görülebilir.



Şekil 5.1. Client, ClientThread, CallingThread, Results sınıflarına ilişkin UML ve Bağımlılık Diyagramları

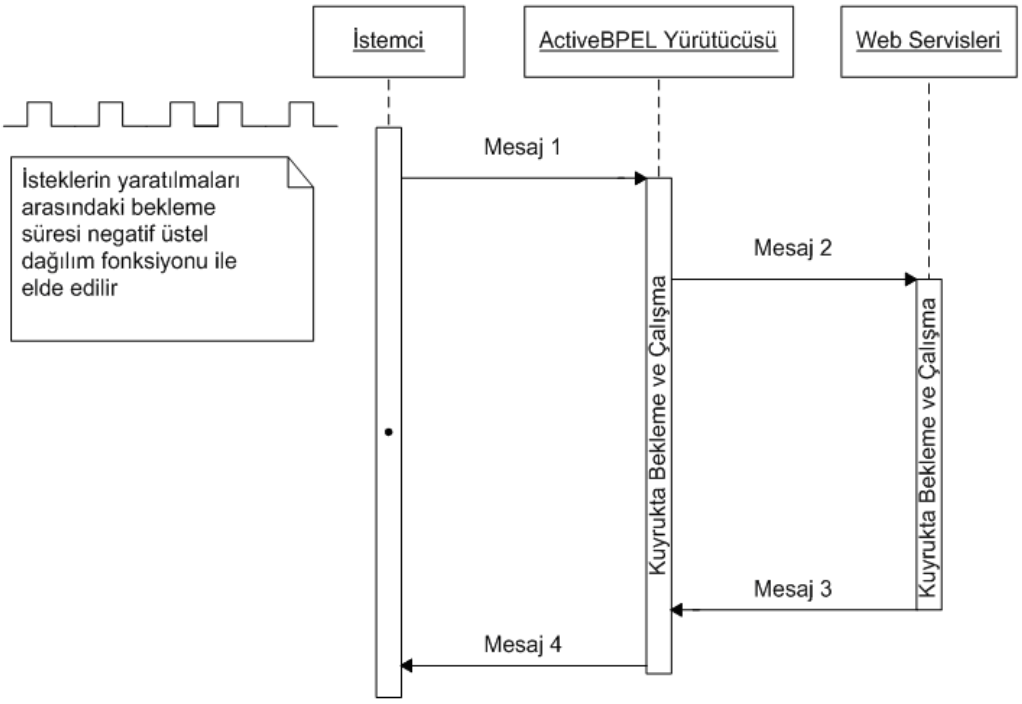
- ActiveBPEL Yürütücüsü (2.0 numaralı sürümü) [30].
    - Daha önceki bölümlerde belirtildiği gibi yürütücü kaynak kodu, servislerin servis kalitesi parametrelerine göre dinamik olarak seçilmesine olanak sağlayacak şekilde değişiklikler ve eklentiler ile yeniden derlenip çalışır hale getirilmiştir.
    - Testin ilk aşaması olarak tek bir servis tipini içeren birleşik servis yapısı (BPEL dosyası) yürütücüye yüklenmiştir.
    - Yürütücü ayrıca kullanıcının gönderdiği değerlere göre servis çağrısının başarılı olup olmadığı istatistiğini tutmak üzere test verilerini dosyalara kaydetmektedir.

- Web Servisler standartlaşmış ve standartlaşmakta olan dağıtık yazılım bileşenleridir ve bu sebeple hangi dilde gerçekleştirildikleri önem taşımamaktadır. Proje kapsamında test amaçlı üç adet Java tabanlı web servis gerçeklenmiştir (JAX-RPC tabanlı web servisler). Bu servisler işlevsellik açısından eş değerlidirler. Üç servis de MYSQL'in "world" veritabanını kullanarak [33] bir takım sorgular gerçekleştirmektedirler ve bu yolla diske bağlı servis içi işleme süresi oluşturulmaktadır.
    - Servislerin ortalama 50 milisaniye, 250 milisaniye ve 300 milisaniye servis içi işleme süreleri bulunmaktadır. Yalnız şunu belirtmekte yarar vardır; bu süreler isteklerin oluşturulma zaman aralıklarına bağlı olarak farklılık göstermektedir. Bu önemli nokta test sonuçlarının sunulduğu bölümde irdelenmektedir.

Test ortamını oluşturan bileşenlerin tanımlaması yapıldıktan sonra test ortamının çalıştırılması aşamasında web servisler ve yürütücünün üzerinde çalıştığı uygulama sunucusu başlatılmalıdır. İstemcinin düzgün başlatılabilmesi dört parametreye ihtiyaç vardır. Bunlar yürütücünün çalıştığı makinenin ip numarası, çağrılması istenen BPEL servisinin adı, kullanıcının tanımladığı işleme süresi ortalaması ve varyans değeri. Örneğin; şu şekilde bir komut işletilirse istemci başarı ile çalıştırılmış olur.

- java src.Client 10.10.228.218 testerService 175 1625
    - java; Java dosyalarını çalıştırmak için gerekli programa komuttur
    - src.Client java programına gönderilen bir parametredir.
    - 10.10.228.218; yürütücünün çalıştığı makinenin ip numarasıdır.
    - testerService; BPEL servisinin adıdır.
    - 175 ve 1625 ise işleme süresi ortalaması ve varyansıdır.

Test ortamının çalışmasına ilişkin yapı şekil 5.2.'den incelenebilir.



Şekil 5.2.- Test Ortamının Çalışması

5.2. Test Ortamında Kullanılan İstatistiksel Fonksiyonlar

Projede iki çeşit istatistiksel fonksiyon kullanılmaktadır. Bunlar; normal ve negatif üstel dağılım fonksiyonlarıdır.

Normal Dağılım Gaus dağılımı olarak bilinen dağılım çeşididir. Ortalama ve varyans değerine göre değişiklik göstermektedir. Ortalamanın sıfır, varyansın 1 olduğu durum çan eğrisi olarak adlandırılmaktadır; çünkü olasılık yoğunluk grafiği çanı andırmaktadır. Bir X; μ ortalaması ve σ^2 varyansı ile normal dağılım gösteren bir rastgele değişken ise şu şekilde ifade edilir [34].

$$X \sim N(\mu, \sigma^2) \tag{5.1}$$

Normal dağılımın olasılık yoğunluk fonksiyonu ise şu ifade ile verilmektedir[24].

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sigma}\varphi\left(\frac{x-\mu}{\sigma}\right), \tag{5.2}$$

μ beklenen değer, σ standart sapma olarak alındığında yoğunluk fonksiyonu ifadesi şu şekildedir[24];

$$\varphi(x) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}x^2} \tag{5.3}$$

Bir Java benzetim paketi olan SimJava'dan [32] faydalanılarak normal dağılımı kullanan bir rastgele sayı üreteci kullanılmıştır ve kullanıcının tanımladığı işleme süreleri değerleri hesaplanmıştır. Normal dağılımı kullanan metodun tercih edilmesini sebebi ise normal dağılımın doğaya en iyi uyum sağlayan dağılımlardan biri olmasıdır.

Jansen'in "Simulation Programming" ders notlarında [35] negatif üstel dağılıma ilişkin şu örnek kullanılmaktadır. Eğer bir olay rastgele oluyorsa ve kendinden önceki veya sonraki olaydan bağımsız meydana geliyorsa olaylar arasındaki sürenin dağılımı negatif üstel dağılım olarak tanımlanabilir. Olasılık yoğunluk fonksiyonu şu şekilde ifade edilmektedir.

$$f(t) = a \exp(-at) \tag{5.4}$$

Yine Java simülasyon paketi olan SimJava [32] kullanılarak negatif üstel dağılımını kullanan bir rastgele sayı üreteci sisteme dahil edilip ve kullanıcının istekleri arasındaki sürenin hesaplaması yapılmıştır. Negatif üstel dağılımı kullanan metodun tercih edilmesini sebebi ise kullanıcı isteklerinin ayrık zamanlı, rastgele ve birbirinden bağımsız olaylar olmasıdır; bu karakteristik negatif üstel dağılımla uyumludur.

5.3. Performans Kriterleri

Test yazılımının test ortamını sağlama görevinden ayrı olarak başlıca özelliklerinden biri test verilerini toplama görevidir. Verilerin sağlıklı toplanması değerlendirme sürecinin de doğrulunu etkileyecek bir husus olarak ön plana çıkmaktadır.

Daha önceki bölümde de değinildiği üzere istemci modülü servislerin cevaplama sürelerini, servislerin cevaplarını ve hangi servisin kaç defa çağrıldığına ilişkin verileri toplamaktadır. Cevaplama süresine ilişkin verilerin toplanmasının sebebi sistemin istek yapan kullanıcıya yönelik performansının ölçülebilirliğini sağlamaktır. Sonuç değerlendirilmesinde de irdelenen başlıca test verisi cevaplama süresi olarak ortaya çıkmaktadır. Servislerin çağrılma sayıları da proje kapsamında önem taşımaktadır; çünkü bu sayılar hem servislerin dinamik olarak çağrılmasına yönelik bir göstergedir hem de servislerin işleme süreleri farklılık arz ettiği için cevaplama süresine etki eden başlıca faktörlerdendir.

Yürütücü modülünde ise kullanıcının gönderdiği servis kalitesi değerlerine göre servis isteğine başarılı bir cevap gerçekleşip gerçekleşmediğine yönelik veriler toplanmaktadır. Örneğin; kullanıcının 200ms'lik bir işleme süresi için istekte bulunmuş olduğu durumda servislerin işleme süreleri sırasıyla 185ms, 215ms ve 400ms ise ilk servis başarılı diğer iki servis ise başarısız kabul edilmektedir. Ayrıca servislerin gerçek zamanlı yük oranları, işleme süreleri, seçilen servislerin adresleri,

güvenilirlik yüzdelerine ilişkin veriler toplanarak hata ayıklama aşamasının sağlıklı yürütülmesi sağlanmıştır.

Web servis modülünde ise her çağrı için servisler kendi servis içi işleme sürelerini hesaplarlar ve servis cevabı içersine bir zaman damgası olarak eklerler. Bu veri test sonuçlarında birebir bulunmamakla beraber, sistemin doğru çalışılırlığını göstermek ve yorumlamalara katkıda bulunmak için yer almıştır.

Yaratılan test ortamının çalıştırılması ve projeye ilişkin test verilerinin elde edilmesinden sonra en önemli nokta bu verilerin değerlendirilmesidir. Test verilerinden elde edilen grafiklere ve verilerin yorumlanmasına bir sonraki bölümde değinilmektedir.

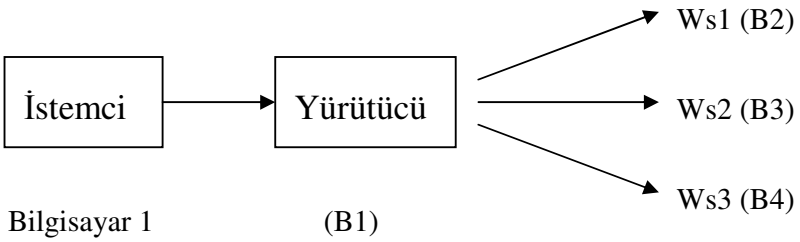
5.3.1. Sistem Özellikleri ve İstemci Tarafı Test Ayarlamaları (Configurations)

İstemci ve BPEL yürütücüsü aynı makineye yüklenmiştir ve makinenin sistem özellikleri şu şekildedir;

- Core Processor 4400+ 2.21Ghz
- 2.00 GB RAM
- Windows Server 2003 Enterprise Edition
- JDK 1.6.0
- ActiveBPEL Engine 2.0
- Tomcat 5.5.17

Üç adet Java tabanlı web servis farklı makinelere yüklenmiştir ve makinelerin eş olan sistem özellikleri şu şekildedir;

- Intel Pentium 1.60 GHz
- 512 MB RAM
- Windows XP Home Edition
- JDK 1.6.0
- Tomcat 5.5.17
- MySQL 5.0



Şekil 5.3.- Test Ortamı

Test ayarlamalarına bakıldığında istemci tarafına ilişkin ayarlamalar ön plana çıkmaktadır ve şu şekilde belirtilmektedir;

İstemci tarafı;

- Zaman aralıkları
  - Başlangıç 10 milisaniye
  - Bitiş 100 milisaniye

- Ardışık zaman aralıkları arasındaki artış
  - 10 milisaniye
- Her zaman aralığı için çağrı sayısı
  - 3000 çağrı
- İstemcinin işleme süresi isteklerini benzetmek için normal dağılım fonksiyonu kullanılmaktadır. Normal dağılım fonksiyonu için girdi değerleri;
  - En az işleme süresine sahip servisin yaklaşık değeri 50ms

- En fazla işleme süresine sahip servisin yaklaşık değeri 300ms
- Ortalama değer;
  - $\frac{50ms + 300ms}{2} = 175ms$
- Değişim
  - $\frac{(175-50)^2 + (175-300)^2}{2} = 15625$

- İki farklı test için ortamdaki en fazla aktif izlek (thread) sayısı 5 ve 50 olarak ayarlanmaktadır.

5.3.2. Sonuçlar

Ortaya konan tasarımı çalışır hale getirmek için daha önce tanıtıldığı şekilde bir gerçekleme yapılmıştır. Tasarımın ve yapılan gerçeklemenin verimliliğini sınamak için bazı zorlama testleri gerçekleştirilmiştir ve testlerin sonuçları ilerleyen kısımlarda açıklanmaktadır. Zorlama testleri sistemlerin kararlı bir şekilde çalışıp çalışmadığını ortaya koyabilen test çeşitleri olarak bilinmektedir. Bu proje kapsamında bu tip testlerin gerçekleştirilmesinin sebebi ise; sistemin sınırlarının uç noktalarında test edebildiğini ve yapılan tasarımların-eklentilerin başarımının bu yolla ortaya konulabildiğini göstermektir. Ayrıca bilgisayar sistemlerinin geleneksel olarak karşılaştırılmasında en kötü, ortalama ve en iyi durumların ele alındığı göz önüne alınması da tez projesi kapsamında zaman açısından en kötü-kötüye yakın durumdan ortalama duruma doğru testlerin yapılması savını desteklemektedir. En iyi duruma ilişkin de bazı testler yapılmakla birlikte bu durumda sistemin başarı göstereceği açık olduğu için bu yöntem kullanılmamıştır. Bu kısımda değinilenler bir örnekle şu şekilde özetlenebilir;

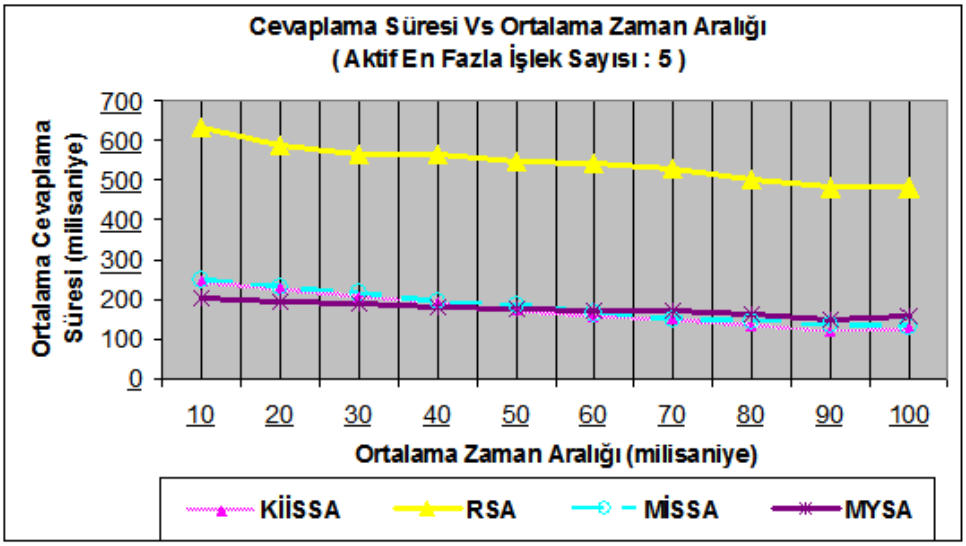
Örneğin 10 adet web servisin en düşük işleme süresine göre dinamik olarak seçiminin yapılacağı bir sistemde, zaman açısından sistemin iyi durumunda (istekler arası zaman aralığı kuyrukta bekleme oluşturmayacak kadar fazla) en düşük işleme süresine sahip olan servisin en iyi performansı göstereceği açıktır. Ancak istekler

arası süre azaltıldıkça gözlemlenmektedir ki yürütücü kısmında süreçler işletilirken, uygulama sunucusu kısmında servisler yürütülürken ve veri tabanı yöneticisi kısmında tablolardan veri çekerken isteklerin kuyrukta beklemeleri gündeme gelmektedir; bu durumda ise en düşük işleme süresine sahip bir servisin cevaplama süresi çok fazla artabilmekte neticede ise cevaplama süresi açısından en kötü performansı ortaya koyabilmektedir. Bütün bu açıklamalar ve verilen örnek açıkça göstermektedir ki zorlama testi mevcut sistemin başarımını ölçmek için en iyi test yöntemi durumundadır.

Testler iki aşamadan oluşmaktadır. İlk aşamada ortamda bir anda bulunabilecek en fazla aktif işlek sayısı 5, ikinci aşamada ise 50 olarak belirtilmektedir. İki ayrı testlerin sonuçları ve bu sonuçlara ilişkin değerlendirmeler ilerleyen bölümlerde açıklanmaktadır.

5.3.2.1. Birinci Aşama Test Sonuçları

Daha önce de belirtildiği gibi birinci aşama test ortamında bir anda bulunabilecek en fazla aktif işlek sayısı 5 ile sınırlanmaktadır. Her zaman aralığı için 3000'er istek yapılmış ve cevaplama süreleri ortalamaları hesaplanmıştır. Buna göre dört algoritmaya ilişkin test verileri Şekil 4.3' ten görülebilir.



KİİSSA Kullanıcı İsteğine En Yakın İşleme Süresine Göre Seçim Algoritması, RSA Rastgele Servis Seçim Algoritması,MİSSA Minimum İşleme Süresine Göre Seçim Algoritması, MYSA Minimum Yüke Göre Seçim Algoritması

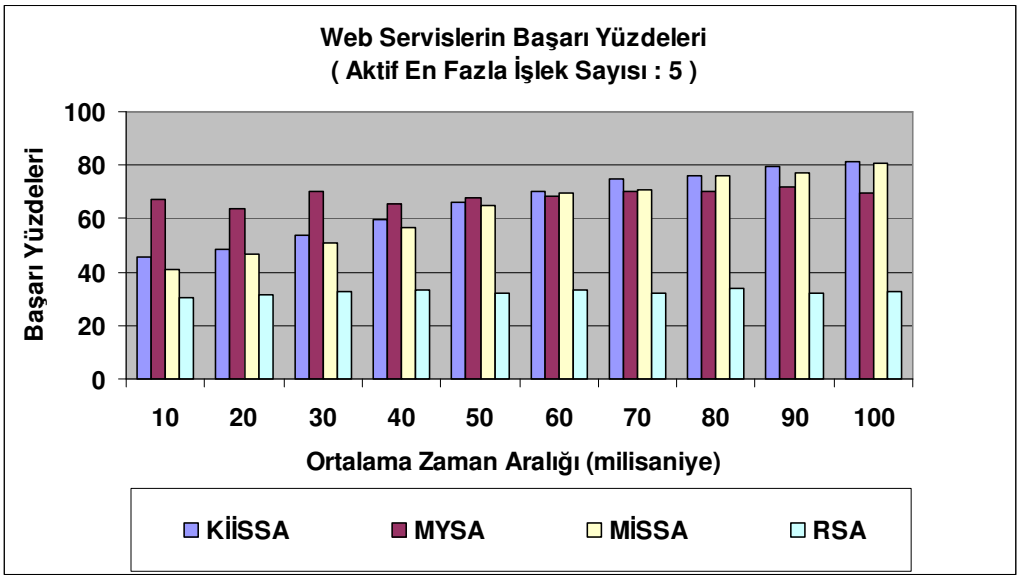
Şekil 5.4 Algoritmaların Ortalama Cevaplama Süresine göre karşılaştırılması

İlk bakışta grafikten görülebilecek bilimsel gözlem KİİSSA, MYSA, MİSSA algoritmalarının RSA algoritmasına göre daha iyi performans gösterdiğidir. Başarı gösteren üç algoritmanın en başarılı yani işleme süresi performansı en yüksek servise yönelmesi beklendiği için bu sonuç teorik olarak da beklenen bir sonuçtur.

Test verileri ve grafikten elde edilebilecek diğer bir sonuç ise dört servis seçim algoritması için de geçerli olan ortalama zaman aralıkları azaldıkça servislerin ortalama cevaplama sürelerinin artması durumudur. Böyle bir sonucun en önemli sebebi düşük zaman aralığında çok yakın aralıklarla gelen isteklerin olmasıdır neticesinde bir isteğe cevabı verilmeden bir veya birden çok yeni isteğin gelmesi kaçınılmaz olmakta bu durumda gerek yürütücü gerekse web servislerinin çalıştığı modülde isteklerin paralel işleme ile değerlendirilmesi kaçınılmaz olmaktadır. Bütün bu anlatılan zincirleme olaylar neticesinde servislerin cevaplama sürelerinin artışı gerçekleşmektedir. Öngörülen ve grafikte gözlenen budur, sonuçlar birbirleriyle uyumludur.

Cevaplama sürelerinden başka elde edilen diğer bir test veri kümesi ise servislerin başarı yüzdeleridir. Servisin gerçek zamanlı işleme süresi değeri kullanıcının istekte

bulunduğu işleme süresinin altında ise servis başarılı olmuş demektir, başarı yüzdesi ise 3000 çağrı içinde kaç isteğin başarılı olduğunun yüzde olarak ifadesini belirtmektedir. Aşağıda servislerin işleme sürelerine göre başarı yüzdeleri verilmektedir.



Şekil 5.5. Algoritmaların Servis Başarı Yüzdelerine Göre Karşılaştırılması

Ortalama zaman aralıklarına göre servislerin başarı yüzdeleri incelendiğinde düşük zaman aralıklarında (10 milisaniyeden 50 milisaniyeye kadar) MYSA algoritmasının hem kararlı hem de en yüksek başarı yüzdesine sahip olduğu görülmektedir. Sonraki zaman aralıklarında ise üstünlük az bir farkla da olsa KİİSSA ve MİSSA algoritmalarına geçmektedir. MYSA'dan başka servis başarı oranı açısından diğer bir kararlı algoritma RSA'dır. RSA'nın diğer tüm algoritmalar arasında tüm ortalama zaman aralıkların içinde en kötü başarı yüzdesine sahip; ancak bir o kadar da kararlı bir karakteristik gösterdiği gözlemlenmektedir.

İrdelenen sonuçların sebepleri maddeler halinde şu şekilde özetlenebilir;

- MİSSA ve KİİSSA algoritmaları en başarılı, başka bir söyleyişle en düşük işleme süresine sahip, servisi seçmeye eğilimlidirler. Bu algoritmalara tek bir servise yönelmeleri sebebiyle servis üzerinde paralel çalışma artar, bu artış da işleme süresini, dolaylı olarak da cevaplama süresini arttırmaktadır. Böylelikle servis yükünün fazla olduğu düşük zaman aralıklarında başarı yüzdesi düşmektedir. Yüksek zaman aralıklarında ise (50 milisaniyeden daha yüksek) paralel işleme giderek azaldığı için başarı yüzdelerinde belirgin bir artış gözlemlenmektedir.

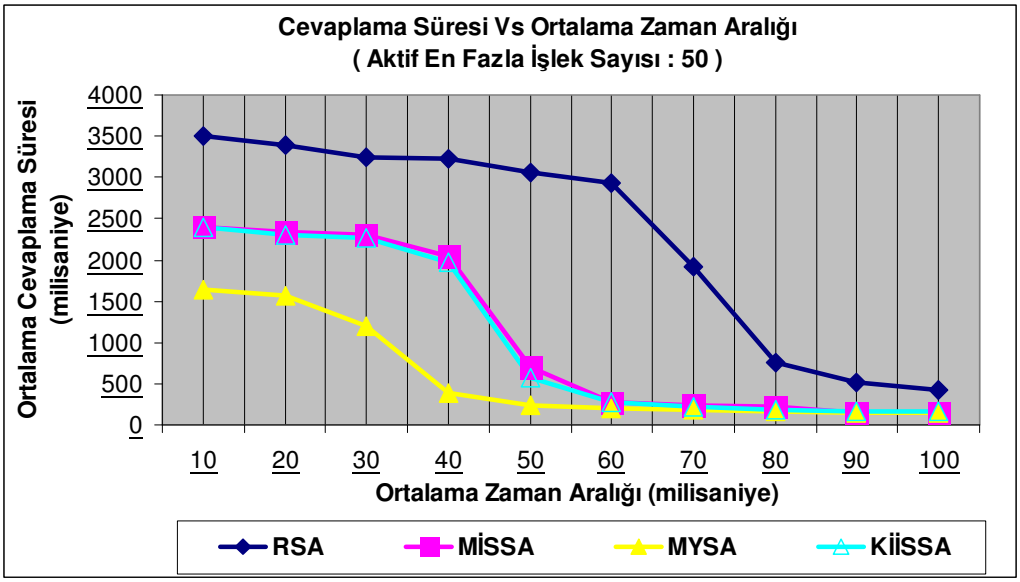
- MYSA; MİSSA ve KİİSAA'nın tersine istek yüklerini 3 servis arasında olabildiğince eşit dağıtmaya ve minimum yüke sahip servisi seçmeye yönelik çalışmaktadır. En hızlı servis çoğunlukla en az yüke sahip olan servistir. En düşük işleme süresine sahip servisin az yük altında seçilmesi ise başarı yüzdesini arttırmaktadır.

- RSA ile istek sayıları bakımından servisler arası mükemmel dağılım gerçekleştirilmektedir; ancak bu algoritmanın da bazı düşük ortalama zaman değerleri hariç iyi performans göstermediği gözükmektedir. Bu durumda istenen sonucu vermeyeceği anlaşılmaktadır.

İlk aşama test sonuçları bazı yorumlar yapabilmemize olanak sağlasa da 5 istemcili yapının zorlama testi için yeterli olmayacağı açıktır. Bu aşamadaki testler ile algoritmaların başarılı çalışmaları konusunda bir ön izlenim oluşsa da istemci sayısının 50'ye çıkartıldığı ikinci aşama testlerine ihtiyaç duyulmuştur, bu testler ile daha açık sonuçlar ve algoritma karşılaşmaları elde edilmiştir. Bir sonraki bölümde bu test sonuçlarına değinilecektir.

5.3.2.2. İkinci Aşama Test Sonuçları

İkinci aşama test ortamında bir anda bulunabilecek en fazla aktif işlek sayısı 50 ile sınırlanmaktadır. Birinci aşama testlerine benzer şekilde her zaman aralığı için 3000'er istek yapılmış ve cevaplama süreleri ortalamaları hesaplanmıştır. Buna göre dört algoritmaya ilişkin ortalama cevaplama süreleri aşağıdaki grafikten görülebilmektedir.



Şekil 5.6. Algoritmaların Ortalama Cevaplama Süresine göre karşılaştırılması

İlk aşama testlerinden farklı olarak algoritmalara ilişkin ortalama cevaplama sürelerinin farklılaştığı gözlemlenmektedir. Bu durum; aktif en fazla 50 işlek sayısı ile zorlama testinin başarı ile gerçekleştiğinin bir kanıtıdır. Grafikten de gözüktüğü üzere en iyi performansı MYSA göstermektedir. KİİSSA ve MİSSA birbirlerine çok yakın performans göstermekte, 5 istemcili çalışmada olduğu gibi burada da RSA bu test çalışması için en kötü durumu oluşturmaktadır. Bir önceki aşamada yapılan ortalama zaman süresi azaldıkça cevaplama süresinin artacağı çıkarımları bu aşamada da yapılabilir. Sonuçlar değerlendirilirken bazı ek verilerle çıkarımları

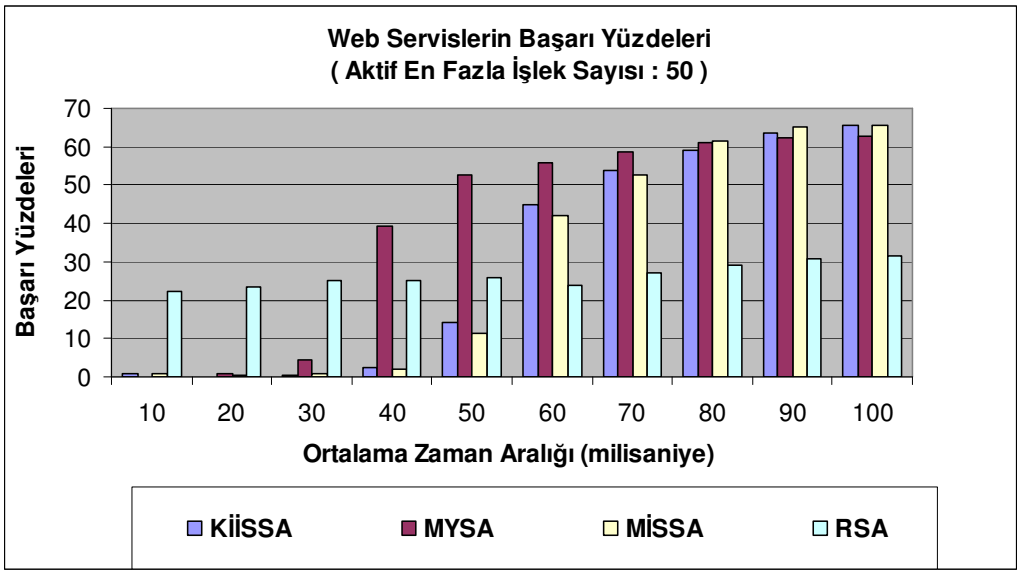
desteklemek gerekmektedir. Bu doğrultuda ortalama zaman aralığının 10ms olduğu durum için servislere yapılan çağrı sayıları irdelenmektedir.

Başarılı istek sayılarına bakıldığında en dengeli dağılımın RSA ile olduğu gözlemlenmektedir. 3000 çağrı neredeyse 1000'erlik bölümlerle servisler arasında eş dağılmaktadır. Eş dağılım gerçek zamanlı bir eş dağılım değil toplam çağrı üzerinden bir eş dağılımdır. Örneğin t anında web servislerin yükleri sırası ile 10-20-5 olabilir; ancak 3000 çağrı sonucunda 938-1062-1000 gibi 3000 çağrı üzerinden yaklaşık olarak eşit bir dağılım gözükmesi beklenmekte ve gözlemlenmektedir. O halde RSA ile gerçekleştirilen durumun ortalama durum olduğu söylenebilir. Bu takdirde en iyi durum hep birinci servisin seçildiği, en kötü durum ise hep üçüncü servisin seçildiği durumdur. KİİSSA ve MİSSA için başarılı istek sayılarının dağılımına bakıldığında yaklaşık olarak aynı oldukları ve en büyük payın birinci yani en hızlı serviste olduğu gözükmektedir. Bu seçim oranı beklenen bir durumdur. İki algoritma ile en iyi duruma en yakın durumun yakalandığı öngörülmekle birlikte elde edilen sonuçlardan en iyi durumun MYSA tarafından yakalandığı gözükmektedir. Bunun en önemli sebebi MYSA' da dağılım açısından üstünlük yine birinci servistedir; ancak ortalama cevaplama süresini arttıran birinci servisteki yükün bir kısmı diğer iki servis arasında paylaştırılmış böylelikle KİİSSA ve MİSSA' ya göre daha başarılı bir sonuç elde edilmiştir. Açıklanan başarılı istek sayısı dağılımlarına ilişkin veriler Çizelge 5.1' de belirtilmektedir.

Çizelge 5.1. KİİSSA, MİSSA ve MYSA için Başarılı İstek Sayılarının Dağılımı

	Başarılı İstek Sayısı			
Ortalama Zaman Aralığı(10ms)	MİSSA	KİİSSA	MYSA	RSA
Web Servis – 1	2875	2937	2017	1001
Web Servis– 2	50	13	487	999
Web Servis – 3	75	50	496	1000
Toplam	3000	3000	3000	3000

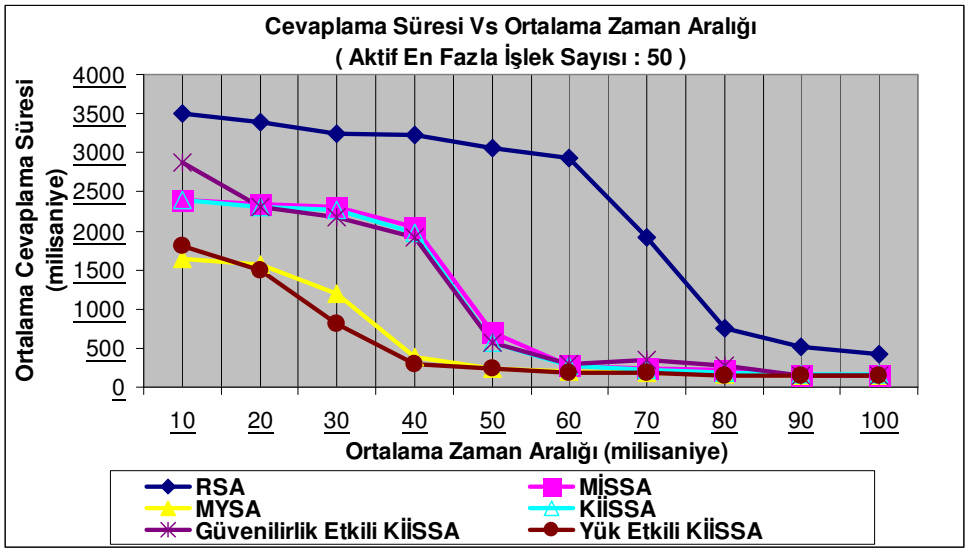
Aktif en fazla izlek sayısının 50 olduğu ikinci aşama testlerde servislerin başarı yüzdelerine ilişkin grafik Şekil 5.7' de verilmektedir.



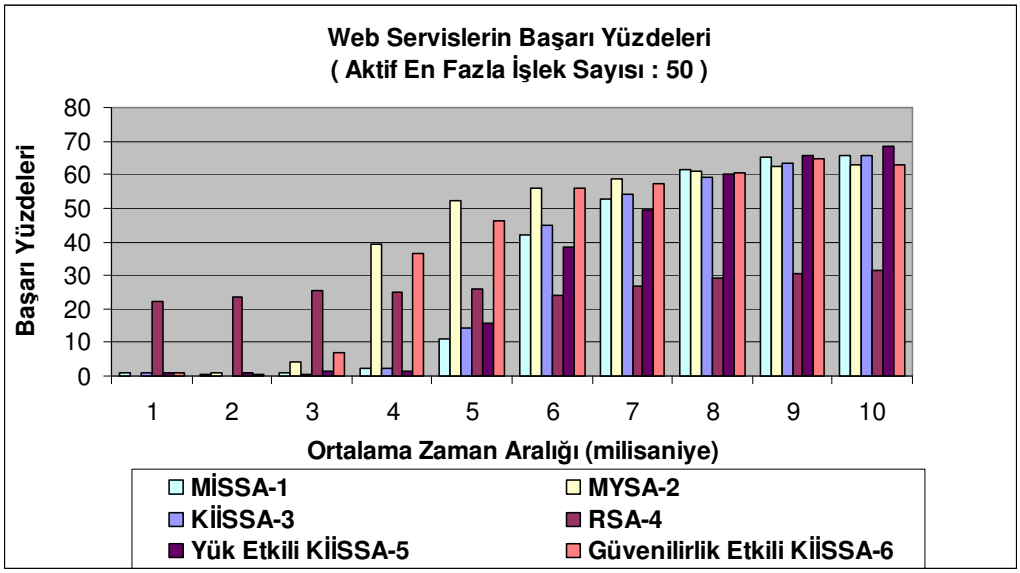
Şekil 5.7. Algoritmaların Servis Başarı Yüzdelerine Göre Karşılaştırılması

Başarı yüzdelerine ilişkin verilere bakıldığında en iyi başarımı gösteren algoritmayı belirlemek ilk bakışta mümkün olmamaktadır. Çünkü zorlama testinin gerçekleştirildiği bu ortamda ortalama zaman aralıklarına göre başarı yüzdeleri oldukça büyük farklılıklar gösterebilmektedir. En kararlı algoritma daha önceki gözlemlerde de olduğu gibi RSA'dır. Hatta bu algoritma ilk üç ortalama zaman aralığında üstünlüğü elinde tutmaktadır ki bunun da en önemli sebebi istek sayılarını servisler arasında hemen hemen eşit olarak dağıtmasıdır. Servislerin yük oranlarının düşük ortalama zaman aralıklarına göre göreceli olarak azaldığı 40ms ve sonraki dilimlerde üstünlük MYSA' dadır ki bunun sebebi daha önce detaylı bir biçimde açıklanmaktadır. KİİSSA ve MİSSA birbirlerine yakın başarım göstermektedirler. Bu sonuçlarda çarpıcı bir nokta ilk üç zaman dilimi için başarı yüzdesinin çok düşük olduğudur.

Başarı yüzdesini arttırmak için son iki test daha yapılmıştır. Yapılan son testlerde iki yeni algoritma kullanılmıştır. Bu algoritmalar tamamen yeni olmamakla birlikte KİİSSA'ya ek olarak yapılmış olan Güvenilirlik Etkili KİİSSA ve Yük Etkili KİİSSA'dır. Karşılaştırmalı sonuçlar Şekil 5.8 ve Şekil 5.9' daki grafiklerden gözlemlenebilir.



Şekil 5.8. İkinci aşama test sonuçları ile Güvenilirlik ve Yük Etkili KİİSSA' larının karşılaştırmaları (Ortalama Cevaplama Süreleri açısından)



Şekil 5.9. İkinci aşama test sonuçları ile Güvenilirlik ve Yük Etkili KİİSSA' larının karşılaştırmaları (Web Servis Başarı Yüzdeleri açısından)

Yük ve Güvenilirlik etkili KİİSSA'lar başarımı öncekiler göre biraz arttırmakla birlikte beklenen yüksek başarımı gerçekleştirememektedirler. Algoritmalara yeni eklentiler yapılması ve en yüksek başarım için yeni algoritmalar tasarlanması gelecek çalışmalara bırakılmıştır. Testin genel sonucu olarak; KİİSSA, MYSA, MİSSA'nın ortalama durumu bildiren RSA' ya göre gerek ortalama cevaplama süresi gerekse web servislerinin başarı yüzdeleri olarak çok daha iyi başarım gösterdikleri sonucuna varılmıştır. Bu durum tasarlanan ve gerçekleştirilen sistemin başarılı olduğunu ortaya koymaktadır.

BÖLÜM 6

6. SONUÇLAR VE GELECEK ÇALIŞMALAR

Web servisler günümüzün hızla gelişen ve kullanılan teknolojilerindendir. Birleşik web servislerle oluşturulan iş akışlarının optimizasyonu, hata toleransının yükseltilmesi ve servis kalitesi özelliklerin destekler hale getirilmesi hem uygulamada hem de akademik dünyada üzerinde yoğun bir şekilde çalışılan alanlardandır. Birleşik web servislerde kullanılan en yaygın endüstri standardı BPEL' dir. BPEL ile web servislerden kurulu bir iş akışı oluşturulabilmekte ve BPEL yürütücüsünde işletilebilmektedir. Standard BPEL ile web servislerin dinamik çalıştırılması gerçekleştirilebilir; ancak otomatik servis seçim desteği sağlanamamaktadır.

Bu tez kapsamında BPEL diline servislerin otomatik seçilmesinde kullanılmak üzere servis kalitesi eklentisi yapılmıştır. Yapılan eklentide kullanılan servis kalitesi parametreleri daha önceki çalışmalardan araştırılmış, yaygın olarak kullanılanları projede de kullanılmış ayrıca bazı yeni parametreler önerilmiştir. Oluşturulan yeni dilin BPEL yürütücüsü ile uyumlu çalışması için açık kaynak kodlu yürütücüde bir takım eklentiler ve değişiklikler yapılmıştır, yürütücüye servis kalitesine göre servis seçme özelliği kazandırılmıştır. Servis kalitesi değerlerinin mevcut web servis altyapısı korunarak gerçek zamanlı olarak elde edilmesi sağlamıştır. Ayrıca yapılan tasarımın başarımını test etmek amacıyla bir test ortamı oluşturulmuş ve zorlama testleri gerçekleştirilmiştir. Sonuç olarak önerilen algoritmalarda gerek ortalama cevap süreleri gerekse servis başarı yüzdeleri açısından olumlu sonuçlar alınmıştır.

Geleceğe yönelik olarak yapılmak istenen yüksek yük altında ve düşük zaman aralıklarında servislerin başarı yüzdelerini arttırmayı hedefleyen yeni algoritmalar oluşturulması ve mevcut algoritmalara eklerin yapılmasıdır. Ayrıca iki servis kalitesi parametresinin bir arada servis seçimlerinde kullanımı örneklerinin çoğaltılması, üç veya daha fazla parametreye göre seçim yapan algoritmaların tasarlanması hedeflenmektedir. Yapılmak istenen bir diğer önemli çalışma ise birden çok tip servis için iş akışı oluşturma algoritmaları kullanılarak başarımlarının ölçülmesidir. Son olarak bütün bu yapılan çalışmaların servis keşfetme özelliği de kazandırılarak mobil ortama taşınması ileriki hedefler arasındadır.

KAYNAKLAR

[1] "SOA and Web Services", erişim adresi: http://www.ibm.com/developerworks/webservices/, erişim tarihi: 20 Şubat 2007.

[2] "Service-Oriented Architecture Technology Center", erişim adresi : http://www.oracle.com/technology/tech/webservices/index.html, erişim tarihi: Mart 2007.

[3] "BEA – Business Software, Business Process Management, Service Bus, Service Oriented Architecture", erişim adresi: http://www.bea.com/, erişim tarihi: Şubat 2007.

[4] "IBM Service Oriented Architecture - SOA", erişim adresi : http://www-306.ibm.com/software/solutions/soa/index.html, erişim tarihi: Mart 2007.

[5] "Service Oriented Architecture", erişim adresi: http://www.oracle.com/technologies/soa/index.html, erişim tarihi: Şubat 2007.

[6] Erl Thomas, "Service-Oriented Architecture: Concepts, Technology, and Design", Prentice Hall, 2005.

[7] Deitel Harvey, "Java Web Services", Deitel Publications, 2002.

[8] "Web Services", erişim adresi: http://www.w3.org/2002/ws/, erişim tarihi: Ocak 2006.

[9] "Java Technology and Web Services", erişim adresi: http://java.sun.com/webservices/, erişim tarihi: Ağustos 2006.

[10] Dustdar, S. and Schreiner, W. "A survey on web services composition", Int. J. Web and Grid Services, Vol.1, No.1, pp.1-30, 2005.

[11] Johann Oberleitner, Florian Rosenberg, Schahram Dustdar, "A Lightweight Model-Driven Orchestration Engine for e-Services", LECTURE NOTES IN COMPUTER SCIENCE, NUMB 3811, pages 48-57, 2006.

[12] M. Tian, A. Gramm, H. Ritter, J. Schiller, "Efficient Selection and Monitoring of QoS-aware Web services with the WS-QoS Framework", IEEE WI, 2004.

[13] "Business Process Execution Language For Web Services 1.1", IBM, Microsoft, Bea, SAP, 2003.

[14] Debmalya Biswas, K. Vidyasankar, "Monitoring for Hierarchical Web Services Compositions", Lecture Notes In Computer Science, numb. 3811, pp. 98-112, 2006.

[15] Zongxia Du, Jinpeng Huai, Yunhao Liu, "Ad-UDDI: An Active and Distributed Service Registry", 31st International Conference on Very Large Data Bases (VLDB 2005) Workshop on Technologies for E-Services, Trondheim, Norway, August, 2005.

[16] M.Tian, T.Voigt, T.Naumowicz, H.Ritter, J.Schiller, "Performance Considerations for Mobile Web Services", Computer Communications, vol.27, pp. 1097 – 1105, 2004.

[17] Xiaohui Gu, Klara Nahrstedt, Wanghong Yuan, Duangdao Wichadakul, "An XML-based Quality of Service Enabling Language for the Web", Journal Of Visual Languages and Computing, vol. 13, part 1, pp. 61-96, 2002.

[18] Anbazhagan Mani, Arun Nagarajan, "Understanding quality of service for Web services", IBM-Developerworks Articles, Jan 2002.

[19] M. Tian, A. Gram, T. Naumowicz, H. Ritter, J. Schiller, "A Concept for QoS Integration in Web Services", Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03), 2004.

[20] Ioannis V. Papaioannou, Dimitrios T. Tsesmetzis, Ioanna G. Roussaki, Miltiades E. Anagnostou, "A QoS Ontology Language for Web-Services", Proc. of the 20th Int. Conf. On Advanced Information Networking and Applications, 2006. AINA 2006, vol.1, pp. 6, 2006.

[21] Ivona Brandic, Siegfried Benkner, Gerhard Engelbrecht, Rainer Schmidt, "QoS Support for Time-Critical Grid Workflow Applications", Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05), 2005.

[22] Dimitris Gouscos, Manolis Kalikakis, Panagiotis Georgiadis, "An Approach toModeling Web Service QoS and Provision Price", Proceedings of the Fourth International Conference on Web Information Systems Engineering Workshops (WISEW'03), 2003.

[23] Michael C. Jaeger and Hendrik Ladner, "Improving the QoS of WS Compositions based on Redundant Services", Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05), 2005.

[24] Conrad Hughes, Jamie Hillman, "QoS Explorer: A Tool for Exploring QoS in Composed Services", IEEE International Conference on Web Services (ICWS'06), pp. 797-806, 2006.

[25] M. Tian, A. Gramm, H. Ritter, J. Schiller, R. Winter , "A Survey of current Approaches towards Specification and Management of Quality of Service for Web Services ", PIK, 2004.

[26] Rami Rifaieh, Uddam Chukmol, Nabila Benharkat, "A Matching Algorithm for Electronic Data Exchange", TES, pp. 34-47, 2005.

[27] Shahram Esmaeilsabzali and Kate Larson, "Service Allocation for Composite Web Services Based on Quality Attributes", Seventh IEEE International Conference on E-Commerce Technology Workshops, pp. 71-82, 2005.

[28] Tao Yu and Kwei-Jay Lin, "Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints", Lecture Notes In Computer Science, numb. 3826, pp. 130-143, 2005.

[29] Tao Yu, Kwei-Jay Lin, "Service Selection Algorithms for Web Services with End-to-End QoS Constraints", Proceedings of the IEEE International Conference on E-Commerce Technology, 2004.

[30] "ActiveBPEL for SOA Orchestration", erişim adresi: http://www.active-endpoints.com/, erişim tarihi: Eylül 2006.

[31] "The Apache Software Foundation", erişim adresi: http://www.apache.org/, erişim tarihi: Ağustos 2006.

[32] "SimJava", erişim adresi: http://www.dcs.ed.ac.uk/home/hase/simjava/simjava-1.2/doc/ref/Package-eduni.simjava.html, erişim tarihi: Mart 2007.

[33] "Setting up world database", erişim adresi: http://dev.mysql.com/doc/world-setup/en/world-setup.html, erişim tarihi: Nisan 2007.

[34] "Normal Distribution", eirişim adresi: http://en.wikipedia.org, erişim tarihi: Mart 2007.

[35] "Simulation Programming Notes", erişim adresi: http://cs.ubishops.ca/ljensen/simulation, erişim tarihi: Nisan 2007.

[36] Erdogan Dogdu and Venkata Mamidenna, "Efficient Scheduling Strategies for Web Services-based E-Business Transactions", Lecture Notes In Computer Science, numb. 3811, pp. 113-126, 2006.

[37] "Eclipse – open development platform", erişim adresi: http://www.eclipse.org, erişim tarihi: Şubat 2007.

EKLER

EK A: Yürütücünün Kaynak Kodunda Servislerin Yüklenmesi Aşamasında Yapılan Değişiklikler ve Eklentiler

ActiveBPEL Yürütücüsünün 2.0 versiyonu incelendiğinde kaynak kodunun projeler halinde düzenlediği gözükmektedir. Projelerin Eclipse geliştirme ortamına eklenmesi ile daha kolay idare edilebilmeleri sağlanabilmektedir. Toplam 13 adet proje mevcuttur. Projelerin bulunduğu klasörler (paket adlandırmaları proje adı olarak belirlenmiştir) şunlardır;

- ddl.org.activeBPEL
- org.activeBPEL.rt
- org.activeBPEL.rt.axis
- org.activeBPEL.rt.axis.BPEL
- org.activeBPEL.rt.axis.BPEL.web
- org.activeBPEL.rt.BPEL
- org.activeBPEL.rt.BPEL.ext.expr
- org.activeBPEL.rt.BPEL.ext.expr.bsf
- org.activeBPEL.rt.BPEL.server
- org.activeBPEL.rt.BPELadmin.help.war
- org.activeBPEL.rt.BPELadmin.war
- org.activeBPEL.rt.tamino
- org.activeBPEL.wsio

Her bir projeye ilişkin .properties uzantılı bir dosya mevcuttur. Bu dosya içersinde proje adı, projenin derlendikten sonra hangi .jar uzantılı dosya olarak saklanacağı, proje sınıflarının hangi yolda bulunacakları gibi bilgiler mevcuttur.

Örnek olarak proje kapsamında sıklıkla başvurulan ve bazı değişiklikler yaptığımız org.activeBPEL.rt.axis.BPEL projesinin .properties dosyasını verebiliriz aşağıdaki tablodaki gibi verebiliriz.

Çizelge EkA.1 org.activeBPEL.rt.axis.BPEL.properties dosyasının içeriği

```
project=org.activeBPEL.rt.axis.BPEL
output.jarfile=ae_rtaxisBPEL.jar
buildnum.dir=org/activeBPEL/rt/axis/BPEL/
buildnum.file=version.properties
perform.javadoc=yes

aux.build=org.activeBPEL.rt.axis.BPEL/support

compile.optimize=no
compile.debug=yes
deprecation=off
fail=true

project.class.path=\
${ae.dist}/ae_wsio.jar;\
${ae.dist}/ae_rt.jar;\
${ae.dist}/ae_rtaxis.jar;\
${ae.dist}/ae_rtBPEL.jar;\
${ae.dist}/ae_rtBPELsvr.jar;\${ae.lib}/commons-logging.jar;\
${ae.lib}/castor-0.9.6-xml.jar;\
```

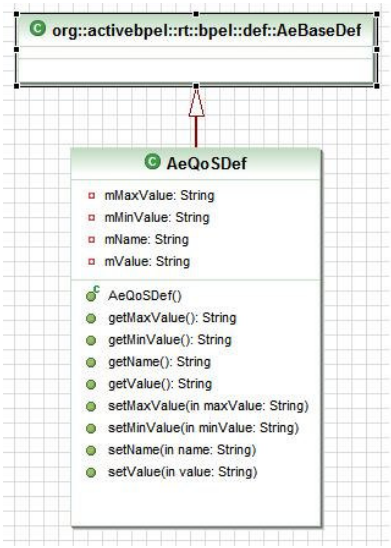
```
${ae.lib}/wsdl4j.jar;\
${ae.lib}/qname.jar;\
${ae.lib}/saaj.jar;\
${ae.lib}/jaxrpc.jar;\
${ae.lib}/axis.jar;\
${ae.lib}/commonj-twm.jar;\
${ae.lib}/jaxen-1.1-beta-8.jar;\
${ae.lib}/commons-httpclient-3.0-rc3.jar

project.javadoc.link1=../../org.activeBPEL.rt/doc/
project.javadoc.link2=../../org.activeBPEL.rt.BPEL/doc/
project.javadoc.link3=../../org.activeBPEL.rt.axis.BPEL/doc/
project.javadoc.link4=../../org.activeBPEL.rt.axis/doc/
project.javadoc.link5=
project.javadoc.link6=
project.javadoc.link7=
project.javadoc.link8=
```

Proje kapsamında yürütücü kaynak kodunda bazı değişiklikler ve eklentiler yapılmıştır. Yapılan değişiklikler maddeler halinde şu şekilde verilebilir.

- org.activeBPEL.rt.BPEL.def.activity.support paketinde
    - AeQoSDef adlı sınıf oluşturuldu
        - Yürütücü ilk başlatıldığında BPEL dosyasında belirtilen servis kalitesi parametrelerinin değerleri bu sınıftaki java

değişkenlerine atanmaktadır. Bu sınıfa ilişkin UML diyagramı aşağıda görülebilir.



Şekil EkA.1 AeQoSDef sınıfının UML diyagramı

- org.activeBPEL.rt.BPEL.def.io.readers.def paketindeki
  - AeReaderVisitor sınıfı içersine
    - public void visit(AeQoSDef aDef) { } metodu eklendi

- org.activeBPEL.rt.BPEL.def.visitors paketi içerisindeki
  - IAeDefVisitor arayüzüne
    - public void visit (AeQoSDef aDef) metod tanımı eklendi

  - AeAbstractDefVisitor abstract sınıfı içersine
    - public void visit(AeQoSDef aDef) { } metodu eklendi

- AeDefPathVisitor sınıfına
  - public void visit(AeQoSDef aDef) { } metodu eklendi

- org.activeBPEL.rt.BPELadmin.war.web.processview paketindeki
  - AeProcessDeftoWebVisitorBase sınıfına
    - public void visit(AeQoSDef aDef) { } metodu eklendi

- org.activeBPEL.rt.BPEL.def paketi içersindeki
  - AeActivityDef sınıfı
    - BPEL dosyasındaki taban aktivite tanımına ilişkin verilerin depolandığı sınıftır. Yeni haliyle AeActivityDef sınıfının UML diyagramı aşağıda gözükmektedir (Yapılan değişiklikler kırmızı ile işaretlenmiştir). Yapılan değişiklikler aşağıdaki gibidir.
      - public void addQoS(AeQoSDef aQoS) { } metodu eklendi.
      - ArrayList tipinden mQoS değişkeni eklendi.
      - Public ArrayList getQoSList() { } metodu eklendi

- org.activeBPEL.rt.BPEL.def.io.writers.def paketindeki
  - AeWriterVisitor sınıfına
    - public void visit(AeQoSDef aDef) { } metodu eklendi.
- org.activeBPEL.rt.BPEL.def.validation paketindeki
  - AeDefValidationVisitor sınıfına
    - public void visit(AeQoSDef aDef) { } metodu eklendi

- org.activeBPEL.rt.BPEL.def.visitors paketindeki
  - AeDefToImplVisitor sınfına
    - public void visit(AeQoSDef aDef) { } metodu eklendi

- AeTraversalVisitor sınıfına
  - public void visit(AeQoSDef aDef) { } metodu eklendi

- IAeDefTraverser arayüzüne
  - public void traverse(AeQoSDef aDef, IAeDefVisitor aVisitor) metod tanımı eklendi
- AeDefTraverser sınıfına
  - public void traverse(AeQoSDef aDef, IAeDefVisitor aVisitor) { } metodu eklendi

- org.activeBPEL.rt.BPEL.def.io paketindeki
  - IAeBPELClassConstants arayüzüne
    - Class tipinden QoS_CLASS değişkeni eklendi
    - QoS_CLASS değişkenine AeQoSDef.class ataması yapıldı

- org.activeBPEL.rt.BPEL.def.io.registry paketindeki
  - AeDefReaderRegistry sınıfının
    - protected void initBPELActivityStandardElements() metoduna ekleme yapıldı.
      - "getStandardReaders().put(makeDefaultQName(*TAG_QoS*), **new** AeDispatchReader(*QoS_CLASS*))" satırı eklendi

- org.activeBPEL.rt.BPEL.def.activity paketindeki
  - AeActivityInvokeDef sınıfı
    - BPEL dili içersindeki invoke aktivitesine ilişkin tanımların yapıldığı java sınıfıdır. Bu sınıfa;
      - public void addQoS(AeQoSDef aQoS) { } metodu eklendi
        - Bu metodun eklenmesi ile BPEL dosyası içersinde, invoke aktivitesi altında, servis

kalitesi parametrelerinin tanımlanması desteklenmiş oldu.

EK B : Yürütücünün Kaynak Kodunda Servislerin Yürütülmesi Aşamasında Yapılan Değişiklikler ve Eklentiler

- org.activeBPEL.rt.axis.BPEL paketindeki
    - AeInvokeHandler sınıfında
        - public IAeWebServiceResponse handleInvoke(IAeInvoke aInvokeQueueObject, String aQueryData) metodunda
            - partner servislerin dinamik seçilmesi ile ilgili kısımlar ilave edilmiştir.

- org.activeBPEL.rt.BPEL.server.engine paketinde
    - ServicesDataStruct sınıfı oluşturulmuştur
    - ServiceData sınıfı oluşturulmuştur
    - AeEngineFactory sınıfında
        - Bu sınıf yürütücünün singleton örüntüsü ile tasarlanmış bir örneğidir.
            - ServicesDataStruct sınıfı türünden sDataStruct değişkeni eklenmiştir.
            - public static ServicesDataStruct getSDataStruct () { } metodu eklenmiştir
            - public static void preInit (IAeEngineConfiguration aConfig) { } metodunda değişiklik yapılmıştır.

- org.activeBPEL.rt.BPEL.impl.activity paketinde
    - AeActivityInvokeImpl sınıfında
            - public void execute() { } metodunda
                - partnerlinklerin alınması, servis kalite parametrelerine ilişkin tanımların nesne üzerinden çağırılması ve istemcinin isteğinin kuyruğa atılması adımlarında değişiklikler yapılmıştır.

EK C : Terim Sözlüğü

Türkçe Terim	İngilizce Terim
ayarlamalar	configurations
birlikte işlerlik	interoperability
insan tarafından okunabilir	human readable
işlem-bilgi	transaction
genel	global
hata ayıklama	debug
melez	hybrid
seğirme	jitter
servis kalitesi kısıtları	QoS Constraints
süreç	process
tabandan tepeye	bottom up
uzaktan hata ayıklama	remote debugging
web servisleri	web services
yerel	local
yukardan aşağı	top down
yürütücü	engine
zaman damgası	time stamp

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: MESCİGİL, Ömer
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 14.08.1983 İstanbul
Medeni hali	: Bekâr
Telefon	: 0 (312) 292 40 76
Faks	: 0 (312) 292 40 76
e-mail	: omescigil@etu.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	İstanbul Teknik Üniversitesi/Bilgisayar	2005

İş Deneyimi

Yıl	Yer	Görev
2005-2007	TOBB Ekonomi ve Teknoloji Üniversitesi	Araştırma Görevlisi

Yabancı Dil

İngilizce