Tasarım Desenleri ve Java Web Servisleri ile Çok Katmanlı Bir Sistem Tasarımı

Arda Göknil[1], Tayfun Elmas[2], N.Yasemin Topaloğlu[3]

Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, 35100, Bornova, İzmir
[1] goknil@bilmuh.ege.edu.tr, [2] elmas@bilmuh.ege.edu.tr, [3] yasemin@bornova.ege.edu.tr

Özet. Bu çalışmada, Rational Unified süreci ile geliştirilen, Java Web servisleri tabanlı çok katmanlı bir sistemin mimarisi ve sistemin esnekliğini sağlamak amacıyla kullanılan tasarım desenleri tanıtılmıştır. Sistemde tasarım desenleri ile kurulan esnek yapı, mimarinin tüm katmanlarındaki bileşenlerin farklı protokol ya da arayüzlerle çalışan eşdeğerleriyle değiştirilebilmelerine ya da nesneye dayalı olarak genişletilebilmelerine olanak sağlamaktadır. Desenlerin sağladığı kolay genişletilebilirlik, Rational Unified süreç kapsamında yinelemeli olarak yeni işlevlerin eklendiği fazlarda, yeni bileşenlerin ana mimariye uyarlanmasını kolaylaştırmıştır.

1 Giriş

Çok katmanlı uygulamaların ağırlık kazanmasıyla sunum mantığı, bilginin asıl işlendiği yer olan iş mantığından ayrılmış ve sistem tasarımcıları bilgi servislerini tüm platformlarda daha etkin sunmak için sunum standartları geliştirmeye başlamışlardır. Bu standartlaşmanın Internet ortamına yansımasının bir sonucu olan Web servisleri, çalıştıkları platformlardan bağımsız haberleşmeleri için *XML* tabanlı protokollerle iletişim kuran bir altyapı üzerinde çalışırlar ve dış dünyaya sadece sundukları bilgi servisinin ara yüzünü bildirirler [1].

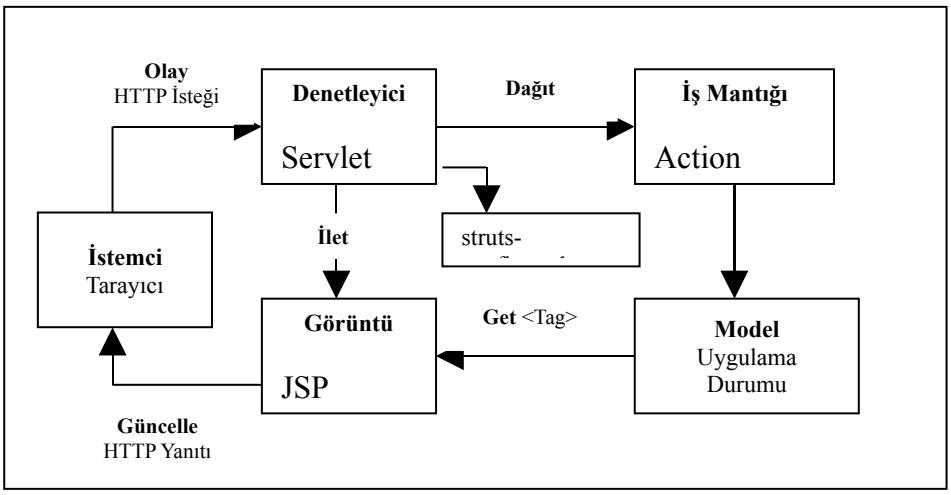
Bu çalışma kapsamında İnternet üzerinde farklı özelliklerdeki istemci uygulamalar tarafından standart protokoller aracılığıyla yüksek performansta hizmet verebilecek çok katmanlı bir sistemin geliştirilmesi amaçlanmıştır. İstemciler farklı sitelerden ve platformlardan bağlantı kuracakları için ara kontrol katmanının dağıtık bir yapıda olması gerekmektedir. Bu durum, ara katmanın uzak sistemi istemciden soyutlayarak ve farklı iletişim protokolleri kullanarak çalışmasını gerektirmektedir. Ayrıca sistemin sunduğu servislere eklemeler yapılabilecek olması da yazılımın her katmanda değişime karşı esnek olmasını gerektirmektedir. Bu gereksinimleri göz önüne alarak birçok bilimsel disipline hizmet sunması beklenen bir İstatistik Analiz Sistemi, Rational Unified Süreci [2] izlenerek, tasarım desenlerinden yararlanılarak tasarlanmış ve Java Web servisleri ile *servlet* teknolojisi kullanılarak gerçekleştirilmiştir.

Bu bildiride, geliştirilen sistemin mimarisi ve kullanılan tasarım desenleri açıklanmıştır. Bildirinin ikinci bölümünde, web servisleri kullanımı ve sistem katmanlarının yapısı tanıtılmış, üçüncü bölümde ise sistemin esnekliğini sağlamak amacıyla kullanılan tasarım desenleri incelenmiştir. Dördüncü bölüm, sonuç bölümünü içermektedir.

2 Sistemin Mimarisi

2.1 Sistemde Yer Alan Yazılım Katmanları

Genişletilebilir yazılım sistemlerinin oluşturulmasında çözümler getiren *Model-View-Controller* (MVC) mimari deseni, tasarım desenlerini temel alan ve tasarım desenlerinin üzerinde uygulama mimarisi kapsamında yer alan bir desendir [4]. MVC mimari deseni, etkileşimli uygulamaları üç parçaya böler. Model katmanı uygulamanın çekirdek fonksiyonlarını ve verisini içerir. Sunum katmanı, kullanıcıya bilgileri sunarken, kontrol katmanı kullanıcı girdilerini yakalama görevini yerine getirir. Sunum ve kontrol katmanları ikisi birlikte kullanıcı arayüzünü oluşturur. Şekil-1'de, MVC'nin sistemimizdeki uygulaması görülmektedir.



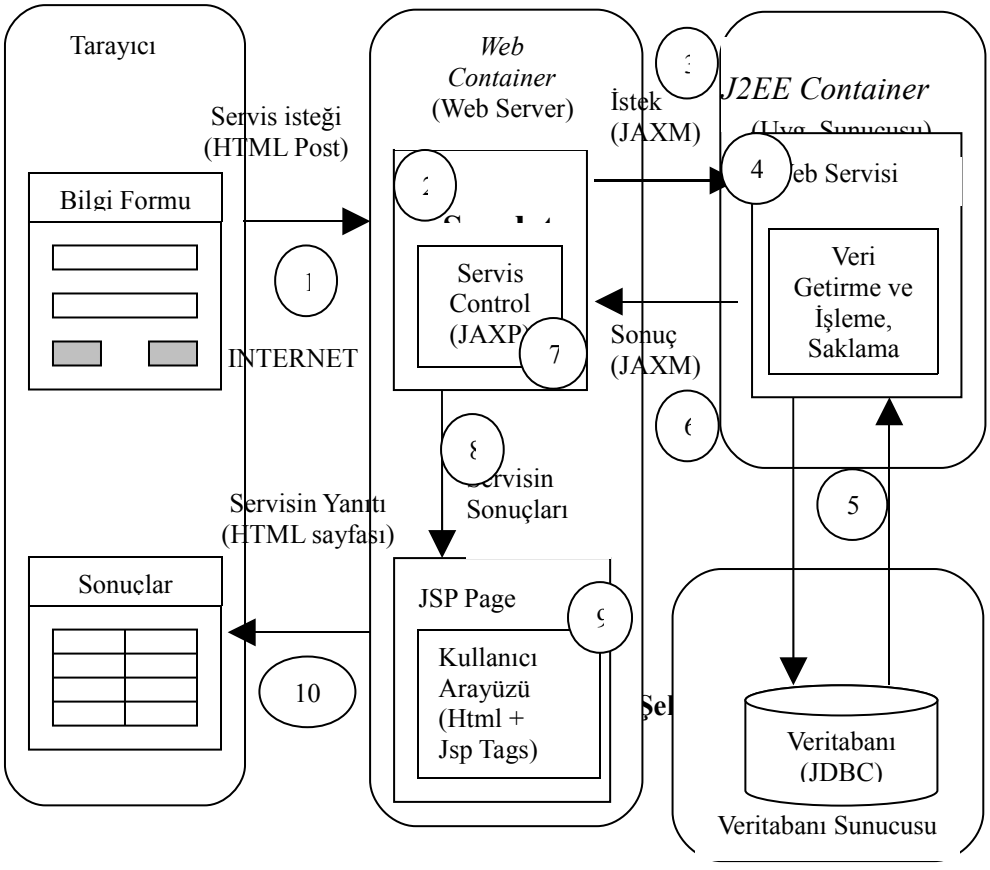
Şekil 1. Sistemdeki MVC Uygulaması

Geliştirilen sistem üç katmanlı bir yapı üzerine kurulmuştur. Sistemde sunulan analizlerin işletilmesini sağlayan servisler, Web servisi olarak gerçekleştirilmiş olup mimarinin en son katmanını yani üçüncü katmanı oluşturmaktadır. Bu katmanın önünde web servislerinden hizmet isteyen, ilgili çağırımları yapan *Java Servlet* ve Java Server Pages (*JSP)* [3] teknolojileriyle gerçekleştirilmiş ikinci katman bulunmaktadır. İkinci ve üçüncü katmanlar aynı makinede çalışabileceği gibi ayrı makinelerde de hizmet verebilirler. Farklı kuruluşların sistemden yararlanmak isteyebilmesi nedeniyle birbirinden farklı ikinci katmanlar aynı üçüncü katmandan hizmet alabilir. İlk katmanda kullanıcıdan veriyi alma ve sonucu görüntüleme işlemleri istemci sistem üzerinde gerçekleştirilmekte, böylece kullanıcı ile etkileşim farklı yöntemlerle (*HTML* formları, grafik tabanlı ara yüzler gibi) sağlanabilmektedir. Bu şekilde temel iş mantığı merkezi bir şekilde hizmet verirken, istemci ve istemcilere arayüz hizmeti veren ara katman dağıtık bir duruma gelmiştir. Kullanıcı ile etkileşim ise *HTML* sayfaları ile sağlanmıştır. Bu durum analizlerin işletildiği Web servisleri ile istemcinin arasına bir de Web sunucu arabiriminin girmesini gerektirmiştir. Kullanıcı etkileşiminin *Java-Swing* kullanan tek başına (*standalone*) bir uygulama ile sağlanması durumunda yazılım doğrudan Web servisleri ile bağlantıya geçebilecektir.

Struts çerçevesi MVC deseninin Web uygulamaları üzerinde bir gerçekleştirimi olup; uygulamanın denetleme ve iş süreçlerini mümkün olduğunca İnternet uygulaması kalıplarından kurtararak *HTTP* ve diğer alt düzey protokollerden soyutlar.

2.2 Katmanlar Arası İletişim

Bölüm 2.1'de tanıtılan katmanlar arasındaki iletişim Şekil-2'de görülmektedir.



Şekil 2. İstatistik Analiz Sistemindeki Katmanlar Arası İletişim

Sistemdeki katmanlar arası iletişim 10 adımdan oluşan bir senaryo içinde incelenebilir:

1. Kullanıcı, Web tarayıcısı üzerinde kendisine sunulan forma işlenecek verilerini girer ve bu formu İnternet üzerinden *HTTP-POST* protokol mesajıyla Web sunucusuna gönderir.

2. Web sunucusu, isteğin yönlendirildiği adresi değerlendirerek bu isteği uygun *servlet* nesnesine yönlendirir. *Servlet,* genellikle yerine getireceği sürece göre özelleşmiş olduğu için bu isteği o sürece ve servisi yerine getirecek Web servisinin eklentisine uygun olarak *SOAP* protokolü çerçevesinde düzenler.

3. *Servlet,* servis isteğini bir *SOAP* mesajı halinde *JAXM API* yardımıyla istemciye saydam bir şekilde Web servisine iletir.

4. Web servisi, isteği *SOAP* mesajından ayırmak için *JAXM API* kullanır. Bu aşamada iş mantığı çalışmaya başlar ve veri işleme sürecine girilir.

5. Verinin işlenmesi sırasında bazı ek verilere gereksinim duyulabilir. Bu ek veri, Web servisinden bağımsız olan bir veritabanı sunucusundan *JDBC* (*Java DataBase Connectivity*) *API* yardımıyla ve *SQL* sorgu dili ile çekilir.

6. İşlem sonunda elde edilen sonucun istemciye geri bildirilmesi için Web servisi yine veri alımında kullandığı *SOAP* protokolünü kullanır. Sonuç gönderme işlemiyle birlikte Web servisinin bir hizmet sunum döngüsü daha sona ermiş olur.

7. *Servletin* sonucu almasıyla sonucun kullanıcıya uygun biçimde iletilmesi süreci başlar. *Servlet,* sonucu *SOAP* mesajından ayrıştırarak ham veriyi elde eder. Bu veriyi kullanıcıya iletme görevi kendisinde değil, *JSP* sayfalarında olduğu için *servlet* sadece veriyi bazı kriterlere göre değerlendirme ve belirli yapıda *JSP* sayfasına göndermekle görevlidir.

8. Sonuçlar standart bir yapıya çevrilerek *JSP* sayfasına geçiş yapılır. Geçiş ve veri aktarımı, *Servlet-JSP API*'de yer alan metot çağrımları ile yerine getirilir.

9. Veri, *HTML* dili ve *JSP*'ye özel *tag* kütüphaneleri yardımıyla kullanıcıya aktarılmak istenen sunum kalıbına çevrilir. Sonuç, istemci tarayıcısına (*browser*) gönderilecek bir *HTML* sayfasıdır.

10. *HTML* sayfası istemcinin bilgisayarına Internet üzerinden *HTTP* protokolü ile gönderilir. Kullanıcı, isteğin yerine getirilmesi için verilen tüm bu hizmetlerin ne olduğundan ve diğer tüm özelliklerinden soyutlanmıştır.

Sistemin Java gibi ortam bağımsız bir dille geliştirilmesi sunucu durumundaki Web servislerinin de tüm ortamlarda çalıştırılmasını sağlamaktadır. Ortam farklılığı servislere bağlanmak için kullanılan protokolleri etkilememektedir.

3 Sistemde Kullanılan Tasarım Desenleri

Tasarım aşamasında performans ve etkinlik kadar yazılımın değişime karşı esnekliği de dikkate alınmıştır. Bu amaçla yazılımın farklı katmanlarında farklı amaçlara yönelik tasarım desenleri kullanılmıştır. Tasarım desenleri, özellikle sistemin Web servislerinin çağrımı, Web servisleriyle İnternet üzerinden iletişimi, analizlerin uygulanması gibi amaçlarla farklı bileşenlerin sisteme dinamik olarak eklenmesi konusunda önemli yararlar sağlamıştır. İstatistik Analiz Sisteminde kullanılan tasarım desenleri [5] aşağıda açıklanmıştır.

Strategy.

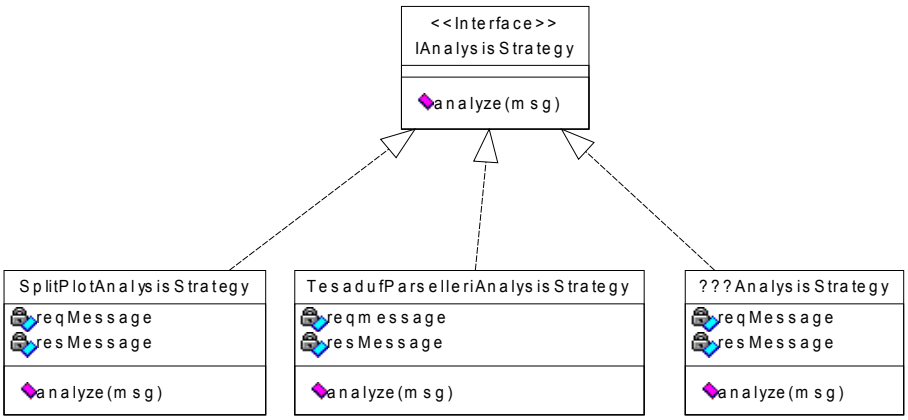
Kullanım Amacı : *Strategy* tasarım deseni, içerik adı verilen bir sınıfta birbiriyle ilintili birden çok algoritmayı tutan ve kontrol eden bir desendir.

Sistemde Uygulanışı : Sistemde birden çok analiz olacağı için değişik tiplerdeki analizleri gerçekleştirmek amacıyla sistemde birden çok analiz algoritması bulunacaktır. Bu nedenle analiz algoritmalarının bir arada tutulması ve uygun olanlarının seçilerek işletilmesi Şekil-3'te görülen *Strategy* tasarım deseni kullanılarak gerçekleştirilmektedir. Ayrıca iletişim kanalından gelen verilerin Web servisinde kullanılan ortak bir veri yapısında tutulması gerekmektedir. Bunun için de veri dönüşümüne gereksinim vardır. Ancak veriler farklı biçimlerde gelebilir. Bu nedenle değişik veri biçimlerinin ortak bir veri yapısında tutulmasını sağlayan değişik algoritmaların bulunması ve bu algoritmalardan uygun olanlarının işletilmesi gerekmektedir. Sistemde bu işlemler için de *Strategy* tasarım deseni kullanılmaktadır.

Builder.

Kullanım Amacı : *Builder* tasarım deseni, çok sayıda bileşen içeren karmaşık nesnelerin oluşturulma işlemi için kullanılan ara yüzü standartlaştırarak nesnelerin iç yapılarının ve bu yapıların oluşturulma biçimlerinin soyutlaştırılmasını sağlar.

Sistemde Uygulanışı : Şekil-2'deki ikinci adımda analizlere ait bilgileri, parametre tanımlarını, veri giriş formlarını ve veri taşıyıcı sınıfları tanımlamak amacıyla *AnalizSpecification* adlı sınıf tanımlanmıştır. Bu sınıftan türetilen nesne, analize ait birçok bilgi ve farklı tiplerde nesneler (*DataSpecification*, sınıf tanımlamaları) içermektedir. Tüm bu bilgilerin ikinci adımda çalışan Web uygulamasının ilk açılışında bir kaynaktan alınması ve *AnalizSpecification* nesnesine eklenmesi gerekir. Bu işlem çok farklı şekillerde yapılabilir. *AnalizSpecification* nesnesine farklı şekillerde ilgili olduğu bilgileri eklemek ve nesneyi son haline getirmek için *Builder* tasarım deseni kullanılmıştır. Böylece farklı *Builder* sınıfları ortak bir arayüzle *AnalizSpecification* nesnelerinin gereksinimi olan bilgileri farklı şekillerde toplayıp ekleyebilirler.



Şekil 3. Analiz Sınıflarına İlişkin *Strategy* Tasarım Deseni

Decorator.

Kullanım Amacı : *Decorator* tasarım deseni, bir nesneye ait bir fonksiyonun gerçekleştirimine dinamik olarak ek sorumluluklar ve işlemler eklemek için kullanılır.

Sistemde Uygulanışı : Şekil-2'deki ikinci adımda *DataSpecification* nesneleri, servlet kodu kapsamında analiz verisi ile ilgili işlemleri gerçekleştirmekten sorumludurlar. Analiz verisi tek bir değişkenden oluşabileceği gibi bir değişken dizisi ya da matrisi olabilir. Dizi ya da matris durumundaki değişkenlerle yapılan veri aktarma (*set* ve *get* işlemleri), biçim kontrolü gibi işlemler *DataSpecification* sınıfının bu işlemleri yerine getirecek metotlarına ek görevler yüklemektedir. Bu eklentileri, temel sınıfı değiştirmeden gerçekleştirmek *Decorator* deseni ile sağlanmaktadır. Bu durumda öncelikle tek bir elemanı temsil eden *DataSpecification* nesnesi oluşturulur. Oluşan bu nesne, dizi ya da matrise özel sorumluluklar eklenmek üzere *ArrayDataSpecification* ya da *MatrixDataSpecification* nesneleri tarafından korunur.

Visitor.

Kullanım Amacı : *Visitor* tasarım deseni, bir nesne grubundaki bileşen nesnelerin her birine farklı ve birbiriyle ilişkisi olmayan işlemler uygulanmak istenildiğinde kullanıcı katmanın iletişim halinde olduğu ana nesnenin ara yüzünü değiştirmeden operasyonların bileşenlere atanarak farklı operasyonların kontrol edilmesi amacıyla kullanılır.

Sistemde Uygulanışı : Şekil-2'deki ikinci adımda Analiz verisini taşıyan taşıyıcı (*Container*) nesnesine, arayüzü sabit kalmak şartıyla analiz tanımında (*AnalysisSpecification*) yer alan farklı isim ve tipteki veri tanım nesnelerinin veri

aktarması işleminde *Visitor* deseni kullanılmaktadır. Böylece taşıyıcı nesnenin arayüzü değiştirilmeden –veri isim ve tipinin farklı olması nedeniyle- farklı işleyişlere sahip operasyonların gerçekleştirimi veri tanım nesnelerine yüklenir.

Proxy.

Kullanım Amacı : *Proxy* tasarım deseni; bir nesnenin, kendisinden farklı bir nesne gibi davranarak bu nesneye erişimi kontrol altına alması amacıyla kullanılır.

Sistemde Uygulanışı : Analiz için gereken verinin alınması ve analiz işletildikten sonra sonucun kullanıcıya görüntülenmesi işlemi *Servlet API* ve *JSP* sayfaları ile yapılmaktadır. Şekil-2'de üçüncü ve altıncı adımlardaki analiz verisinin gerçekleştirime aktarılması ve sonucun alınması *AnalysisCall* arayüzü ile gerçekleşmektedir. *Servlet*, analizin doğrudan analize ait özel sınıfın çağırılarak mı yoksa ağ üzerinden bu özel sınıfa bir iletişim protokolü üzerinden mesaj gönderilerek mi işletileceğinden soyutlanmıştır. Ancak yapılması gereken, sonuçta yerel olarak da olsa analize ait özel sınıfın çağrılmasıdır. Burada *Proxy* deseni, uzak bilgisayar üzerindeki sınıfa ulaşmak için bir ara katman tanımlamaktadır.

Adapter.

Kullanım Amacı : *Adapter* tasarım deseni; sistemde ortak olarak kullanılan ya da değişime uğramaması gereken bir arayüzün, farklı bileşenlere ait ara yüzlere uyarlanması amacıyla kullanılır. Böylece sistemin kullandığı arayüz değişmediği için çok farklı gerçekleştirim metotları sistemle bütünleştirilebilir.

Sistemde Uygulanışı : Analize ait parametrelerin kullanıcıdan alındıktan sonra analiz gerçekleştirimine gönderilmesi gerekmektedir. Analiz gerçekleştirimi; istemcileri ile Web servisleri, CGI, Servlet, RMI gibi farklı protokoller üzerinden iletişim kurmak üzere hazırlanmış olabilir. Bu nedenle istemcinin kodunda değişiklik olmaması için istemcinin kullanacağı arayüzü standartlaştırıp farklı protokoller için kullanılan arayüzler arasında çevrimi gerçekleştirecek protokollere özgü *Adapter* sınıfları tanımlanması sağlanmıştır.

Facade.

Kullanım Amacı : *Facade* tasarım deseni, sistemde birbiriyle bağlantılı birden çok işin tek bir sınıf tarafından kontrol edilmesi amacıyla kullanılır.

Sistemde Uygulanışı : Şekil-2'deki üçüncü adımda servis isteğinin *SOAP* mesajı halinde *JAXM API* yardımıyla web servisine iletilmesi sırasında web servisi tarafından gelen isteği *HandlerServlet* adlı *servlet* nesnesi karşılar ve analizlerin başlatılabilmesi için gerekli çağrımları gerçekleştirir. *Facade* tasarım deseni kapsamında oluşturulan *Controller* sınıfının *starts* metoduna analiz oluşturma ve işletme işleminin mantığını bulunduran çağrımlar yazılarak bu çağrımların doğrudan *HandlerServlet* içerisine yazılmasının önüne geçilmiştir. Böylece iletişimde bulunan sınıf, iş mantığından ayrılmıştır ve iletişimde bulunan sınıfın değişmesi durumunda yeni iletişim sınıfı sadece iş mantığını içinde bulunduran *Controller* sınıfına çağrım yaparak bütün iş mantığıyla ilgili verileri geçirebilecektir.

Factory.

Kullanım Amacı : *Factory* tasarım deseni, koşullara bağlı olarak mümkün olan birden çok sınıftan birinden ilgili nesneyi yaratmayı sağlayan tasarım desenidir.

Sistemde Uygulanışı : Sistemde, F*actory* desenini temel alan *ServicesFactory* sınıfı özellikle *Strategy* tasarım deseni kullanılarak oluşturulan analiz nesnelerinin yaratılmasından sorumludur.

Singleton.

Kullanım Amacı : *Singleton* tasarım deseni, programlama sırasında bir sınıftan sadece bir tane nesnenin türetilmesi gerektiği durumda kullanılır.

Sistemde Uygulanışı : *Singleton* tasarım deseninin *factory* sınıfları üzerinde uygulanmasıyla F*actory* nesnelerinin Web servisine gelen her istek karşısında yeniden yaratılması önlenmiş ve tek bir nesnenin referanslarından işlemlerin kontrol edilmesi sağlanmıştır.

4 Sonuç

Bu çalışmada, Web Servisleri ile geliştirilen çok katmanlı bir servis sisteminin mimarisi tanıtılmıştır. Sistemin geliştirilmesinde *Rational Unified Process* kullanılmıştır. Unified sürecin ilk fazlarında risk taşıyan kısımlar ele alınmış,

diğer aşamalarda ise mimari açıdan daha az risk taşıyan ancak sistem işleyişindeki ayrıntıları ele alan parçaların işlenmesi planlanmıştır.

Web servislerinin *servlet* teknolojisiyle kullanılması sisteme üç katmanlı bir yapı kazandırmıştır. Bu üç katmanlı yapıda üçüncü katmanı oluşturan Web servisleri birden fazla ikinci katmana (*servlet*) hizmet verebilmektedir. Tasarım desenleri ve MVC mimari deseni kullanılarak oluşturulan esnek yapı, bundan sonraki çalışmalarda sisteme yeni servislerin eklenmesine olanak sağlayacağı gibi ileride gösterim ve sunum katmanında gerçekleştirilecek değişikliklerin birbirini etkilememesini sağlayacaktır. Bu kapsamda, ileride sisteme mobil cihazlardan erişimin gerçekleştirilmesi planlanmaktadır.

Kaynakça

1. E. Armstrong, S. Bodoff, D. Carson, M. Fisher, D. Green, K. Haase, The Java Web Service Tutorial, 2002
2. C. Larman, Applying UML and Patterns, Prentice-Hall,2002
3. Marty Hall, Servlets and Java Server Pages, Sun Microsystems Press, 2000
4. F. Buschman, R. Maunier, H. Rohnert, P. Sommerlad, M. Stal, A System Of Patterns, John Wiley & Sons
5. E. Gamma, Design Patterns, Addison-Wesley, Reading, Mass., 1994