Kullanıcı Arayüzü Tasarımlarının Üst Düzey Programlama Dillerine Dönüştürülmesine Bir Yaklaşım

Şefik Temel[1], Umut O. Turgut[2], Volkan Bağcı[3], Mehmet S. Aktaş[4]

[1,4] Bilgisayar Mühendisliği Bölümü, Yıldız Teknik Üniversitesi, İstanbul
[2] Bilgisayar Mühendisliği Bölümü, Orta Doğu Teknik Üniversitesi, Ankara
[3] Master of Science, Computer Science, University of Arkansas, Little Rock
[2,3] Ar-Ge Merkezi, Cybersoft, İstanbul

[1] sefiktemel@gmail.com
[2] umut.turgut@cybersoft.com.tr
[3] volkan.bagci@cs.com.tr
[4] aktas@yildiz.edu.tr

Özet. Yazılımın kullanıcı ile olan iletişimini sağlayan kullanıcı arayüzlerini tasarlamak ve programlamak, gereksinim mühendisinin ve programcının ürün üretme sürecinde, uğraş harcadıkları önemli adımlardır. Gereksinim mühendisi, bu adımları kendi adına kolaylaştırmak için çeşitli tasarım araçları kullanmayı tercih eder. Bu tasarım araçlarını kullanarak, ekranları müşteri ile görüşmeler yaparken kolaylıkla ve kısa sürede tasarlayabilmekte ve müşteride görsel bir etki bırakabilmektedir. Ancak, programcı, tasarlanan bu ekranları gerçekleme sürecinde, tasarım araçlarının çıktılarını el yordamlı yöntemlerle programlamak zorunda kalmaktadır. Bu işlem de yazılım geliştirme sürecinde önemli bir zaman kaybına sebep olmaktadır. 'Rapid Application Development' yazılım geliştirme sürecinde doğru ürün elde edilene kadar sürekli yeni bir ilk örnek hazırlanır, eksikleri gözden geçirilir, eksikleri giderilmiş yeni ilk örnek hazırlanır. İlk örnek hazırlama sürecinde tasarlanan ekranların çevrim işlemi, yazılım ekibine fazladan uğraş getirir. Bu bildiri, bahsedilen problemi giderebilmek için, kullanıcı arayüzü tasarımı ve programlaması adımlarında yapılabilecek optimizasyonların araştırılması üzerinde odaklanmıştır. Bu konuda yaptığımız çalışma ile tasarım araçlarının çıktılarının kaydedildiği dosyaları girdi olarak kullanan, tüm tasarımları yazılımcının kullanacağı programlama diline ya da kullanıcı arayüzü tanımlama diline çevirebilen genel bir çözüm üretilmiştir. Çalışmamızda, tasarım aracı olarak Balsamiq Mockups (yaygın kullanımı ve tasarlanan ekranları dosyalara kaydedebilme özelliğinin olması nedenleriyle) kullanılmıştır. Geliştirdiğimiz dönüştürücü yaklaşımı, Balsamiq Mockups tasarım aracından üretilen kulllanıcı arayüzülerinin herhangi bir programlama diline ya da kullanıcı arayüzü tanımlama diline dönüşümünü başarı ile sağlayabilmektedir. Çalışmamızda elde edilen sonuçları bu bildiride paylaşıyoruz.

Anahtar Kelimeler: Kullanıcı arayüzü tasarımı, DOM tabanlı kod çevirme, EBML, HTML5, XSLT, Balsamiq Mockups, Genel kod oluşturma, Facade tasarım kalıbı

1. Giriş

Anlaşılması ve kullanımı kolay bir yazılımı üretmek için dikkat edilmesi gereken nokta kullanıcı arayüzleridir. Kullanıcı arayüzü bir yazılımın kullanıcı ile iletişim kurabileceği tek yol olduğu için sistemi anlamlı kılacak en önemli bileşenlerden birisidir. Bir kullanıcı arayüzü hedeflenen kullanıcıya hitap edebilecek kadar basit aynı zamanda sahip olduğu tüm işlevlerini kullanıcıya sunabilecek kadar bütünleşik olmalıdır[1]. Yazılım mühendisliğinde bu görev gereksinim mühendisinin elindedir ve gereksinim mühendisi müşteriden aldığı talep doğrultusunda uygulanabilirliği makul ve en basit sistemi tasarlamaya çalışır. Bu işi daha hızlı yapabilmek için gereksinim mühendisleri, kolayca kullanabilecekleri belirli tasarım araçlarını kullanmayı tercih ederler. Kullanılabilir tasarım araçlarına Pencil Project[2], Flair Builder[3], Balsamiq Mockups[4] örnek olarak verilebilir.

Bu araçlardan birisi olan Balsamiq Mockups hem web uygulaması sağlaması nedeniyle hem de basitliği ile oldukça yaygın kullanılan bir kullanıcı arayüzü tasarım aracıdır. Balsamiq Web uygulaması ile kullanıcı arayüzleri çevrimiçi ortamda tasarlanabilmekte ve tasarlanan ekranlar ".bmml" uzantılı Balsamiq Mockups dosyaları halinde kaydedilebilmektedir. Ayrıca Balsamiq Mockups ile tasarlanan ekranların XML kodu dışarı aktarılabilmektedir. Elde edilen saf XML kodunu geliştiriciler kendi aralarında paylaşabilmektedir[5].

Araştırma problemimizi tetikleyen gerçek hayat senaryosu Cybersoft firmasının Bankacılık Sektörü'ne yönelik ürünler geliştiren yazılım ekibinden gelmektedir: Gereksinim mühendisleri temel bankacılık uygulamalarını geliştirme sürecinde kullanıcı arayüzlerini tasarlamak için Balsamiq Mockups tasarım aracını kullanmaktadır. Bu sayede, bankacı uzmanlarla yapılan görüşmelerde çizim sürecini çok daha hızlı gerçekleştirmektedirler. Hazırlanan ekranlar bankacılık sektöründe kullanılacak bileşenleri içeren çizimlerdir ve sayıları, güncellenme sıklıkları oldukça yüksektir (örnek: operasyonel bankacılık işlemleri için 1000'ler mertebesinde ve sürekli güncelleme istekleri alan ekrarlar mevcuttur.) Burada karşılaşılan en büyük problem, tasarım aracı ile çizilen tasarımların, kolaylıkla programcı tarafından kodlanabilir olmamasıdır. Cybersoft firması, kendilerinin geliştirdiği "Aurora Framework" [6,7] adlı yazılım katmanında kullanıcı arayüzlerini geliştirmektedir. Gereksinim mühendisinin kendilerine ilettiği kullanıcı arayüzü ekranlarını, Aurora Framework sistemi tarafından kullanılan kullanıcı arayüzü tanımlama dilinde tekrardan tasarlamak zorunda kalmaktadırlar. Bu denli fazla kullanıcı arayüzünün gerçeklendiği bir yazılım evinde, böyle bir durum önemli bir iş gücü ve zaman kaybına, dolayısıyla ekonomik kayba da sebep olamaktadır.

Klasik yazılım geliştirme sürecine kolaylık sağlayacak bu yöntem aynı zamanda James Martin'in önerdiği 'Rapid Application Development' (RAD) yazılım geliştirme modeli [8] için de büyük kolaylık sağlayacaktır. Sürekli yeni tasarımların hazırlanmasını ve kullanıcı onayına sunulabilecek bir ön ürünün oluşturulmasını hedefleyen bu modelde de bizim önerdiğimiz yöntem kullanılarak yazılım ekibinin çevrim ile vakit kaybı yaşamaması sağlanacaktır.

Bu bildiriyle, problemin çözümüne yönelik olarak, tasarımcıların ve programcıların kullanabilecekleri ortak bir KATD (Kullanıcı Arayüzü Tanımlama Dili) kullanıl-

masını öneriyoruz. Bu araştırmanın çıktısı da, tasarım araçlarının çıktılarını, kullanıcı arayüzleri tanımlama dillerine dönüştürebilen, tasarım aracı ya da programlama dili bağımlılığı olmayan bir dönüştürücü yazılım sistemidir.

Geliştirilen yazılımın ilk örneğinin testi için, tanımlama dili olarak 'Enhanced Bean Markup Language' (EBML) [6,7] tanımlama dilinin kullanımını seçiyoruz. EBML tanımlama dili XML tabanlı bir KATD olup, özellikle bankacılık uygulamalarının kullanıcı arayüz tasarımlarının tanımlanmasında oldukça başarılı sonuçlar vermiştir. Bu dili seçmenizin diğer bir nedeni de, gerçek hayat senaryolarını aldığımız Cybersoft firması tarafından kullanılan kullanıcı arayüz tanımlama dili olmasıdır.

Örnek tasarım aracı olarak kullanmakta olduğumuz, Balsamiq Mockups, çizim ekranlarını dosyaları ".bmml" uzantılı dosyalara kaydetmektedir. Bu dosyalar XML tabanlı olup ekrandaki her bir kontrolü ve kontrole ait özellikleri tüm detayları ile içermektedir. Bu araştırmayla; tasarım çıktısı olan XML tabanlı dosyaları (örnek: .bmml dosyaları), bir veya birden fazla kullanıcı arayüzü tanımlama dillerine (örnek: EBML ya da HTML5) dönüştürebilen, genişleyebilir (birden fazla tanımlama dilini yeni kodlar yazmadan destekleyebilir) bir mimari yapıya sahip dönüştürücü bir yazılım sistemi geliştirmeyi hedefledik. Bu sayede, tasarımcı ile program yazan ekip arasında orta bir katman olacak ve kod yazan ekibin tasarım ile tekrar ilgilenmesine gerek kalmayacaktır. Fazladan iş yükü, zaman harcanmasının maliyet üzerine ekleyeceği yük ortadan kalkacaktır. Aynı zamanda proje geliştiren ekip kullanıcı arayüzünü tasarlamak için uğraşmayacağından dikkati dağılmayacaktır, gözden kaçan herhangi bir kontrolün olmaması da garanti altına alınmış olacaktır.

Bu bildiriyle geliştirilen yazılımın mimari yapısını, teknolojisini, kullanım şekillerini ve elde ettiğimiz tecrübeleri detaylarıyla paylaşıyoruz. Bu bildirinin yazım organizasyonu şu şekildedir. 1. bölüm, problemin tanımını, araştırma problemini ve bildirinin amacını açıklamaktadır. 2. bölüm, kullanılabilecek kullanıcı arayüzü tasarım araçlarını, tasarımları başka dillere çevirebilen uygulamaları ve bu uygulamaların yeterliliklerini açıklayan literatür aramasını içerir. 3. bölüm, hazırladığımız sistemin genel mimarisini içerir. 4. ve 5. bölümler sistem üyelerinden tasarım aracı ve kullanıcı arayüzü tanımlama dili adaptörlerinin nasıl çalıştığını, giriş ve çıkış üyelerinin neler olduğunu açıklar. 6. bölüm, yapılan çalışmada kullanılan metodolojiyi ve bu metodolojinin faydalarını açıklar. 7. bölüm, çalışmanın sonucunu elde edebilmek için yapılan uygulamaya ait detayları içerir. 8. bölüm, çalışma sonucunda elde edilen sonuçları, karşılaşılan problemlerin nasıl aşıldığını ve gelecekte bu konuda nasıl bir çalıma yapılacağını açıklamaktadır.

2. Altyapı ve İlgili Çalışmalar

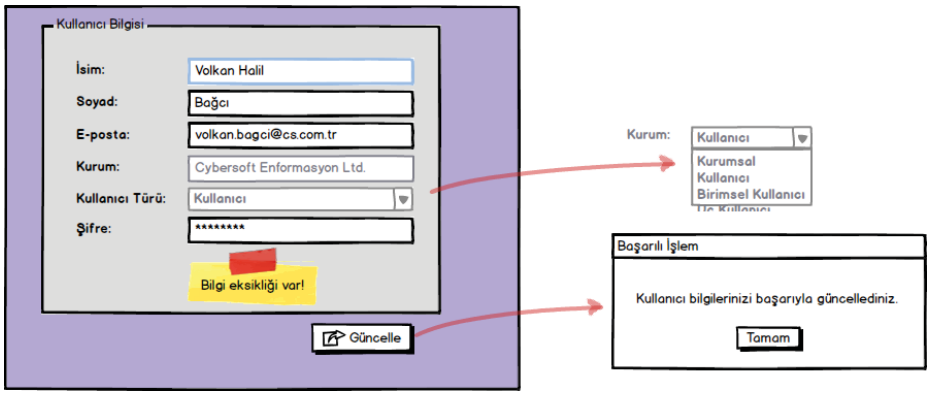
Yazılım geliştirme süreci içinde ilk adım müşteri ile problemin çözümü için ortak bir noktada anlaşmaktan geçer. Bu anlaşma sürecinde müşteri taleplerde bulunur. Gereksinim mühendisi istenilen talepleri irdeler ve uygulanabilir çözümleri müşteriye sunar [9].

Üretilecek ürünün görsel bir örneği hazırlanır ve müşterinin kabulüne sunulur. Müşteriye yapılan sunum, işlevsel olmayacak ölçüde basit, aynı zamanda müşterinin

istediği bütün detayları içerecek kadar detaylı olmalıdır. Tasarımın hazırlanması sürecinde minimum maliyet ve minimum zaman ana fikir olmalıdır [10]. Gereksinim mühendisi kendisi için pratik yöntemler kullanır, kullanımı kolay bir tasarım aracı ile kullanıcı arayüzlerini kısa sürede hazırlar ve müşteriye sunar.

Günümüzde kullanılabilecek birçok tasarım aracı mevcuttur. Çoğu tasarım aracı belirli kullanıcı arayüzlerini tasarlamak için özelleşmiştir. Bazı tasarım araçları, tasarımları sadece görsel olarak bize sunarken, bazı tasarım araçları ise, tasarlanan ekranların kendi içerisinde anlamlı kodlarını bize sunabilmektedir. Günümüzde kullanılan bir kaç tasarım aracından bahsetmek gerekirse:

- **Pencil Project:** Kullanıcı arayüzü tasarımı adına birçok bileşene sahip olduğu için kullanılabilir bilgisayar uygulaması tasarımları hazırlama araçlarından tercih edilebilecek bir araçtır. Pencil Project ile hazırlanan tasarımların anlamlı kodların ortamlar arası aktarımı da yapılabilmektedir.
- **Balsamiq Mockups:** Masaüstü uygulama tasarımları, web sayfası tasarımları ve IPhone ekran tasarımlarını hazırlama imkânı sağlar. Kullanıcı arayüzlerinin kolaylıkla tasarlanabildiği bir araç olan Balsamiq Mockups, hazırlanan tasarımların PNG ve PDF formatındaki ekran görüntülerini kaydetme imkânı sunar [5]. Ayrıca bu tasarımların DOM [11] doküman formatında çıktılarına erişilebilmektedir.



Şekil 2.1 Balsamiq ile Hazırlanan Kullanıcı Arayüzü

```
<mockup version="1.0" skin="wireframe" fontFace="Balsamiq Sans" measuredW="920" measuredH="386" mockupW="895" m
  <controls>
    <control controlID="0" controlTypeID="com.balsamiq.mockups::FieldSet" x="61" y="25" w="388" h="287" measure
      <controlProperties>
        <backgroundAlpha>1</backgroundAlpha>
        <color>14540253</color>
        <text>Kullan%u0131c%u0131%20Bilgisi</text>
      </controlProperties>
    </control>
    <control controlID="1" controlTypeID="com.balsamiq.mockups::Label" x="95" y="64" w="36" h="22" measuredW="3
      <controlProperties>
        <bold>true</bold>
        <color>0</color>
        <state>up</state>
        <text>%u0130sim%3A</text>
      </controlProperties>
    </control>
    <control controlID="2" controlTypeID="com.balsamiq.mockups::TextInput" x="203" y="63" w="220" h="-1" measu
```

Şekil 2.2 Balsamiq Tasarımına Ait BMML Dosyası

Balsamiq Mockups ile hazırlanan basit bir kullanıcı arayüzü ekranı Şekil 2.1'de verilmiştir. Örnekte de olduğu gibi Balsamiq ile yapılan tasarımlarda, ekran üzerine, kodlama yapılırken dikkat edilmesi gereken bölümler "mockup" adı ile gösterilmektedir. Tasarlayıcı, ekranları hazırladığı ve sunduğu zaman mockup'ları görünür veya görünmez olarak ayarlayabilmektedir. Bu özellik ile çizilen tasarım sunulurken de kolaylık elde edilmektedir. Çevrim işlemi için bu mockup'lar seçimli olarak dikkate alınmaktadır. Tasarımın çıktısı olan BMML dosyasının bir bölümü Şekil 2.2'de verilmiştir. Bu dosyanın içeriğinden 4. bölümde bahsedilecektir.

KATD, interaktif uygulamaların içerdiği öğelerin karakteristiklerinin açıklandığı üst düzey bir bilgisayar dilidir. KATD, sözdizimi kurallarını (dil açısından nasıl ifade edilebildiği) ve anlamsal özellikleri (gerçek dünyada ne anlama geldiği) içermektedir [12].

Microsoft ürünlerinde .Net Framework 3.0 ve üzeri versiyonları ile desteklenen XAML, deklaratif dildir. XML'e benzer bir dil olan XAML, tek başına bir uygulama için yeterli olmayıp arka tarafta C#, J# veya VB.NET ile programlaması yapılabilmektedir. Cybersoft tarafından hazırlanan Aurora Framework'u üzerinde çalışan kullanıcı arayüzü tanımlama dili olan EBML ile kullanıcı arayüzleri ve programlamaları yapılabilmektedir [13].

Tarayıcıların desteklediği KATD'ler, her hangi bir derleyici tarafından derlenmeye ihtiyaç duymadan, doğrudan işletilebilen dillerdir. HTML, javascript buna örnek olarak verilebilir.

Hazırlanan kullanıcı arayüzü tasarımlarını çalışan ortamda kullanılabilecek olan KATD'ne dönüştürebilen araçlar vardır. Bu araçlar hazırlanan tasarımları belirli bir formata dönüştürmeyi hedeflerler (HTML, EBML gibi). Yapılan araştırmada tasarımların HTML kod karşılıklarını üreten uygulamalar bulunmuş ve aşağıda sunulmuştur.

- **NedeCo Balsamiq Mockups To HTML/CSS Converter:** NedeCo Converter [14], Balsamiq Mockups ile hazırlanan tasarımları CSS ile desteklenen HTML kodlarına dönüştürmektedir. Çevrimleri başarılı bir şekilde yapan bu araç, Balsamiq Mockups dosyalarını alıp HTML kodlarına dönüştürdüğü için genel değil, sınırlı bir araçtır.
- **Napkee:** BMML dosyalarını HTML ve CSS kodlarına dönüştürebilen Napkee [15], HTML5 çıktıları üretebilmektedir. Bütün Balsamiq Mockups araçlarını dönüştürebilen Napkee, Balsamiq Mockups dışındaki başka bir tasarım aracının çıktılarına hizmet edememektedir.
- **WireFrame:** WireFrame [16], Balsamiq tasarımlarını HTML ve CSS kodlarına dönüştürebilse de basit bir tasarımı çok sayıda kontrol içeren HTML kodlarına çevirmektedir. Bu yüzden WireFrame görünüm olarak hoş bir çevirim yapsa da arka plandaki kod oldukça karmaşıktır ve geliştiriciler tarafından kullanıma çok da uygun değildir.

Yukardaki paragrafta özetlenen araçlar aracılığıyla kod dönüşümü yapılabilmektedir. Fakat kullanıcının ikinci bir dile çevirme özgürlüğü ne yazık ki yoktur. Ayrıca bu araçlar sadece belirli bir tasarım aracının çıktılarını kabul etmektedir. Bahsedilen araçlarda kullanıcı, Balsamiq Mockups dışında başka bir tasarım aracını kullanamamaktadır. Önerdiğimiz yöntem ile bu bağımlılıkları ortadan kaldırmayı hedeflemekte-

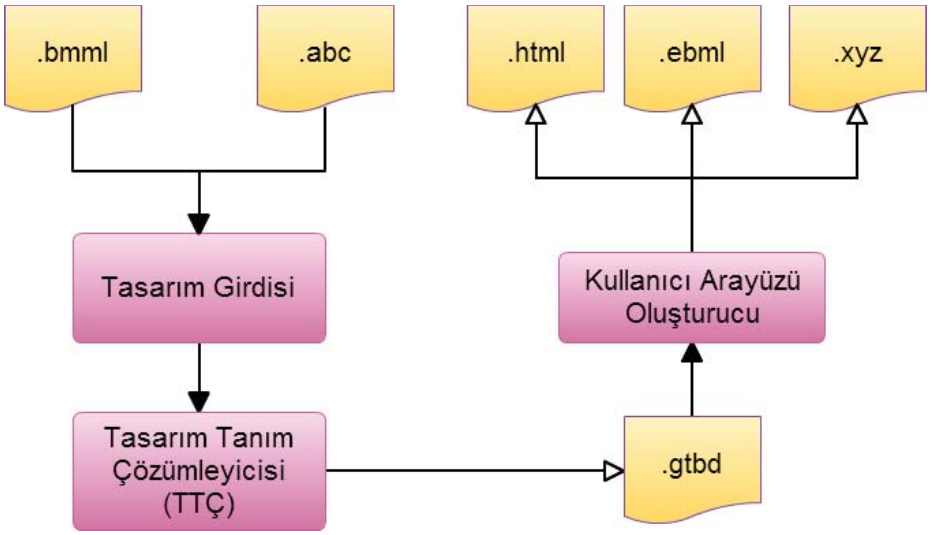
yiz. Sistemin sahip olması gereken iki temel özellik bulunmalıdır. Bunlardan ilki farklı bir tasarım aracının çıktılarının da sistem tarafından tanınabilmesi, ikincisi ise, sisteme tanıtılan tasarımın farklı bir KATD'ye dönüştürülebilir olmasıdır. Bu sayede sistem diğer çevrim araçlarına göre daha kapsamlı bir yapıya sahip olur.

3. Genel Sistem Mimarisi

Sistem yapısı genel bir kapsamı hedeflemektedir. Hedeflenen genel kapsam ile farklı tasarım araçlarından elde edilen tasarımlar okunabilmeli ve üretilecek çıktı tek bir formatta değil, başka formatlarda da olabilmelidir.

Balsamiq Mockups tasarım dosyasını denetleyen sistemde bu tasarımların içereceği her elemanın bir karşılığı olmalıdır. Dolayısıyla bankacılık sektöründeki yazılımlarda içermesi beklenen bütün kontrolleri ve özelliklerini sistem tanıyabilmelidir. Sistemin bu kontrollere bu şekilde hâkim olmasının sağladığı avantaj, bu aşamadan sonra sistemin programlandığı şekilde istenilen başka işaretleme dillerine çevrim yapabilmesidir. Bu yöntem ile birçok XML tabanlı işaretleme diline tasarım yapılabilir. Bunların başında XAML, HTML, HTML5 gibi evrensel diller olabilir. İşaretleme dillerinin avantajı, bir derleyici tarafından derlenmeden de çalıştırılabilir olmasıdır, kod yazıldıktan sonra sadece çalıştırmak doğrudan çıktı elde edilmesi için yeterli olmaktadır. Ayrıca çalışmada Cybersoft firmasının kullanıcı arayüzleri için kullandığı EBML programlama diline de çevrim yapılması hedeflenmektedir.

Hazırlanan sistem Balsamiq çıktısında bulunan XML içeriği bir başka doküman formatına doğrudan çevirmez, bunu kendisinin anladığı bir yapıya çevirir, bu yapıdan sonra çevrim için sistem hazır olur ve programlandığı şekilde çevrimi yapar. Böylece sistem bir metin dosyasını başka bir metin dosyasına çevirmekten öte, onu tanıyabildiği bir halde dönüştürmüş olur.



Şekil 3.1 Genel Sistem Mimarisi

Sistemin işleyişi Şekil 3.1'de genel olarak verilmiştir.

- **Tasarım Girdisi:** Tasarım aracından alınan dosyalar özel olarak hazırlanan TAA(Tasarım Aracı Adaptörü) ile tanınmaktadır. Tasarım aracı adaptörünün girdilerinden ".bmml" uzantılı dosyalar, Balsamiq Mockups tasarım aracının ürettiği çıktıları temsil ederken, ".abc" uzantılı dosyalar ise herhangi bir tasarım aracının ürettiği çıktıları temsil etmektedir.
- **Tasarım Tanım Çözümleyicisi:** Tasarım girdisi tarafından elde edilen veriler sistem için anlamlı olan GTBD (Genel Tasarım Biçimleme Dili) yapısına dönüştürülmektedir. GTBD, DOM formatındadır ve ortak bir yapıdır. GTBD dosyaları ".gtbd" uzantılıdır.
- **Kullanıcı Arayüzü Oluşturucu:** KATD adaptörü, GTBD dosyalarını istenilen formata dönüştürmektedir. KATD adaptörünün dönüştürdüğü formatlardan ".html" ve ".ebml", HTML ve EBML dosyalarını, ".xyz" ise geçerli olabilecek herhangi bir KATD'yi temsil etmektedir.

TAA ve KATD adaptörleri dışa dönük modüller olduğu için özel olarak tasarlanmak zorundadır.

4. Tasarım Aracı Adaptörlerinin Çalışma Sistemi

Kullanılan tasarım araçları kendisine göre anlamlı olarak nitelendirilebilen şekilde XML kodlar üretmektedirler. Bu XML kodlarını, üretildikleri tasarım araçlarına özel olarak hazırlanan adaptörler ile sisteme tanıtmak gerekmektedir. Her adaptörün, ilgili tasarım aracıyla oluşturulan XML dosyalarına ait XML Schema'da belirtilen bileşenleri tanıması gerekmektedir.

XML doküman formatı bir çeşit ağaç yapısını temsil eder. Bu ağaç yapısında iç içe üyeler bulunabilir: Bir panelin içerisinde bulunan bileşenlerin o panele ait olduğu, o panelin içerisinde yazılmasından anlaşılır veya yine bir panelin içerisinde bulunan buton, tablo gibi kontroller bu paneli tanıtan XML kodları altına yazılır. Bu hiyerarşik tasarım XML'in ebeveyn-çocuk yapısının özelliğidir.

Balsamiq, ekran çıktılarını XML tabanlı ".bmml" dosyalarında kaydeder fakat bu dosyaların diğer bilinen klasik XML dosyalarından ayrı bir yapısı söz konusudur. XML özellikleri ile birlikte bileşenleri belirtir. Balsamiq dosyaları için XML formatındaki ebeveyn-çocuk hiyerarşisinin anlamı farklıdır. Kontrollerin konumsal değerleri ebeveyn, kontrollerin kendilerine özel değerleri (Bir textbox'ın rengi, yazısının italik olması gibi) çocuk olarak belirtilmektedir. Her kontrolün ebeveyn özelliklerinin türleri aynıdır. Fakat kontrolün çeşidine göre çocuk özelliklerinin türleri farklılık göstermektedir. Bunu bir örnekle açıklamak gerekirse bir butonun da textbox'ın da konumunu ve büyüklüklerini belirten öğeler aynıdır, fakat buton ile textbox kendilerine özel farklı özellikler içerebildikleri için çocuk özellikler tipe göre farklılık gösterir.

Balsamiq dosyaları her ne kadar XML tabanlı dosyalar olsa da iç içe olan kontroller ayrı ayrı yazılmıştır. Üyeler kat kat yerleştirilmiş olarak düşünülmüş ve her üyenin kaçıncı katta oluğu "zOrder" adındaki özellikte tutulmuştur. Ayrıca her öğenin başlangıç ve bitiş koordinasyonları bilindiği için hangi üyenin hangisinin içerisinde bulunduğu koordinasyon karşılaştırması yapılarak elde edilebilmektedir.

Bir Balsamiq tasarımında bankacılık sektöründe kullanılan arayüz kontrollerini sisteme tanıtmak gerektiği için önce bu kontrollerin neler olabileceği araştırılmıştır ve kontroller aşağıdaki gibi belirlenmiştir:

- Button
- Canvas
- CheckBox
- CheckBoxGroup
- ComboBox
- DataGrid
- DateField
- FieldSet
- Label
- RadioButton
- RadioButtonGroup
- StickyNote
- TabPane
- TextArea
- TextInput
- Title
- TitleWindow

Bu kontroller kullanıcı arayüzünü tasarlarken kullanılan kontrollerdir, belirtilen kontrollerin içinde bulunan "StickyNote" tasarımın üzerine eklenen notlar için kullanılır[17]. Bu kontrol, sistem çevriminde istenildiği takdirde çevrilmeyerek yazılımcıya yardımcı olmaktadır.

Sistemin genel modeli olan GTBD dosyalarında iç içe üyeler ebeveyn-çocuk yapısı ile belirtilmektedir. Balsamiq dosyalarındaki üyeleri ebeveyn çocuk bilgileri için derinlik (zOrder) sırasından yardım alınır. Derinlik değeri en alttaki kontrol için 0 (sıfır) olarak belirlenir ve her bir kat bir fazlası olarak belirlenir. Fakat her kontrol bir diğerinin alt kontrolü olamaz, bir kontrolün ebeveyn kontrol olabilmesi için kapsayıcı kümesinden bir kontrol olması gerekir. Bu kümenin elemanları "Canvas, CheckBoxGroup, FieldSet, RadioButtonGroup, TabPane, TitleWindow" kontrolleridir ve diğer kontroller bu kontrollerin altında çocuk olarak bulunabilirler.

Sistem karşılaşacağı problemlere karşı dayanıklı olmalıdır. Karşılaşılabilecek önemli iki problem aşağıda verilmiştir:

1- Sistemde tanımsız olan bir kontrolle karşılaşılma durumu: Böyle bir durumda sistem bu kontrolün adını ve özelliklerini kayıt dosyalarında saklar ve sistemin geliştirilmesi için bir sonraki güncellemede bu kayıt dosyalarından faydalanılır.
2- Bir kontrole ait özellik sistemde tanımsız olabilir: Sistem tasarlanırken her ne kadar bütün özellikler sisteme eklenmiş olsa da Balsamiq güncel bir uygulamadır ve tasarım için yeni bileşenler hazırlanabilir. Bir kontolün özellikleri tanınmaya çalışıldığında eğer bu şekilde bilinmeyen bir özellik ile karşılaşılırsa bu durum da rapor edilir ve bu raporlar sistemin geliştirilmesine yardımcı olur.

Belirtilen durumların ikisinde de sistem çalışmasını aksatmamalı çevrim içinde devam edebilmelidir. Bu çalışma "Tanıdığın kadarını çevir, tanımadıklarını yoksay" olarak nitelendirilebilir. Sistem bu kötü senaryolar ile karşılaşırsa bunları raporlar aynı zamanda bu kontroller veya özellikler yokmuş gibi çevrim işlemine devam eder.

5. Kullanıcı Arayüzü Tanımlama Dili Adaptörü

Sistem tarafından, genel modeldeki bilgileri kullanarak kullanıcı arayüzü tasarlanmaktadır. Bu aşamaya kadar tasarlanacak ekran TAA ile tanınmış, GTBD formatında iç içe yapılandırılmış DOM dosyası hazırlanmıştır. KATD adaptörü ile GTBD dosyasındaki içerik istenilen formata çevrilmektedir. KATD adaptörü dışa hizmet veren bileşen olduğu için ortamda ihtiyaç duyulan dile göre tasarlanmaktadır.

Hazırlanan KATD adaptörü, TAA ile oluşturulmuş olan GTBD dosyasını EBML ve HTML5 dosyalarına çevirmektedir. EBML ve HTML5 dosyalarında da iç içe üyeler GTBD dosyasındaki ebeveyn-çocuk hiyerarşisinde olduğu için çevrim işleminde sıralama kontrolü gibi bir detay olmamaktadır. EBML dosyaları hazırlanırken her kontrol "bean" etiketinde hazırlanır ve özellikleri "style" etiketi altında belirtilir. Sistem tanıdığı bütün kontrolleri içeren EBML dosyasını oluşturur ve süreç tamamlanır. HTML5 dosyaları için ise yine benzer bir durum söz konusudur. Eşleştirme tanımlarını yapabilmek için XSLT[18] dosyaları oluşturulur. Sistem bu XSLT dosyalarının vasıtası ile çevrimin nasıl yapılacağına karar verir.

6. Metodoloji

Sistem Şekil 3.1'de gösterildiği gibi, dışarı dönük olan iki adaptör içermektedir. TAA dışarıdan alınan veriyi sistem için optimize ederek genel model oluşturulmasına zemin hazırlar, genel model hazırlanır ve sistem, tasarıma ait bütün bilgileri bu genel modelde istenilen dile çevrim için hazır olarak saklar. Eğer varsa bilinmeyen kontroller ve özellikler ayrı ayrı kayıt edilerek sistemin optimize edilmesi için saklanır. Daha sonra genel model, KATD adaptöründe hazırlanan metodlar ile istenilen dile çevrilir.

Sistem Facade tasarım desenine göre hazırlanmıştır. Facade tasarım deseninde yapısal gruba ait alt sistemlerin doğrudan kullanılması yerine, alt sisteme erişen başka bir nesne üzerinden alt sistem kullanılır. Facade tasarım deseni sayesinde alt sistem ile olan iletişim minimize edilir. Basit ve sade bir arayüz ile karmaşık alt sistemler daha kolay kullanılabilir. Sistem genel modele çevirerek başka dillere geçiş sağladığı için Facade tasarım desenini kullanmaktadır [19,20,21].

KATD adaptörünün kullandığı XSLT dosyaları her bir dil için ayrı olarak tasarlanmaktadır. KATD adaptörünün içerdiği dönüştürücü metodu sabittir. Sistem ile hangi dile çevrim yapılacaksa o dili tanımlayan XSLT dosyası sisteme yüklenerek çevrim işlemi yapılmaktadır.

7. Gerçekleştirme Detayları

Hazırlanan aracın bu kontrolleri ayrı ayrı tanıyabilmesi gerektiğinden nesneye dayalı modelleme kullanılmıştır. Bu tasarıma göre her bir üye tipi ayrı birer sınıf ve kendisine ait özel kontrolleri bu sınıflarda tanımlıdır. Bütün üyelerin ortak olarak kullandığı konum, büyüklük, derinlik sıralaması gibi özellikler ise bir üst sınıfta belirtildi ve kontroller bu sınıftan türetilmiştir.

Sınıfların değişkenleri tanımlanırken iç içe üyeler belirlenirken, ölçüm yapmak gerektiği için başlangıç koordinatları, en-boy ölçüleri ve derinlik değerleri "integer", diğer özellikler ise "string" türünde seçilmiştir.

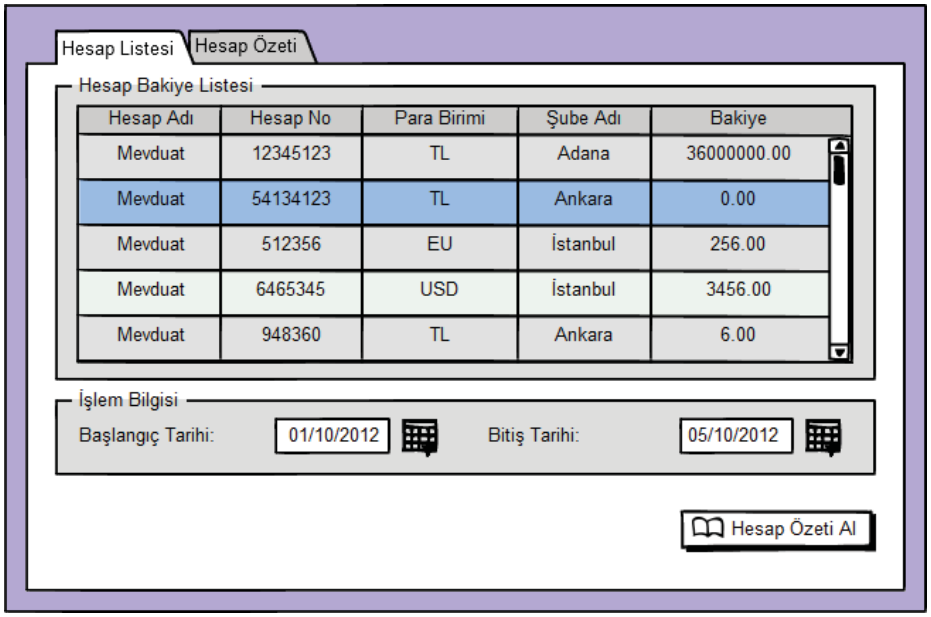
Araç java dilinde hazırlanmıştır. Balsamiq çıktılarının sisteme tanıtılması için JDK ile birlikte gelen "DOM XML parser" fonksiyonları [22,23] kodlamada büyük kolaylıklar sağlamıştır. XML tabanlı BMML dosyalarının bütün üyeleri bu fonksiyonlar kullanılarak elde edildi [24]. Sistem Balsamiq dosyalarını bu kütüphanenin fonksiyonlarını kullanarak okumakta ve dinamik olarak nesneler oluşturmaktadır. Bu nesneler sistemin elinde liste yapısı altında tutulmakta ve bu nesnelerden orta katman GTBD üretilmektedir. Bu aşamada sistem başka dile çevrim yapmadan önce orta katmanı oluşturmaktadır. Bu orta katmanda Balsamiq Mockups dosyasının içerdiği aynı zamanda sistemde tanımlı olan bütün kontroller bulunmaktadır. Tanınmayan kontroller rapor dosyalarında tutulmaktadır. Oluşturulan GTBD, sisteme tanıtılan XSLT dosyalarının referans verdiği şekilde EBML ve HTML5 dosyalarına çevrilmektedir.

Sistemin GTBD oluşturma işlemini yapabilmek için önemli noktalardan birisi ebeveyn-çocuk hiyerarşisini oluşturabilmektir. Bunu sistem derinlik değerini takip eder. Bir kontrolün ebeveyn olabilmesi için öncelikli şart kapsayıcı tipinde bir kontrol olmasıdır. Bunlar; "tab item, field set, panel, canvas, title window" kontrolleridir. Bu kontrollerden bir tanesi ile karşılaşılırsa bu kapsayıcı kontrolün derinlik değerinden daha büyük derinlik değerine sahip olan ve başlangıç ve bitiş koordinatları bu kapsayıcınn içerisinde olan diğer kontroller, bu kapsayıcının çocuk kontrolleri olarak yazılır. Bu yöntem ile ebeveyn-çocuk hiyerarşisi oluşturulmuş olur.

KATD adaptörü, java kütüphanelerinden "javax.xml.transform.Transformer" kütüphanesini kullanır. Oluşturulan transformer nesnesi XSLT dosyalarında belirtilen bilgiler doğrultusunda GTBD dosyalarından çevrim işlemini yapar. Çevrim işleminin seçimli olması planlanmıştır. Bu seçimler yazılımcıya yardımcı olması için tasarım üzerindeki notların da çevrilmesi veya çevrilmemesi olarak seçimli olabilmektedir. Bir tasarımda HTML5 çevrimi yapılacaksa bu ekran üzerindeki notlar uyarı mesajını ToolTipText olarak gösteren bir inaktif TextBox ile ifade edilecektir. EBML tasarımında yine bu notlar inaktif TextBox üzerindeki ToolTipText ile gösterilecektir.

8. Sonuçlar ve Gelecekteki Çalışmalar

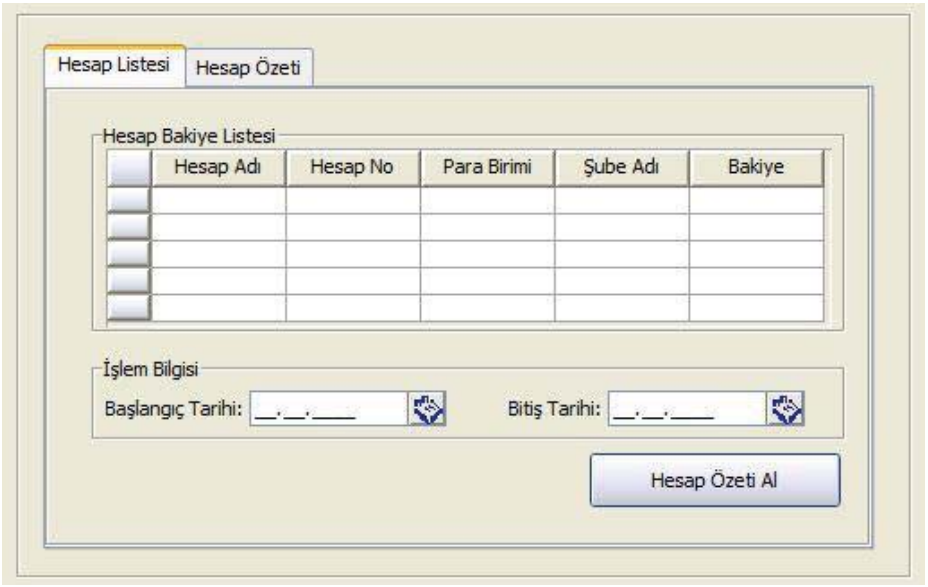
Yapılan çalışma ile oluşturulan uygulamanın testi için Balsamiq Mockups'ta bankacılık sektöründe kullanılan bir hesap yönetim ekranı tasarlanmıştır. Tasarlanan Balsamiq Mockups çizimi Şekil 8.1'de verilmiştir. Tasarlanan ekranın hazırlanan uygulama ile çevrilmesi ile elde edilen EBML karşılığının ekran görüntüsü Şekil 8.2'de verilmiştir.



Şekil 8.1 Balsamiq Mockups ile Hazırlanmış Hesap Yönetim Ekranı

Balsamiq Mockups tasarımında olup, çevrilen ekranda olmayan özellikler, tablonun içeriği (sütun başlıkları hariç) ve tarih seçim alanındaki verilerdir. Bu veriler tasarım ekranının kolay anlaşılması için eklenen veriler olduğu için çevrilmesine ihtiyaç duyulmamaktadır. Aynı zamanda EBML dilinin yapısı gereği tablo verileri EBML kodlarında değil veri tabanında olacağı için çevrilecek ekrana bu verileri eklemek gibi bir durum söz konusu değildir.

Bu ekranın GTBD çıktısı da çevrim sırasında oluşturulmuştur. Bu GTBD çıktısı elimizdeki genel formdur. Balsamiq Mockups dışında başka bir ekran kullanılacak olur ise, bu ekranın hazırlanacak yeni sürümlerinin GTBD çıktıları ile elde bulunan eski ekrana ait GTBD çıktısı karşılaştırılabilir.



Şekil 8.2 Hesap Yönetim Ekranının EBML Arayüzü

Sistemin Facade desenli tasarım olması birçok imkânı beraberinde getirmektedir. Balsamiq haricinde, Balsamiq gibi kodlarına kolaylıkla erişilebilen başka bir çizim aracı kullanılması durumunda yine bir çevrime ihtiyaç duyulacaktır. Tasarlanan çizimi bir şekilde anlatan kod, bu çalışmada oluşturulan orta katman olarak (GTBD) sisteme tanıtılabilir. Bu adımdan sonra 5. bölümde açıklanan KATD adaptörü kullanılarak çevrim tamamlanabilecektir.

7. bölümde, orta katmanda GTBD dosyaları hazırlanırken sistemin iç içe kontrolleri derinlik değerleri ile belirlediği belirtilmişti. Derinlik değeri BMML dosyalarında sıralı olarak bulunmadığı için arama yapmak gerekmektedir. Bu noktada sistemde karmaşıklık hesabı yapmak gerekir. Derinlik ile koordinat değerleri bilinen ve gözden geçirilmemiş olan kontrollerin hepsi aranarak, ebeveyn kapsayıcının altına yazılmaktadır. Kontrollerin koordinat değerleri sırasız olduğundan dolayı karmaşıklığı O(n)=n olan "linear search" kullanılmıştır. Hazırlanacak bir ekranda kontrol sayısının en fazla 80-100 olacağı varsayıldığında bu işlem günümüz bilgisayarlarını çok zorlamayan bir işlem olmaktadır. Bir çalışmada hazırlanan ekran çıktılarını başka bir işaretleme diline çevrilmek istenildiğinde gereken XSLT dosyasının hazırlanarak sisteme tanıtılması yeterli olacaktır.

Önerdiğimiz bu yaklaşımın bir sonraki adımı olarak hazırlanan ekran tasarımlarını "Ext JS" [25] kodlarına dönüştürmeyi hedeflemekteyiz. "Ext JS", Sencha firmasının programlanabilir dinamik web uygulamaları geliştirmek için hazırladığı javascript kütüphanesidir.

Kaynaklar

1. Harris, B., John, B.E., Brezin, J.: Human Performance Modeling for All: Importing UI Prototypes into CogTool. CHI '10. ACM, pp. 3481-3482, Atlanta, GA, USA (2010)
2. Pencil Project, http://pencil.evolus.vn/Default.html
3. FlairBuilder - Wireframes. Mockups. Prototypes, http://flairbuilder.com/
4. Balsamiq Mockups- Balsamiq, http://balsamiq.com/products/mockups/
5. Faranello, S.: Balsamiq Wireframes Quickstart Guide. pp 13, Packt Publishing, Birmingham (2012)
6. Altintas, N. I., Surav, M., Keskin, O., Cetin, S.: Aurora software product line. Turkish Software Architecture Workshop, Ankara (2005)
7. Altintas, N. I., Cetin, S., Dogru, A. H., Oguztuzun, H.: Modeling Product Line Software Assets Using Domain-Specific Kits. IEEE transactions on software engineering 38(6): 1376-1402 (2012)
8. Martin J.: Rapid Application Development. Macmillian (1991)
9. Misic, M. M.: Journal of Systems Management. 47, 34-40 (1996)
10. Wood-Harper, A. T., Corder, S., Wood, J. R. G., Watson, H.: How we profess: the ethical systems analyst. Communications of the ACM, Volume 39, Issue 3, pp 69-77, New York, NY, USA (1996)
11. Document Object Model (DOM) Level 3 XPath Specification, http://www.w3.org/TR/DOM-Level-3-XPath
12. Souchon, N., Vanderdonckt, J.: A Review of XML-compliant User Interface Description Languages. DSV-IS 2003, LNCS 2844, pp. 377-378, Springer-Verlag Berlin Heidelberg (2003)
13. Lori MacVittie, XML in a Nutshell, pp.3 , O'Reilly Media Inc. (2006)
14. Balsamiq Mockups to HTML/CSS Converter, http://development.nedeco.de/blog/2011/11/03/balsamiq-mockups-to-htmlcss-converter/
15. Napkee – Make Your Mockups Come Alive, http://www.napkee.com/
16. Wire2app – Beta, http://www.wire2app.com/product.aspx
17. Faranello, S.: Balsamiq Wireframes Quickstart Guide. pp 101, Packt Publishing, Birmingham (2012)
18. Burke, E.M.: Java and XSLT. pp. 135-142, O'REILLY (2001)
19. Mel Ó Cinnéide, Paddy Fagan. Design Patterns: the Devils in the Detail. pp. 6-8, PLoP '06 Proceedings of the 2006 conference on Pattern Languages of programs. ACM, Article No. 33, New York, NY, USA (2006)
20. Pandey, R. K.: Managing software design complexity: facade vs role-based design. ACM SIGSOFT Software Engineering Notes, Volume 34 Issue 1, pp. 1-4, New York, NY, USA (2009)
21. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. pp. 208-217, Addison-Wesley (1995)
22. Reading XML Data into a DOM (The Java™ Tutorials > Java API for XML Processing (JAXP) > Document Object Model), http://docs.oracle.com/javase/tutorial/jaxp/dom/readingXML.html
23. Chen, H., Tompa, F.Wm.: Set-at-a-time Access to XML through DOM. Proceedings of the 2003 ACM symposium on Document engineering, pp. 225-233, New York, NY, USA (2003)
24. Harold, E.R.: Processing Xml with Java, Addison-Wesley Longman Publishing Co., chapters 9-13, Inc. Boston, MA, USA (2002)

25. JavaScript Framework for Building Rich Desktop Web Applications | Sencha Ext JS | Products | Sencha, http://www.sencha.com/products/extjs/