Web Uygulamaları için Model Tabanlı Çevik Süreç Yöntemi ile Yazılım Geliştirme

Gürkan Alpaslan[1] ve Oya Kalıpsız[2]

[1,2] Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye
[1] gurkana@yildiz.edu.tr
[2] oya@ce.yildiz.edu.tr

Özet. Model tabanlı geliştirme, sağladığı dokümantasyon, otomatik kod dönüşümü ve yüksek soyutlama düzeyi ile platform bağımsız geliştirme sağlaması ile yazılım mühendisliğinde avantajlar sunmaktadır. Bu yaklaşımın çevik süreç prensipleri ile birleştirilerek uygulanmasının, yazılım geliştiriciye, iki yöntemin avantajlarından faydalanarak geliştirme imkanı sunacağı düşünülmektedir. Bu bildiri çalışmasının amacı, web uygulamalarına özel bir çevik model tabanlı yaklaşım önermek ve yaklaşımın uygulanabilirliğini, zorluklarını ve avantajlarını değerlendirmektir.

Anahtar Kelimeler: Mockup tabanlı yazılım geliştirme, Web uygulamalarında yazılım geliştirme, Model tabanlı geliştirme

1 Giriş

Model tabanlı yazılım geliştirme, programın kaynak kodları yazılmadan önce, yazılım modellemelerinin yapılmasını öngören bir yaklaşımdır [1]. Model tabanlı geliştirme, gelişmiş programlama dillerine göre daha yüksek bir soyutlama düzeyinde yazılım geliştirme olanağı sağlar [2]. Bu avantaj, platformdan bağımsız yazılım geliştirilmesine [3] olanak sunmaktadır. Geliştirici, yazılım platformunu düşünmeden sistem için gerekli modellemeleri yapar; modellemelerin sonrasında sisteme uygun programlama dilleri ile kodlama aşamasına geçer. Model tabanlı geliştirmenin bir diğer avantajı da, modelden istenilen platforma göre kod dönüşümünü sağlayan yazılım geliştirme araçlarından faydalanabilmesidir. Otomatik kod dönüşümü ile, kodlama aşamasının büyük bir kısmı yardımcı araçlara bırakılarak, modeller üzerinden çalıştırılabilir yazılım parçalarına ulaşılabilmektedir. Kodlamayı yardımcı yazılım araçlarına bırakmak, maliyet, hız, hata oranını azaltma gibi bir çok avantajı da yanında getirmektedir.

Çevik süreç ile yazılım geliştirme, yazılımın parçalara bölünerek tekrarlı olarak geliştirilmesini sağlamaktadır [4]. Çevik süreçte yazılım dokümantasyonu ikinci plana bırakılarak, çalıştırılabilir yazılıma ulaşmak hedeflenir [5]. Çevik süreç prensibine uygun olarak model tabanlı geliştirme yapmak, iterasyonel ve arttırımsal olarak yazılımın, üretilen modellerden geliştirilmesini gerektirir [6]. İki yöntemin bir arada kullanılmasındaki amaç, iki yazılım sürecinin avantajlarından faydalanmaktır.

Geliştirilen modeller, yazılım dokümantasyonu olarak kullanılarak, çevik sürecin dokümantasyon eksiği giderilmektedir. 2006 yılında, askeri projelerde ve çok ajanlı sistemlerde kullanılmak üzere geliştirilen, bir çevik model tabanlı süreç olan Sage yönteminde [7] iki süreç birleştirilerek kullanılmıştır. Çok ajanlı sistemlerin kısa adımlar ile geliştirilmesi gerekliliği tekrarlı yazılım geliştirme yapılmasına ihtiyaç duymaktadır. Ancak bu sistemlerde dokümantasyon gereksinimi de güvenlik ve geliştirmenin ilerleyebilmesi açısından zorunludur. Bu zorunluluktan dolayı dokümantasyon imkanı da sağlayan çevik süreç yaklaşımı için, model tabanlı geliştirme, çevik süreç prensipleri ile bir arada kullanılmıştır. Literatürde iki yöntemi birleştiren farklı yöntemler mevcuttur. Her yöntemin süreç akışı, geliştirilen sistemin yapısı ve geliştirilme amacına göre de farklılıklar içermektedir. Literatürde, iki yöntemi birleştiren yöntemler, bildirinin ikinci bölümünde incelenmiştir. Ancak bu yöntemleri incelediğimizde, sadece web uygulamalarına özel, çevik süreç ile model tabanlı süreci birleştiren bir yöntem uygulanmamıştır. Bu bildirinin amacı, literatürdeki farklı yöntemlerden faydalanarak, web uygulamalarına özel bir çevik model tabanlı süreç yaklaşımı sunmaktır.

Web uygulamalarında model tabanlı geliştirme [8,9] mockup'lar ile uygulanabilmektedir. Mockup, web uygulamalarının modelleridir. Bir web sayfasındaki her element ve mimari yapısı mockup araçları ile tasarlanabilmektedir. Bu tasarımlar üzerinden kod geliştirilmesi mockup tabanlı yazılım geliştirme olarak tanımlanır. Mockup tabanlı geliştirme araçları ile tasarımın yanında web uygulamaları için otomotik kod dönüşümü de yapılabilmektedir. Bu dönüşüm mockup'tan HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) ve JavaScript kodlarına doğrudur. Bir diğer avantaj, mockup'lar ile gereksinim analizinin kolaylaştırılmasıdır. Görsel formatta hızlı geliştirme, sistem analizinde avantaj sağlamaktadır.

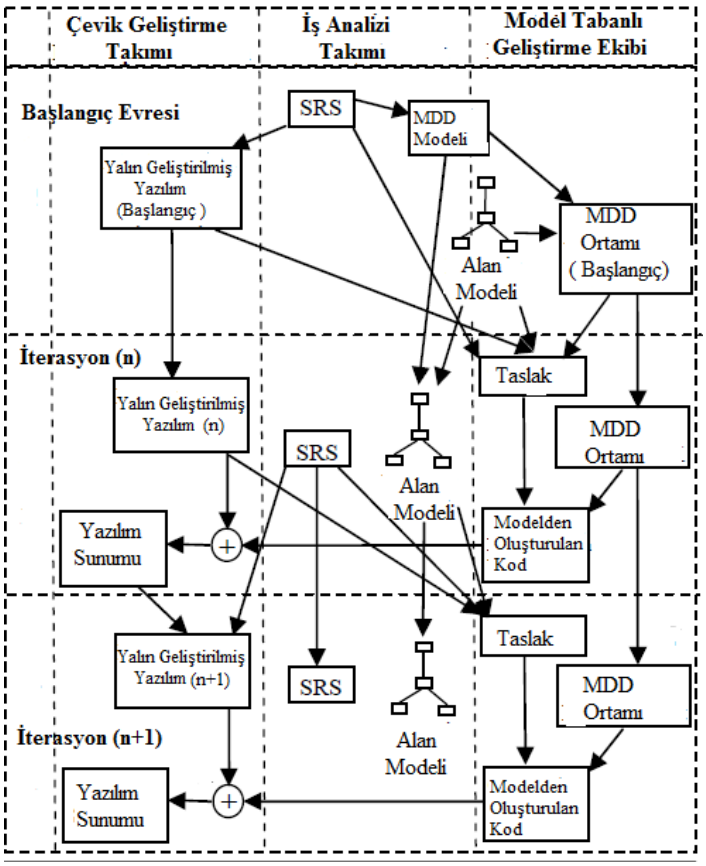
Bildirinin ikinci bölümünde literatürde bu konu üzerine yapılan çalışmalar incelenmiştir. Yapılan çalışmalardan yola çıkılarak web uygulamaları için oluşturulan yaklaşımımız bildirinin üçüncü bölümünde anlatılmıştır. Bildirinin dördüncü bölümünde, bu yaklaşımın öğrenci projelerinde uygulama çalışması tanıtılmıştır. Son olarak, bildirinin beşinci bölümünde değerlendirme ve sonuçlar anlatılmıştır.

2 Literatür Özeti

Literatürde bu konu ile ilgili ilk çalışma 2004 yılında yapılmıştır. Bu çalışma, iki sürecin birlikte uygulanabilmesi için, uyulması gereken genel prensipleri belirlemekte ve süreç akış şeması sunmaktadır. Bu yöntem, üst seviye yaşam döngüsü [10,11] olarak tanımlanmıştır. Yöntem, başlangıç ve geliştirme olarak iki ana evreden oluşmaktadır. Başlangıç evresi, proje kapsamının, amacının belirlendiği ve başlangıç gereksinimlerinin modellendiği aşamadır. Ayrıca bu aşamada, sistemin genel mimari yapısı da modellenmektedir. Tüm çevik model tabanlı süreçlerde olduğu gibi, bu aşamanın prensiplerine uygun olarak uygulanması, klasik yöntem ile geliştirmeye alışmış kişiler için zordur. Çünkü bu aşamada, modellemelerin sadece genel hatlar üzerine yapılması, mimarinin işleyişinin genel anlamı ile çıkarılıp detaya girilmemesi önemlidir. Bu aşama, iterasyon 0 ya da ilk modelleme aşaması olarak da isim-

lendirilir. Gereksinim aşamasında gerçeklenecek iterasyonların belirlenmesi genel amaçtır; daha fazlası çevik sürecin tam anlamı ile uygulanabilmesine zarar vermektedir. Gereksinim aşaması, iterasyonların gerçeklendiği aşamadır ve iterasyon modellemesi, model gözden geçirme ve test tabanlı yazılım geliştirme [12] olarak üç ana aktiviteden oluşmaktadır.

2009 yılında üst seviye yaşam döngüsünün daha geliştirilmiş bir hali olarak değerlendirilebilecek Hybrid model tabanlı geliştirme (Hybrid-MDD) [13] yöntemi önerilmiştir. Bu yöntemin temel farkı işleyişte birbirleri ile iletişimi kuvvetli ve paralel çalışan ekipleri de süreç akışı içerisinde tanımlamasıdır. Bu ekipler, iş analizi ekibi, çevik geliştirme ekibi ve model tabanlı geliştirme ekibidir. Ekipler kendi alanlarındaki görevleri gerçekleştirmekte ve mümkün olduğunca birbirlerini beklemek yerine paralel çalışarak süreci hızlandırma prensibine dayanmaktadır. İş analizi ekibi, gereksinim analizi ve müşteri ile iletişimden sorumludur. Model tabanlı geliştirme ekibi, modelleme altyapısı ve modelden dönüştürülen kod kısmından sorumludur. Çevik geliştirme ekibi, el ile geliştirilen koddan sorumludur. Hybrid-MDD yönteminin süreç akışı Şekil 1'de gösterilmiştir.



Şekil 1. Hybrid-MDD süreç akışı [13,14]

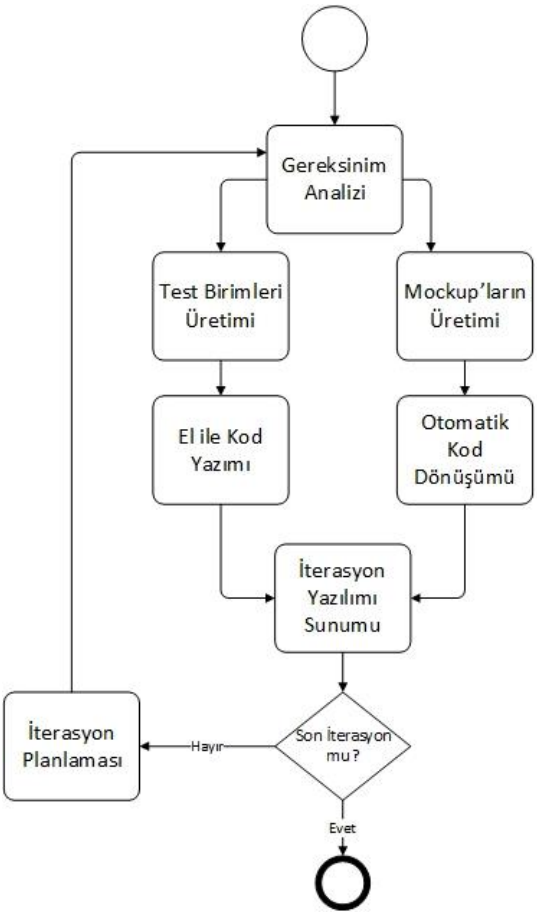
2006 yılında çok ajanlı sistemlerde kullanılmak üzere Sage geliştirme yöntemi ve 2011 yılında Motorola'nın şirket içi projelerinde kullandığı çevik yöntemi geliştirmesi ile oluşturulan SLAP-MDD [15] yöntemi de literatürde bulunan birer çevik model tabanlı yöntemlerdir. Ancak, web uygulamaları için yaklaşımımızı oluştururken temel aldığımız yöntem Hybrid-MDD yöntemidir. Bu yöntemi seçmemizin sebebi üst seviye yaşam döngüsüne göre daha geliştirilmiş bir yöntem olması ve küçük ölçekli projeler için tasarlandığından dolayı öğrenci gruplarında uygulanabilme imkanı sunmasıdır. Bu yöntemdeki süreç işleyişi iskelet olarak alınarak, web uygulamaları için özelleştirmeyi hedefledik. Kendi yaklaşımımızın Hybrid süreçten temel farkı bir web geliştirme yöntemi olan mockup tabanlı geliştirme prensibini uygulamasıdır. Bu yöntemin uygulaması mockup tabanlı geliştirme yönteminin kendi yardımcı geliştirme araçlarını sürece adapte etmeyi de zorunlu kılmıştır. Yaklaşımda mockup'lardan sadece tasarım yönünde değil, müşteri ile gereksinim kabulü almada ve otomatik kod dönüşümü ile kodlama aşamasında da faydalanılmaktadır. Ayrıca hybrid süreç genel bir bakış açısına göre tasarlanmışken, yaklaşımımız görüşü küçülterek web uygulamaları ile sınırlamakta, bu sayede genel projeler yerine, web uygulamalarının gereksinimleri göz önüne alınarak sürecin ilerlenmesi sağlanmaktadır.

3 Web Uygulamalarında Çevik Model Tabanlı Yaklaşım

Web uygulamalarında programlama, istemci ve server taraflı olarak yapılmaktadır. Yaklaşımımızda HTML,CSS ve JavaScript gibi modelden dönüştürebileceğimiz istemci taraflı programlama dilleri kullanılmaktadır. Ayrıca mimari tasarım da modeller üzerinden uygulanarak, analizin daha sağlıklı yapılabilmesi sağlanmaktadır. Çünkü mockup'lar görsel arayüze işlev ekleme mantığı ile çalışır ve bu avantaj, gereksinim analizi ve mimari tasarımda kullanılarak yararlanılabilmektedir.

Yaklaşımda, Hybrid-MDD yöntemini iskelet olarak kullanıyoruz. Bu süreç akışı üzerinde mockup tabanlı geliştirme uyguladık. Hybrid mockup tabanlı geliştirme olarak isimlendirdiğimiz bu yöntemin süreç akışı Şekil 2'de gösterilmiştir. Şekilde her aktivite dikdörtgen şekiller ile gösterilmiştir. Hybrid yönteminde olduğu gibi geliştiriciler çevik geliştirme ekibi, model tabanlı geliştirme ekibi ve iş analizi ekibi olmak üzere üç ekibe ayrılmaktadır. İş analizi ekibi, müşteri ile iletişimden ve gereksinim analizinden sorumludur. Model tabanlı geliştirme ekibi, geliştirilecek sisteme uygun mockup geliştirme aracının seçilmesi, mockup altyapısının kurulması ve otomatik kod dönüşümünden sorumludur. Mockup'lar web sayfalarının tasarımlarıdır. İlk aşama mockup'lar üretildiğinde, oluşturulacak web sayfalarının görsel tasarımları elde edilmiştir. Mockup tabanlı geliştirme araçlarının kattığı özellik, web sayfalarındaki her bir elemana eylem atayabilmesidir. Örneğin, web sayfası yüklenirken yeni bir pencere açılması isteği ya da bir butona tıklandığında web sayfasına yeni bir resim ekleme eylemi bu araçlar ile kod yazmadan üretilmektedir. Gerekli kaynak kodlar, otomatik kod dönüşümü ile araç tarafından oluşturulmaktadır. Ancak bu kodlar, istemci taraflı kodlarla sınırlıdır. İlişkisel bir veritabanı ile bağlantı oluşturulması gibi server taraflı programlama ise, yaklaşımımızda el ile oluşturulan kodlar olarak tanımlanmıştır. El ile oluşturulan kodlar, otomatik kod dönüşümü ile oluşturulan ko-

dların üzerine eklenmesi öngörülmüştür. Çevik geliştirme takımı, el ile oluşturulan kodlardan sorumludur. Ayrıca test birimlerinin oluşturulması da çevik takımın sorumluluğundadır. Yaklaşımda test tabanlı geliştirme önerilmiştir. Gereksinim analizinden sonra, çevik takım gereksinimlere uygun test birimlerini oluşturmaya başlar. Aynı süre dolayında paralel olarak, MDD takımı mockup'ları üretmek ile uğraşmaktadır. Paralel çalışma ile sürecin hızlandırılması ve ekiplerin birbirlerini bekleme sürelerinin kısaltılması hedeflenmiştir. Bu sırada çevik prensibi olarak müşteriyi de sürece katmak önemlidir. İş analizi ekibi sürekli olarak müşteri ile iletişim halinde olarak, üretilen web modellerini müşteriye gösterir ve geri bildirimleri ekip ile paylaşır. Bu geri bildirimlere göre gereksinim ve mockup'lar güncellenebilir. Otomatik kod dönüşümü ile elde edilen kodlara, server tabanlı kodlar, çevik takım tarafından birleştirilir. Üretilen test birimlerine göre kaynak kod test sürecinden geçirilir ve iterasyon yazılımı ortaya çıkar.



Şekil 2. Hybrid mockup tabanlı geliştirme süreç akışı

İterasyon yazılımı müşteriye sunulur ve iterasyon kazanımları dokümante edilir. Bir sonraki iterasyonu planlamak üzere tüm ekip toplanarak gelecek iterasyon planlaması yapılır. Bu süreç iterasyonlar tamamlanıp, yazılım son halini alana kadar devam eder.

4 Yaklaşımın Diğer Yöntemler ile Karşılaştırılması

Yaklaşımımızı üç farklı grup yöntem ile karşılaştırabiliriz. Yaklaşımın geleneksel sürece göre karşılaştırılması, sadece model tabanlı geliştirme ile karşılaştırılması ve sadece çevik süreç ile karşılaştırılması bu grupları oluşturmaktadır.

Geleneksel süreç, uzun analiz ve tasarım aktiviteleri içermektedir. Bu aktiviteler tamamlanana kadar kodlamaya geçilmemekte ve doğrusal bir yapıda tekrarsız olarak aktiveler birbirini izlemektedir. Bizim yaklaşımımızda çevik sürecin gereği olarak aktivitelerin tekrarlı olarak geliştirilmesi ve arttırımlı programlama esas alınır. Ayrıca test tabanlı yazılım geliştirme de önerilmektedir. Test süreçleri yazılım kodlamasından önce gerçekleştirilmektedir. Geleneksel süreçte geniş dokümantasyon imkanı sunmakla birlikte, dokümantasyonu modele bağlamamaktadır. Bizim yaklaşımımız yeterli dokümantasyon sağlamaktadır; ancak çevik modellemenin gereği olarak dokümantasyon olarak modelleme yapılmaktadır. Geleneksel süreç dokümantasyon temelli bir yazılım geliştirirken, yaklaşımımız model tabanlı geliştirme yapmaktadır; ancak modeli değil, çalışan yazılımı hedef almaktadır.

Sadece model tabanlı geliştirmeyi web uygulamaları için düşünürsek, Şelale modelini izleyen mockup tabanlı geliştirme ortaya çıkar. Yine geleneksel sürecin çevik sürece karşı olan süreç içerisindeki değişime uyum problemi, paydaşların sürece katılım eksiği gibi dezavantajları bünyesinde barındırmaktadır. Model tabanlı geliştirmenin sağladığı görsel analiz üzerinden kod üretme, arayüz tasarım kolaylığı ve arttırılmış anlaşabilirlik gibi avantajlar ise yaklaşımla ortaktır.

Sadece çevik süreç ile karşılaştırdığımızda, model tabanlı geliştirme prensibini kullanmayan tekrarlı ve arttırımsal bir süreç modeli düşünmeliyiz. Bu model ile çevik modelleme prensipleri arasında üçüncü bölümde tanıttığımız SLAP-MDD yaklaşımında ayrıntısıyla ortaya koyan sıkı bir bağ vardır. Bu anlamda aktivitiler birbirleri ile ilişkilidir; ancak çevik modellemede prensiplerin model üzerinde uygulanması düşünülmektedir. Örneğin çevik sürecin ikili programlama prensibi ikili modelleme olarak uygulanmaktadır. Çevik sürecin arttırımsal programlama prensibi ise, arttırımsal modelleme olarak uygulanmaktadır. Ayrıca otomatik kod dönüşümü ile düşük hata olasılığı ve zaman kazanımını, sadece çevik süreçte sağlayamayız.

5 Uygulama Çalışması

Önerdiğimiz yöntemi web uygulamalarında kullanmak üzere, üniversitemizin yazılım mühendisliği dersinde öğrencilerin dönem projeleri üzerinde denedik. Sağlıklı deney yapabilmek üzere, farklı konularda iki proje grubu seçtik. Ekiplerden biri online sinema bilet sistemi tasarlarken, diğer öğrenci grubu online dil kursu kayıt sistemI üzerinde çalışmıştır. Öğrenci grupları beşer kişilik ekipten oluşmaktadır ve

genel programlama bilgilerine sahiptirler. Öğrencilere öncelikli olarak yaklaşımımız tanıtıldı. Öğrencilerden üç gruba ayrılmaları ve çevik, iş analizi ve model tabanlı ekip olarak iş paylaşımı yapılması istendi. Her ekibin kendi alanındaki konuları yapması ve geliştirmede birbirleri ile iletişimin kuvvetli olması istenmiştir. Haftalık olarak toplantı gerçekleştirerek, sürece uygun olarak geliştirmeleri ve ilerlemeler takip edilmiştir. Mockup tabanlı geliştirme aracı olarak Axure [16] aracı seçildi. Bu aracın seçilmesinde otomatik kod dönüşümüne olanak sağlaması ve kullanımının basitliği kriterleri göz önüne alınmıştır.

Test yöntemi olarak kara kutu test tekniğinin model tabanlı web uygulamaları için uygun olduğuna karar verildi. Kara kutu test tekniği, uygulamanın kodlarının test edilmesi yerine, çalışan yazılımın doğru çalışıp çalışmadığının kontrol edilmesi prensibine dayanır. Otomatik kod dönüşümü ile el ile yazılan kodun azlığı ve kod üretiminin büyük kısmının araçlara bırakılmasından dolayı bu yöntem, hızlı test süreci geçirilmesi için seçilmiştir. Projelerin gerçeklenmesi üç aylık bir sürede tamamlanmıştır.

6 Değerlendirme ve Sonuçlar

Uygulama çalışmasında öğrencilerin yaşadığı en büyük sıkıntı, otomatik kod dönüşümü ile oluşturulan kodlar ile sunucu tabanlı kodların entegrasyonudur. Mockup tabanlı araçlar, dönüşüm sonucunda beklenenden çok uzun kod parçası üretmiştir. Basit bir web sitesi tasarımı için binlerce satır HTML,CSS ve JavaScript kodu üretmiştir. Bu uzunluğun sebebi en basit bir gösteriminin bile ayrı ayrı etiketler içerisinde tanımlanarak dönüşüm yapılmasıdır. Aslında basit HTML ifadelerinden oluşan bu kod yığını, öğrenciler için karmaşıklık yaratmıştır. Bizim bu konudaki önerimiz, koda değil, modele odaklanmalarıdır. Kod üzerinde geliştirme yapmaya alışkın öğrenciler için modeller ile geliştirmeye adapte olup, eski uyguladıkları yöntemleri bırakmaları ve alışmaları kolay olmamıştır. Bu sebeple bu yöntemin düzenli takip edilme ihtiyacı vardır. Kod birleşimi için, etiketler ile HTML içerisine server tabanlı kod parçalarının yerleştirmeleri önerilmiştir. Bunun için örneğin "<% %>" etiketleri kullanılarak ilgili yerlerde ASP.NET programlama dili kullanmaları tavsiye edilmiştir. Ayrıca karmaşıklığı azaltmak amacıyla çoklu modelleme prensibinin uygulamalarını tavsiye ettik. Yapılacak modeli ya da web uygulaması için mockup'ı tasarlarken büyük parçalar halinde tasarlamak yerine küçük model parçaları halinde tasarlamaları istenmiştir. Bu sayede her küçük model kendi içinde gerçeklenerek bir anda otomatik kod dönüşümünden ortaya çıkan kod karmaşısının anlaşılabilirliği arttırılmıştır.

Yaklaşım ile görsel olarak programlama yapılmış, basitlik yönünden beğenilmiştir. Sistem analizi görsel tasarım ile yapılarak gereksinimlere uygun olarak geliştirme yapılması sağlanmıştır. Otomatik kod dönüşümü ile kodlama aktivitesi kısa bir süreye indirgenmiş ve yazılım geliştirme hızı arttırılmıştır. Bu hız arttırımını bir kaç sebebe dayanmaktadır. Uzun kod parçaları modeller ile birkaç dakika da ifade edilebilmektedir. Bu modelin oluşturulmasında kullanılan model tabanlı geliştirme araçları mimari hatalara izin vermeyerek, modelin mimari yapısının doğruluğunu desteklemektedir.

Ayrıca otomatik kod dönüşümü ile kodlama daki insan faktörü ortadan kaldırılarak daha hızlı sonuca ulaşma sağlanmaktadır. Bu sayede hatalı kod yazımı ve tespiti gibi sıkıntılar minimalize edilmiştir. Bu anlamda test aşamasının da hızlandığı söylenebilmektedir.

Sonuç olarak, yaklaşımın küçük ölçekli web uygulamalarında uygulanabilirliği kanıtlanmıştır. Ancak büyük ölçekli projelerde uygulanabilirliği araştırmaya açık bir alandır. Ayrıca koddan mockup'a dönüşüm gibi tersine mühendislik çalışması da incelenebilecek çalışmalardandır.

Kaynaklar

1. Stahl, T., Volter, M.: Model-Driven Software Development. John Wiley & Sons, (2006)
2. Vale, S., Hammoudi, S.: Context-aware model driven development by parameterized transformation. In: Proceedings of MDISIS, pp. 167-180, (2008)
3. Nguyen, V.C., Qafmolla, X.: Agile Development of Platform Independent Model-Driven Architecture. In: Third International Conference on Information and Computing vol.2, pp.344-347, (2010)
4. Dyba, T., Dingsory, T.: What Do We Know about Agile Software Development?. IEEE Software, pp. 6-9 (2009)
5. Cockburn, A.: Agile Software Development. Addison-Wesley, Boston, (2002)
6. Ambler, S.W.: The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.0, Cambridge University Press, New York (2004)
7. Kirby J.: Model Driven Agile Development of Reactive Multi Agent Systems. In: Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), (2006)
8. Benson, E.: Mockup Driven Web Development. In: 22nd Intenational World Wide Web Conference, pp. 337-342, (2013)
9. Basso, F., Pillat, M., Frantz, F.R., Frantz, R.Z.: Study on Combining Model-Driven Engineering and Scrum to Produce Web Information Systems. In: 16th International Conference Enterprise Information Systems, pp. 137-144, (2014)
10. Ambler, S.W.: Agile Model Driven Development. XOOTIC Magazine, (2007)
11. Agile Model Driven Development, http://agilemodeling.com/essays/amdd.htm, 10.05.2015
12. Astels, D.: Test Driven Development: A Practical Guide. Prentice Hall, (2003)
13. Guta G., Schreiner W., Draheim D.: A Lightweight MDSD Process Applied in Small Projects. In Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications, IEEE, (2009)
14. Matinnejad R.: Agile Model Driven Development: An Intelligent Compromise. In: Software Engineering Research, Management and Applications (SERA), 9th International Conference on, pp. 197-202, (2011)
15. Zhang Y., Patel S.: Agile Model-Driven Development in Practise. IEEE Software, (2011)
16. Interactive Wireframe Software & Mockup Tool, http://http://www.axure.com, 10.05.2015