Çevik Süreç ile Model Tabanlı Yazılım Geliştirme
Model Driven Software Engineering with Agile Process

Gürkan Alpaslan, Oya Kalıpsız

Elektrik-Elektronik Fakültesi
Yıldız Teknik Üniversitesi
gurkana@yildiz.edu.tr

Elektrik-Elektronik Fakültesi
Yıldız Teknik Üniversitesi
oya@ce.yildiz.edu.tr

Özet

Model tabanlı yazılım geliştirme süreci, yazılımın kaynak kodları yazılmadan önce yazılım modellerinin oluşturulmasını öngören bir süreçtir. Çevik süreç ile yazılım geliştirme müşteriyi de sürecin içine katan, yazılımın iterasyonlara bölünerek döngüler halinde geliştirildiği bir yöntemdir. Çevik süreci, model tabanlı olarak geliştirmek yazılımın kısa aralıklar ile hızlı bir şekilde geliştirilmesi sağlarken, süreç içerisinde dokümantasyonun da üretilmesine olanak verir. Bu iki süreci birleştirmek birbirinin eksik yönlerini kapatırken, yazılım oluşturma süresini, üretkenliğini ve yazılım kalitesini arttırmaktadır. Bu çalışmamızda, model tabanlı çevik süreç ile uygulama geliştirme aşamaları tanıtılmış, bu yöntem ile yazılımın nasıl geliştirileceğini ve süreç adımlarını gösteren bir uygulama gerçekleştirilmiştir. Geleneksel yöntem ile model tabanlı çevik süreç arasında kıyaslama yapılmış, farklılıkları belirtilmiştir.

Abstract

Model driven software development process means creating the software models before writing the source code. Software development with agile process is a method which interact with the customer through the process and developing the software with iterations. To develop agile process with model driven provide to develop software fast and also provide to creating the software documentation. Combining this two process decrease the their disavantage sides and increase the development time, productivity and software quality. In this study, introduced the segments of agile model driven development and a case study by this methodology implemented. Traditional method and this method compared and the differences are specified.

1. Giriş

Çevik süreç ile yazılım geliştirme kısa aralıklar ile yeni sürümler çıkartması ve esnek yapısıyla önemli bir yere sahiptir. Bu yaklaşım ile yazılım geliştirmede temel amaç müşterinin istediği çalışır halde bir ürünü seri bir şekilde üretebilmektir. Bu sebeple çevik süreç içerisinde dökümantasyon genel olarak ikinci planda kalmaktadır [1].

Model, bir sistemin soyut bir şekilde gösterimidir. Soyutlama yazılımın daha anlaşılır olmasını ve problemlere karşı daha etkin çözüm üretilmesini sağlamaktadır. Model tabanlı yazılım geliştirilmesi herşey modeldir yaklaşımına göre şekillenmektedir.

Bu bildirinin amacı, model tabanlı sürecin, çevik değerler, prensibler ve tekniklere uygun olarak geliştirilmesini gösterebilmektir. Bu bildirinin ikinci bölümünde model tabanlı yaklaşım tanıtılmış, üçüncü bölümde model tabanlı çevik süreç olarak belirttiğimiz bu yapı tanıtılmış ve aşamaları gösterilmiştir. Çalışmamızın dördüncü bölümünde bu süreç ile bir uygulama geliştirilmiş, bu uygulama üzerinden sürecin işleyişi kısaca örneklendirilmiştir. Bu bölümde geleneksel yöntem ile model tabanlı çevik süreç kıyaslanmıştır.

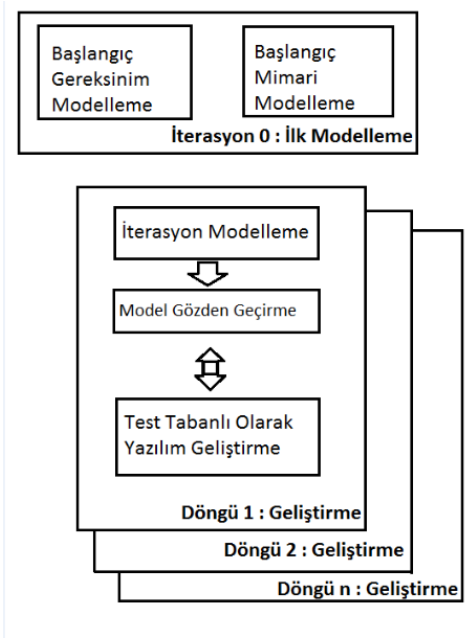
2. Model Tabanlı Yaklaşım

Model, bir sisteme ait işlev, yapı ve davranışların biçimsel belirtimidir. Modeller, karmaşık sistemleri soyutlayarak daha esnek bir biçimde gösterilmesini sağlayan yapılardır ve temeli UML'e dayanır [2][3]. Model tabanlı geliştirme yönteminde, yazılım kodları üretilmeden önce modellemeler yapılır [4]. Bu yaklaşımda herşey model olarak tanımlanır. Nesneye yönelik yaklaşımdaki herşeyi nesne olarak tanımlama durumu gibi, model tabanlı yaklaşımda da herşey modeldir yaklaşımı vardır [5]. Amaç, yazılım ihtiyaçlarının belirlenerek doğru çözümlerin üretilebilmesini sağlamak ve ortak bir dilde yapıyı anlatabilmektir.

3. Model Tabanlı Çevik Süreç

Model tabanlı çevik süreç iterasyonların ilerlemesi şeklinde olur [6]. Bu iterasyonlar sırasında modellemeye öncelik verilir, modelleme yapılmadan kod yazımına geçilmez. Model tabanlı çevik süreç, iterasyon 0 (döngü 0) olarak tanımlanan başlangıç modelleme aşaması ile başlar [7]. Bu aşama başlangıca özgüdür, sonraki iterasyonlarda tekrarlanmak zorunda değildir; ancak sonraki iterasyonlarda bu aşamada

üretilen modeller güncellenebilir. Başlangıca özgü ilk iterasyondan sonra, yazılımın her modülü için tekrar tekrar uygulanacak üç aşamalı iterasyonlara geçilir. Bu iterasyonlardaki aşamalar iterasyon modellemesi, model gözden geçirmesi ve test tabanlı geliştirme aşamalarıdır. Model tabanlı çevik süreçte uygulama geliştirilmesi test tabanlı (test yönelimli geliştirme) olarak yapılır. Test tabanlı geliştirmede yazılımcı test birimleri oluşturulmadan kaynak kodların yazımını reddeder. Her modül için o module özgü test modülleri oluşturulur. Ayrıca her iterasyon için genel görüş/değerlendirme aşaması da bulunmaktadır. Bu aşama, her aşama için görüş bildirimi ve notların alındığı isteğe bağlı olarak uygulanan bir aşamadır.



Şekil 3.1 : Model Tabanlı Çevik Süreç [8]

3.1. Başlangıç Modellemesi

Başlangıç modellemesi, iterasyon 0 olarak belirlenen yazılım başlangıcında uygulanan aşamadır. Bu aşama ayrıca sonraki iterasyonlar uygulanırken geri dönülüp üzerinde değişiklik yapılan aşamadır. Uygulama geliştirilirken bu kısıma dönülüp analiz ve tasarımlar güncellenir.

Bu aşamada,

- Yazılımın amaç ve kapsamı belirlenir.
- Yazılımın gereksinimleri oluşturulur.
- Yazılım modüllere ayrılır, önem ve öncelik sırasına göre bu modüller kuyruğa yerleştirilir.
- Kullanım diyagramları, sınıf diyagramları gibi modellemeler ile yazılım mimari tasarımı yapılır.

Modelleme ile gereksinim analizi birbiri ile etkileşimli olarak esnek bir yapıda olmalıdır. Gereksinimler değiştikçe modellemeler üzerinde güncelleme yapılabilmelidir.

İterasyonlar oluşturulurken her modül, bir veya birkaç gereksinimi karşılayacak ya da bir gereksinimin önemli bir aşamasını tamamlayabilecek şekilde oluşturulmalıdır.

Modellemeler oluşturulurken, çevik metoduna uygun şekilde çevik modellemeler ile sadece ihtiyaç görecek düzeyde modelleme yapılmalıdır. Çevik modelleme, çevik prensiblerine uygun modelleme olarak tanımlanır. Modellemeler basit ve anlaşılır olmalıdır. Amaç, çok iyi modelleme yapmak değil, modellemelerden de faydalanarak hızlı bir şekilde yazılım prototiplerini çıkartmaktır.

Başlangıç modelleme süreci, yazılım büyüklüğüne göre kurumsal yazılımlarda birkaç gün sürer.

3.2. İterasyon Aşamaları

Her modül için tekrarlı olarak uygulanacak aşamadır. Bu aşamada iterasyon kuyruğunun tepesinden bir modül seçilir. Gereksinimler tamamlanıncaya kadar iterasyon 1, 2, 3… şeklinde devam ettirilir. Bu iterasyonlar birbirleri ile çift yönlü etkileşimli olarak iterasyon modellemesi, model gözden geçirme ve test tabanlı uygulama geliştirme süreci olmak üzere üç adımdan oluşur.

İterasyon Modellemesi, seçilen iterasyonun içeriği, kapsamı incelenip, iterasyon planının çıkarıldığı aşamadır. Doğru tahmin ve planlar yapılabilmesi için seçilen iterasyona özgü çevik modellemeler yapılır.

Model Gözden Geçirme, iterasyonun gerçeklenmesi sırasında çıkan problemleri tartışıldığı aşamadır. Duruma göre model üzerinde değişiklik yapılıp ekibin bilgilendirildiği bir aşamadır.

Test yönelimli yazılım geliştirme yaklaşımı, kaynak kodların yazılmadan önce, test birimlerinin oluşturulmasını öngeren bir yaklaşımdır.[9] Test yönelimli yazılım geliştiren programcı, test birimleri olmayan programı yazmayı reddeder. Model tabanlı çevik süreçte de her iterasyon için test yönelimli yazılım geliştirilmesi öngörülür. Her iterasyon için öncelikle o iterasyonun birim testleri hazırlanır.

Test yönelimli yazılım geliştirilmesi küçük adımlar şeklinde olur. Bir gereksinim için birim test yazılır, sonra bu testi geçecek kod yazılır, sonrasında da bu kod üzerinde iyileştirmeler yapılır.

Şekil 3.2 : Test Yönelimli Programlama Yaşam Döngüsü

Kod üzerinde yapılan iyileştirmeler, yapının davranışsal özelliklerini değiştirmeden yapılan ufak değişikliklerdir.

Veritabanı tabloları üzerinde de normalizasyon işlemleri yapılabilir.

4. Uygulama

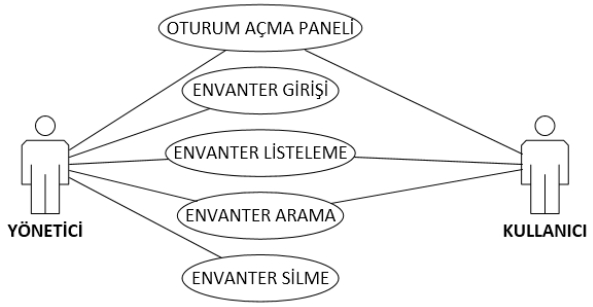
Bu çalışmamızda, bir masaüstü uygulaması olarak ürün ekleme, silme, güncelleme işlemlerinin yapıldığı "Stok Takip Sistemi" uygulaması, model tabanlı çevik yöntem ile geliştirilmiş; süreç aşamalarının nasıl ilerleyeceği gösterilmiştir.

Başlangıç modellemesi aşamasında sistemin gereksinimleri kabataslak olarak çıkartılır. Sistemimiz bir firma için envanterindeki ürünlerinin kaydını tutabileceği bir sistemdir. Bu sistem için gereksinimler Çizelge 4.1' de gösterilmiştir.

Çizelge 4.1: Yazılım gereksinimleri

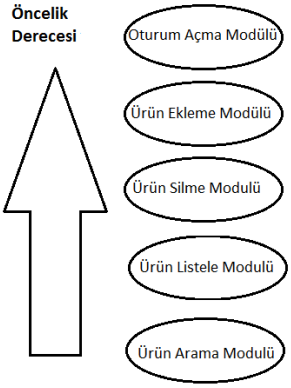
Sisteme sadece yetkili kişiler erişebilmeli
Sistemin bir oturum açma paneli olmalı
Kullanıcılar ve yöneticiler olmak üzere yetkilendirme 2 türe ayrılmalı
Her kullanıcı ve yöneticinin kendisine özel bir kullanıcı adı ve şifresi olmalı
Yöneticiler stoktaki ürünlerin listesini görebilmeli
Yöneticiler stoktaki ürünler içinden arama yapabilmeli
Yöneticiler stok içine ürün ekleyebilmeli
Yöneticiler stok içindeki ürünleri silebilmeli
Kullanıcılar ürün listesini görebilmeli
Kullanıcılar ürün listesi içinde arama yapabilmeli

Gereksinimler çevik sürecin bir gereği olarak esnek olmalı ve süreç içerisinde müşteri ile etkileşimli olarak düzenlenebilmelidir. Başlangıç modellemesinde yapının mimari tasarımı da modellenebilir. Kullanım diyagramları bu çalışmamızda mimari tasarım için kullanılmıştır.



Şekil 4.1: Yazılım kullanım diyagramı

Başlangıç modellemesi aşamasında amaç gereksinimlerin anlaşır düzeye gelmesidir. Gereksinimler belirlendikten sonra bu gereksinimler iterasyon modülleri ile eşleştirilmeli. Modüller önem sırasına göre iterasyon kuyruğuna yerleştirilir.



Şekil 4.2 : İterasyon Kuyruğu

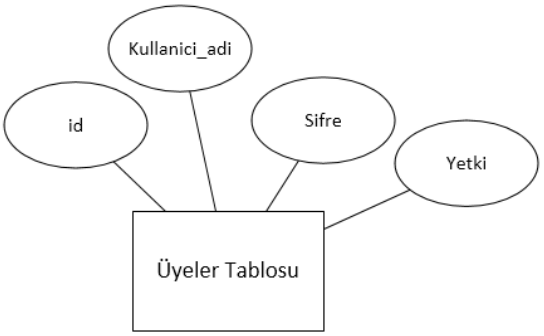
4.1. İterasyon 1 : Oturum Açma Modulü

İterasyon kuyruğunun en üstündeki modül ile çalışmaya başlanır. Modül için module özgü iterasyon gereksinim analizi ilk analizden farklı olarak çıkartılır.

Çizelge 4.2 : İterasyon gereksinim analizi

Kullanıcı adı ve şifre girişi yapılmalı
Buton ile tetiklenerek veritabanındaki üyeler girilen bilgiler eşleştirilmeli
Eşleştirilme sonucu sistem mesajı ile gösterilmeli
Bilgiler eşleştiğinde yetkilendirme derecesi sorgulanmalı
İki tür yetkilendirme derecesi var. Eğer yönetici ise yönetici paneli açılmalı, eğer kişi kullanıcı yetki derecesinde ise kullanıcı paneli açılmalı

Gereksinimler belirlendikten sonra gerçekleştirilecek modül eğer bir arayüz tasarımı gerektiriyor ise iterasyon arayüz tasarımı basit bir şekilde belirlenir. Modül için veritabanı tabloları da modellenir.



Şekil 4.3 : İterasyon E-R Diyagramı

Modellemeler tamamlandıktan sonra uygulama geliştirme aşamasına geçilir. Burada test tabanlı bir geliştirmenin gereği olarak öncelikle test birimleri üretilir. Uygulamamızda Black Box (Kara kutu) test tekniği kullanılmıştır. Kara kutu tekniğinde uygulamanın düzgün ve doğru bir şekilde çalıştığı kontrol edilir, iç yapısına bakılmaz.

Çizelge 4.3 : İterasyon test birimleri

Kullanıcı adı ve şifre girişi yapıldığında veritabanı ile doğru bir şekilde eşleşiyor mu? *Kullanıcı ve yönetici yetkilendirmesi ayırt edilip farklı paneller açılıyor mu?* *Hatalı giriş yapıldığında uyarı veriliyor mu?* *Doğru giriş yapıldığında sistem bilgilendirme mesajı veriyor mu?*

Modul, bu test birimlerine ve modellemelere göre geliştirilir. Bu çalışma da başlangıç modellemesi ve ilk iterasyon aşamasının, model tabanlı çevik süreç ile ilerleyişi en basit haliyle gösterilmiştir. İterasyon kuyruğundaki sonraki modüller de aynı adımlar ile ilerler.

Geleneksel süreç ile geliştirdiğimizde uzun bir analiz ve tasarım aşamasından geçiriliyor. Gereksinimler değişime çok kapalı kalıyor. Ancak model tabanlı çevik süreçte gereksinimler ve modeller esnek bir yapıya sahiptir ve değişim yapılmasını kolaylaştırmıştır. Geleneksel süreçte uygulama kodları geliştirildikten sonra test sürecine tutulmaktadır. Model tabanlı çevik süreçte ise test yönelimli uygulama geliştirme ile küçük birimler halinde testler oluşturulup, test birimleri oluşturulduktan sonra uygulama kodları oluşturulmaktadır. Bu süreç uygulamayı küçük parçalar halinde geliştirerek, geleneksel yapıya göre çok esnek bir yapı oluşturmaktadır. Süreç, yazılımı iterasyonlara, iterasyonları da test birimlerine ayırarak geliştirmektedir.

5. Sonuçlar

Model tabanlı yazılım geliştirme, dokümantasyonu arttırsa da, yazılım geliştirmeyi yavaşlatmaktadır. Bu sebeple model tabanlı geliştirmeyi çevik süreç üzerinde uygulayarak, çevik sürecin dokümantasyon eksikliği gibi dezavantajları kapatılırken; çevik sürecin hızlı uygulama geliştirme yöntemlerinden faydalanarak model tabanlı yöntemin hız dezavantajını kapatılabilmektedir. Çevik süreç ile model tabanlı yazılım geliştirme, birbirlerini uyumlu bir şekilde tamamlayabilmiştir.

6. Kaynaklar

[1] Matinnejad R. , "Agile Model Driven Development: An Intelligent Compromise" , Software Engineering Research, Management and Applications (SERA), 2011 9th International Conference on, Pages:197-202

[2] Akgül Ö. "Model Güdümlü Yazılım Geliştirme", *İstanbul Üniversitesi Online Bilişim Dergi*, Temmuz 2010

[3] Sezen E. "Business Process Automation with Model Driven Development, Yüksek Lisans Tezi, Dokuz Eylül Üniversitesi, İzmir, 2010

[4] Öz C. "Model Güdümlü Yazılım Geliştirme İle Gömülü Kaynakların Yönetimi, Ege Üniversitesi" , Yüksek Lisans Tezi, Ege Üniversitesi, İzmir, 2011

[5] Sıkıcı A. Topaloğlu Y. "Yazılım Geliştirmede Model Sınıflandırma", *3.Ulusal Yazılım Mühendisliği Sempozyumu (UYMS'07) Bildiri Kitabı*, Eylül 2007

[6] Zhang Y., Patel S. "Agile Model-Driven Development in Practise", *IEEE Software* , 2011

[7] Ambler, S.W. "Agile Model Driven Development" , *XOOTIC Magazine*, 2007

[8] Ambler, S.W. *The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.0 New York: Cambridge University Press*, 2004

[9] Ateş Ü. "Test Güdümlü Programlama" , TMMOB Bilgisayar Mühendisleri Odası BM Dergi, 2013