Dokuz Eylül Üniversitesi-Mühendislik Fakültesi
Fen ve Mühendislik Dergisi
Cilt 19 Sayı 55.1 (Özel Sayı) Ocak 2017

Dokuz Eylul University-Faculty of Engineering
Journal of Science and Engineering
Volume 19 Issue 55.1. (Special Issue) January 2017

DOI: 10.21205/deufmd.2017195532

# Son Kullanıcı Geliştirme için Otomatik Kod Üretim Aracının Tasarımı ve Gerçeklenmesi

**Akhan Akbulut*, Fatma PATLAR AKBULUT, Hakan KÖSEOKUR, Çağatay ÇATAL**

İstanbul Kültür Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, 34156, İstanbul

(Alınış / Received: 08.07.2016, Kabul / Accepted: 12.08.2016,
Online Yayınlanma / Published Online: 09.01.2017)

**Anahtar Kelimeler**
Son kullanıcı geliştirme,
Son kullanıcı yazılım mühendisliği,
Otomatik kod üretimi,
Rezervasyon sistemi,
Yazılım aracı,
Görsel programlama

**Özet:** Son kullanıcı geliştirme yaklaşımları; yazılım mühendisi olmayan kullanıcıların, yazılım çıktılarını kendilerinin oluşturabileceği, değiştirebileceği ve uyarlayabileceği teknolojiler ve yöntemlere odaklanmaktadır. Bu amaçla; birleştirme teknolojileri, örnek ile programlama, görsel programlama, model tabanlı yaklaşımlar, servis yönelimli mimariler ve otomatik kod üretimi kullanılabilmektedir. Bu çalışmada, görsel programlama ile birlikte otomatik kod üretimi tercih edilmiş ve bir son kullanıcı geliştirme aracı gerçeklenmiştir. Seçilen uygulama alanı olan rezervasyon sistemleri için otomatik kod üretimi sağlanmıştır. Bu sürecin temel faydası; geliştirme zamanın kısaltılması, son kullanıcıların geliştirmesini yapabilmesi ve sistem tasarımı ile üretilen uygulama arasındaki farklılıkların en aza indirgenerek, tutarlılığın sağlanmasıdır. Farklı rezervasyon sistemleri için kullanılması amaçlanan bu sistem, kullanıcıların iş modellerini görsel ara yüzler ile tanımlamalarına ve bu tasarımdan web tabanlı uygulamanın çalışması için gerekli tüm dosyaların üretilmesine imkan tanımaktadır. Son kullanıcılarda web teknolojilerine ait herhangi bir geliştirme tecrübesi aranmayıp, uygulamanın çalışması için gerekli ara yüzler, stil ve tasarım dosyaları, veritabanının oluşturulması otomatik kod üretimi ile gerçekleştirilmiştir. Bu çalışmada; son kullanıcı geliştirme için görsel programlama ve otomatik kod üretimi tekniklerinin, alana özel uygulanması gerektiği, jenerik yaklaşımların etkin olmayacağı sonucuna varılmıştır.

# Desing and Implementation of an Automatic Code Generation Tool for End-User Development

**Keywords**
End-user development,
End user software engineering,
Automatic code

**Abstract:** End user development (EUD) approaches focus on technologies and methods, which can help end users who are not software engineers, to create, change, and adapt their own software artifacts. For this purpose, composition technologies, instance-based programming, visual programming, model based approaches, service oriented architectures, and automatic code

generation, Reservation system, Software tool, Visual programming

generation techniques can be used. In this study, visual programming in conjunction with automatic code generation was opted and an end-user development tool was implemented. As the case study, automatic code generation was performed for reservation systems. The main aims of this process are to shorten the development time, let the end users develop their own tools, and create the consistency between system design and generated application. This system, which is aimed to be used in different reservation applications, lets users define their business models with graphical objects, and generate all the necessary files from this design for a web-based application. It is not required by the end-users to know web technologies and in addition to the database, interfaces, style and design files are generated automatically. In this study, it is shown that visual programming in conjunction with automatic code generation must be applied for the specific domain instead of a generic purpose implementation and this is a more efficient approach from end-user development perspective.

*Sorumlu yazar: a.akbulut@iku.edu.tr

## 1. Giriş

Son kullanıcı geliştirme (end-user development) veya son kullanıcı yazılım mühendisliği kavramları [1], programlama veya yazılım mühendisliği eğitimine sahip olmayan, sadece günlük iş yükünü kolaylaştırmak amacıyla programlar oluşturan son kullanıcıların, yazılım geliştirmesini hedefleyen ve bu amaçla yöntemler, teknolojiler ve araçlar ortaya koyan yaklaşımlardır. Profesyonel yazılımcı sayılarıyla kıyaslandığında, ofis işlerini sürdürebilmek üzere tablolama ve veri tabanlarıyla çalışan, Matlab gibi ortamlarda simülasyonlar gerçekleştiren, farklı teknolojilerle web siteleri tasarımı hazırlayan çok daha fazla sayıda son kullanıcı mevcuttur.

Son kullanıcı yazılım mühendisliği kavramı, sadece yazılımın oluşturulmasını sağlayan programlama aktivitesine odaklanmakla kalmayıp, diğer aktiviteler olan gereksinim analizi, tasarım, test ve bakım faaliyetleriyle de ilgilenmektedir [2,3]. Bazı kaynaklarda bu ayrım sunulmamış olsa da, son kullanıcı programlama ve son kullanıcı yazılım mühendisliği arasında bu tür bir farktan bahsedilebilmektedir. Son yıllarda bu konuda araştırma yapan araştırmacıların sayısı artmış, sadece bu konuya odaklanan konferanslar düzenlenmeye başlamıştır. Belirli bir meslek grubuyla sınırlı olmayan bu son kullanıcılar, farklı eğitim altyapılarından gelmekte, bu nedenle ilgili alanın terminolojisine uygun olarak gerekli yazılım platformları tasarlanmalıdır. Jenerik yaklaşımlar yerine, doğrudan o alana özel yaklaşımların kullanılmasının üretilecek çıktı kalitesi açısından daha olumlu etkisi olacağı değerlendirilmektedir.

Son kullanıcı geliştirme veya programlama amacıyla; birleştirme teknolojileri (composition technologies) uzun zamandan beri uygulanmıştır. Örneğin; mash-up araçları [4] bu kapsamda bilinen teknolojilerden olup farklı veri kaynaklarından yararlanarak uygulama geliştirilmesi sağlanabilmektedir. Kurumsal mash-up platformlarından, daha genel amaçlı olanlarına ve bilimsel iş akışı ortamlarına kadar çok çeşitli gruplamalar söz konusudur. Son kullanıcı programlama amacıyla kullanılabilecek diğer yaklaşımlar ise model tabanlı yöntemler,

örnek ile programlama (instance-based programming), servis yönelimli mimari yaklaşımlar, görsel programlama ve otomatik kod üretimi teknikleridir.

Bu çalışmanın temel motivasyonu, çok sayıda teknik ve teknolojinin mevcut olduğu bu problem için, etkin bir yaklaşımın belirlenmesi ve örnek bir durum çalışmasında seçilen bu yaklaşımın test edilerek gerekli değerlendirmelerin yapılmasıdır. Bu amaçla, örnek durum çalışması olarak rezervasyon sistemlerinin ele alınmasına karar verilmiştir. Rezervasyon sistemleri denildiğinde, havayolu firmalarında koltuk rezervasyonu veya otellerde oda rezervasyonu gibi profesyonel uygulamalar akla gelebilmektedir. Oysaki rezervasyon problemi, günlük hayatımızda çok sayıda alanda karşımıza çıkabilen temel bir problemdir. Örneğin, kuaföre giden bir müşterinin, diğer müşterileri beklemesi ve vakit kaybetmesi yerine, son kullanıcı programlama yazılımıyla oluşturulan bir yazılım üzerinden rezervasyon sağlaması oldukça etkin bir yaklaşım olacaktır. Bu çalışmada, bu tür bir rezervasyon sisteminin son kullanıcı bakış açısıyla geliştirilmesi durum çalışması olarak ele alınmıştır. Farklı alternatifler değerlendirildikten sonra genel yaklaşım olarak görsel programlama ve model tabanlı geliştirme yaklaşımlarının uygulanmasına karar verilmiştir.

Uygulamaların kullanıcıların isteklerine göre kişiselleştirilebilmesi ve kodlama becerisi aranmaksızın kurulumlarının tamamlanabilmesi, günümüzde yaygın olarak sunulan bir özelliktir. Basit ara yüzler kullanarak süratli bir şekilde kişisel web siteleri (Wordpress, Drupal, Joomla, Weebly, Jekyll, Octopress, Hugo vb.), e-ticaret sistemleri (Magento, osCommerce, nopCommerce vb.), emlak siteleri (flexmls, LoopNet, kunversion vb.) veya sağlık sistemleri (CareTech CareWorks vb.) kullanıma açılabilmektedir. BuiltWith Trends ve W3Tech 2016 yılı istatistiklerine göre 75 milyon web sitesi doğrudan Wordpress kullanmaktadır. Bu sayı ülke nüfusumuza yakın bir rakam olup, dünya üzerindeki tüm sitelerin %18.9'una karşılık gelmektedir. Bu örneklere de baktığımızda, son kullanıcı geliştirme yaklaşımlarının, esasında birçok altyapı üreten firma tarafından hali hazırda kullanılmakta olduğu, bu sayede milyonlarca son kullanıcıya erişebildiği görülmektedir.

Ancak işletmelerin çalışma modellerine göre uygulamaları çalışacak şekilde yapılandırabilmeleri, RAfta HAzır Ticari Araçlar (RAHAT) (commercial of the shelf-COTS) tarafından çoğu zaman sunulamayan bir özelliktir. Çoğu müşteri, bu tür hazır ticari araçlar kullanmaktan kaçınmalarını, satılan yazılımların ihtiyaçlarına uygun olmamasına veya kendi çalışma modellerini uygulamasına müsaade etmemesine bağlamaktadır. Bu bağlamda müşterilerin kendi çalışma modellerini, kodlama becerisine ihtiyaç olmaksızın görsel ara yüzler vasıtasıyla tasarlamasına imkan veren yazılımlar, bu tür paket programlar diyebileceğimiz programlara göre tercih sebebi olmaktadır.

Büyük kitlelere ulaşmanın günden güne önem kazandığı günümüzde, tüm dünya genelinde etkin bir dağıtım kanalına sahip olmak, hiç şüphesiz çok önemli bir güçtür. Teknolojinin bu denli önem kazandığı çağımızda, internet kitlelere ulaşmak için en doğru araç haline gelmiştir ve bu gücü arkasına alan işletmelerin üretim hacimleri hızla büyümektedir. Bu kapsamda kullanıcı ihtiyaçlarını en efektif şekilde karşılamak üzere tasarlanmış rezervasyon sistemleri de bu dağıtım kanalının yapı taşlarından biridir. Çevrim içi rezervasyon sistemlerinin işletmelere ekonomik katkısı göz önüne alındığında sektördeki stratejik önemi daha da ortaya çıkmaktadır. Fakat sistem kurulumları ve işletme maliyetlerinin az olabilmesi, işletmelere özel uygulamaların

kişiselleştirilebilmesi çoğu ticari yazılım tarafından kullanıcılara sunulamamaktadır. Bu çalışmada; işletmelerin kendi çalışma modellerini uygulamaya adapte edebildikleri bir otomatik kod üreten sistemin tasarımı açıklanmaktadır. Görsel programlama ve model tabanlı geliştirme sayesinde, otomatik kod üretimi sağlanabilmiş, son kullanıcıların rahatlıkla bu sistemi kullanabilmesi hedeflenmiştir. Jenerik bir dil yerine, doğrudan ilgili alan özel bir yazılım altyapısının tasarlanması sistemin son kullanıcılar tarafından benimsenmesini arttırmıştır.

Son kullanıcı geliştirme ve otomatik kod üretimine ilişkin yapılan çalışmalar ikinci bölümde anlatılmaktadır. Üçüncü bölüm önerilen rezervasyon sistemleri için geliştirilen aracı detaylandırılmakta ve son bölüm sonuç ve gelecek çalışmaları sunmaktadır.

## 2. İlişkili Çalışmalar

Son kullanıcı geliştirme veya son kullanıcı yazılım mühendisliği yaklaşımları, çok sayıda yazılım mühendisliği araştırmacısının ilgisini çekmiş ve halen çekmeye devam etmektedir. Doğrudan bu konuda konferanslar düzenlenmekte ve bazı yazılım firmaları bu konuda AR-GE projeleri üretmektedir. Birçok avantaj sunmasıyla birlikte, bazı araştırmacılar gerek güvenlik hususları gerekse de test konularında endişelerini ortaya koymaktadır. Yazılım test mühendisliği konusunda gerekli bilgi seviyesine sahip olmayan bir son kullanıcının geliştireceği yazılımın ne oranda test edileceği gibi bazı sorular gündeme gelmekte ve yeterli test edilmemesi durumunda kurumların karşılaşabileceği diğer problemler bir endişe kaynağı olarak raporlanmaktadır. Profesyonel yazılım uzmanlarının bile geliştirdiği yazılımlarda güvenlik açıklıkları ortaya çıkarken, son kullanıcıların geliştireceği yazılımlarda bu tür sorunların hiç oluşmayacağını söylemek gerçekçi olmayacaktır. Bu nedenlerle; ilgili alanının kritiklik durumu dikkate alınarak, gerekli altyapının tasarlanması ve gerekli test durumlarının ilgili yazılıma tümleştirilmesi önem arz etmektedir.

Son kullanıcı geliştirme için farklı teknolojiler ve araçların varlığı önceki bölümde açıklanmıştı. Özellikle birleştirme teknolojileri uzun yıllardan beri uygulanmasına rağmen, yeterli başarıyı sağlayamadığı için eleştirilmektedir. Bu çalışmada hedeflenen bir görsel programlama altyapısının tasarlanması ve otomatik kod üretiminin gerçekleştirilmesidir. Görsel programlama ve otomatik kod üretiminin, alana özel geliştirilmesinin son kullanıcı geliştirme kapsamında etkinliği bu çalışmada değerlendirilmektedir. Otomatik kod üretimi denildiğinde, ilk bakışta standart bir yöntem olduğu düşünülebilir ancak farklı otomatik kod üretimi teknikleri incelendiğinde, konunun detayı daha rahat görülebilmektedir. Bu bölümde otomatik kod üreticileri açıklanmaktadır.

Otomatik kod üreticileri, üst seviye kodu otomatik olarak üreten programlar olarak tanımlanmaktadır. Otomatik kod üretimi, geleneksel kodlama yaklaşımına göre farklı avantajlar sunmaktadır. Daha az hatadan oluşan, yüksek kaliteli kodların esnek bir mekanizmaya göre oluşturulması sayesinde, geliştirici üretkenliği beraberinde artmakta ve yazılım projeleri daha kısa sürelerde tamamlanabilmektedir.

Kod üretici araçlar, aktif ve pasif olmak üzere iki kategoriye ayrılmaktadır. Pasif modelde kod üretici araç bazı kaynak kodlar üretmekte ve daha sonra, yazılım geliştirici bu üretilen kodları yeniden düzenleyerek son haline getirmektedir. Pasif kod üretici aracın, kaynak kod üzerinde kısa veya uzun dönemde bir sorumluluğu bulunmamaktadır. Tümleşik geliştirme ortamlarındaki sihirbaz destekleri, pasif

kod üretici araçlara örnek olarak verilebilir. Aktif kod üretici araçlar ise kod üzerinde sorumluluğa sahip olduğundan, aynı çıktı üzerinden birkaç defa çalıştırılabilmektedir. Kod üzerinde değişiklik gerekli olduğunda, yazılım geliştiriciler kod üretici araca gerekli parametreleri sağlayarak, kodun yeniden üretimini gerçekleştirebilmektedir. Aktif kod üreticiler 6 model altında ele alınabilmektedir [5]:

**Kod Şekillendirme (Code Munging)**: Munging İngilizce argoda bir kelime olup, bir şeyi bir biçimden diğer bir biçime şekillendirme anlamında kullanılmaktadır. Kod şekillendirici (code munger), kaynak kodu okuyup analiz eder, bir ya da daha fazla çıktı dosyasını şablonları kullanarak oluşturur. Örneğin, Java dilinde yazılmış bir kodu C# diline dönüştüren araç, kod şekillendirici modelinin kullanıldığı bir araç olarak değerlendirilebilir. Javadoc, ANTLR gibi yazılımlar bu kategoriye girmektedir.

**Satır içi kod açıcı (inline-code expander)**: Kaynak kod içerisine, çeşitli sentaks eklentileriyle farklı dillerin eklenebilmesi olarak açıklanabilir. Çoğu durumda, kaynak kod içerisine SQL ifadeleri eklemek amacıyla kullanılan bir yaklaşımdır. Özel gösterim şekliyle SQL kodu, kaynak kod dosyasında işaretlendikten sonra, kod üretici bu özel kodları SQL sorgusuna veya komutuna dönüştürebilmektedir. Bu sayede geliştirme kodu, sorguyu yönetecek olan altyapıdan bağımsız olarak geliştirilebilmektedir. Pro*C, SQLJ gibi yazılımlar bu kategoriye girmektedir.

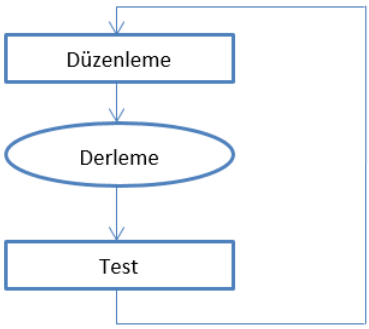
**Karma kod üretimi (mixed-code generation)**: Bu tür bir araç, bir kaynak kod dosyasını okur, değiştirir ve ilk dosyayla bu yeni dosyayı yer değiştirir. Bu işlemin ardından, örneğin o dile özgü bir derleyici çalıştırılarak, yürütülebilir bir kod üretilebilir. Satır içi kod açıcı modelden daha farklı bir yaklaşımdır, bunun sebebi ise üreticinin çıktısının girdi dosyasına geri konulmasıdır. Örneğin, C++ kodları içerisine birim testlerine dönük eklentiler yapmak amacıyla bu modelden yararlanılabilir.

**Kısmi-sınıf üretimi (partial-class generation)**: Kod üretici; bir tanım dosyasını okur ve farklı şablonları kullanarak bir veya daha fazla temel sınıf oluşturur. Daha sonra bu temel sınıflar, türetilmiş sınıflarla birlikte kullanılır. Nihai ürünü elde etmek üzere tüm sınıflar bir arada derlenir. Özellikle, veri tabanı erişim sınıflarının üretiminde kullanılabilmektedir.

**Katman üretimi (tier/layer generation)**: Bir uygulama içerisindeki tüm bir katmanın oluşturulması ve idame ettirilmesi sağlanır. Model güdümlü üretim bu modele örnek olarak verilebilir. Model güdümlü üretimde; bir XML dosyasından UML bilgileri elde edilerek, otomatik kod üretimi elde edilmektedir. Kısmi-sınıf üretimi ile katman üretimi arasındaki en büyük fark, ilgili katman için tüm kodun, katman üretimi aracında oluşturulmasıdır. Katman üreticisi oluşturmadaki zorluk, özel durumların tasarlanmasıdır. Kısmi-sınıf üretiminde ise çoğu durumda bu işlemler daha basittir ve zamanla katman üretimi modeline geçiş sağlanabilir.

**Tam-alan dili (full-domain language)**: İlgili alanda daha kolaylıkla kavramları temsil etmek üzere uyarlanmış bir Turing tam dilidir. Araç kullanıcıya yeni bir dil sunar ve bu dilin kendine özgü tipleri, sentaksı ve işlemleri mevcuttur. Matlab bu konuda örnek verilebilir.

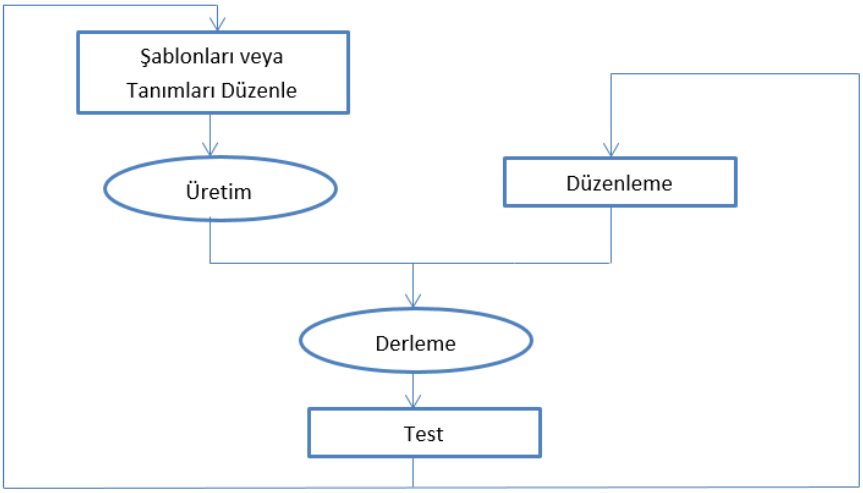
Yazılım geliştirmede temel iş akışı kodun düzenlenmesi, derlenmesi ve test edilmesi şeklindedir. Şekil 1'de bu durum temsil edilmektedir. Kod üretimi durumunda birkaç tane iş akışı elemanı daha bu resme dahil edilmektedir. Şekil 2'de kod üreticinin kullanıldığı durumdaki iş akışı gösterilmektedir.




**Şekil 1.** Kodlama iş akışı [1]

Şekil 2'nin sol bölümü üretim iş akışını göstermektedir. İlk olarak şablonlar ve tanım dosyaları düzenlenir ve kod üretici çalıştırılarak çıktı dosyaları oluşturulur. Daha sonra çıktı dosyaları, özel kodlarla birlikte derlenir ve uygulama son olarak test edilir.

Kod üretimi konusu, farklı uygulama alanlarıyla birlikte oldukça popüler bir alan haline gelmiştir. Vavilina ve Gaigals (2015), sinyal işleme algoritmalarını simulasyon modunda test etmek üzere kullanılabilen LabVIEW ortamında, iyileştirilmiş kod üretimine odaklanmış ve bu kapsamda bir araç geliştirmişlerdir [6]. Li ve arkadaşları (2015), önceki çalışmalarında Siber Fiziksel Sistem uygulamalarının modellenmesi için alana özel bir dil geliştirmiş ve yeni çalışmalarında, bu modelleme dili için bir kod üretim çerçevesi önermişlerdir [7]. Çerçeve yazılım; modeli okuduktan sonra, uygulamanın tüm konfigürasyonunu üretebilmekte ve yürütülebilir uygulamayı inşa edebilmektedir.




**Şekil 2.** Kod üreticinin dahil olduğu durumda kodlama iş akışı [6]

Mehmood ve Jawawi (2013), sistematik bir eşleme çalışması gerçekleştirerek, modellerden ilgiye yönelik (aspect-oriented) kod üretimi konusunda yapılan çalışmaları incelemişlerdir [8]. Çok sayıda ilgiye yönelik modelleme ve kompozisyon mekanizmasının olduğu çalışmanın varlığı raporlanmış, model güdümlü kod üretimi alanında daha fazla araştırmaya ihtiyaç olduğu bildirilmiştir. Tolvanen ve arkadaşları (2015), medikal cihazlar için (örnek çalışmada kan ayırıcısı) meta modelleme ve ilişkili kod üreticilerini kullanarak yürütülebilir kodu üretmişlerdir. Ek olarak; kod üreticisi yardımıyla, modelde hata ayıklama desteği sağlamışlar, modelin bir durumu ve programın durumu arasında senkronizasyon ihtiyacına dönük özellik eklemişlerdir [9]. Solís-Martínez ve arkadaşları (2015), mobil platformlarda oyun geliştirme sürecini hızlandırmak

amacıyla, BPMN notasyonuna dayalı VGPM isminde bir notasyon önermekte ve bu sayede, oyun mantığının önemli aşamalarını hızlıca tanımlamak ve maliyetleri azaltmak mümkün olabilmektedir [10]. Kim ve arkadaşları (2013), farklı hedef platformlara farklı kaynak kod üretebilen platform-bağımlı bir kod üretim çerçevesi geliştirmişlerdir [11]. Hedef platformun donanım/yazılım mimarisi AADL (Architectural Analysis Description Language) modelleri kullanılarak ifade edilebilmektedir. AADL modelleri içerisindeki bileşenlerin gerçeklemesine ilişkin fonksiyonlar ise platform-bağımlı kod havuzlarında yer almaktadır. Bu iki eleman, kod üretim algoritması tarafından kullanılarak, ilgili platform için kod üretilebilmektedir. Klöckner ve arkadaşları (2011), grafik işlemci birimleri (GPU) için çalışma anında kod üretimi sağlayan betik dili tabanlı bir yöntem önermişlerdir [12]. Graichen ve D'Amato (2011), elektrik santrali optimizasyonu için simulasyon testleri geliştirmek üzere bir çerçeve yazılım geliştirmiştir ve bu yazılım sayesinde otomatik kod üretimi gerçekleştirilebilmektedir [13]. Wang ve arkadaşları (2012), emniyet kritik aviyonik yazılımlar için AADL (Architectural Analysis and Design Language) modellerinden ARINC653 uyumlu RT-Java kodu oluşturma teknolojisini ortaya koymuşlardır [14]. Modellere dayalı otomatik kod üretimi, aviyonik yazılımlar için önemli bir çalışma alanıdır. Mehmood ve Jawawi (2011), ilgiye yönelik kod üretim yaklaşımlarını bir inceleme çalışma kapsamında ele almış, çoğu yaklaşımın sınıf diyagramlarına odaklandığını raporlamış, davranış diyagramlarının da dikkate alınarak tam kod üretiminin mümkün olabileceğini ifade etmiştirler [15]. Morkoç ve arkadaşları (2014), kalıcı mıknatıslı senkron motorun (PMSM) kontrol sistemi için DSP tabanlı gömülü kod üretim yazılımı geliştirmişlerdir ve bu sistem sayesinde, prototipleme kolaylaşmış, geliştirme süreleri kısalmıştır [15]. Wang ve arkadaşları (2012), türetim kurallarına dayalı olarak kod üretiminin gerçekleştirilebileceğini ifade ederek bu yönde bir modelleme metodu önermiştir [16]. Lachgar ve Abdali (2015), alana özel bir dil geliştirerek iOS platformuna daha hızlı kod üretimi sunmayı hedeflemiştir [17]. Bernardi ve arkadaşları (2016), web uygulamalarının geliştirimi için model güdümlü mühendisliği önermiştir [18]. 4 metamodel ve deklaratif bir süreç modelleme dilinin entegrasyonu ile işlemler gerçekleştirilmektedir. Umuhoza ve arkadaşları (2015), çok platformlu mobil uygulamalar için otomatik kod üretim yaklaşımlarını incelemiş ve endüstriyel deneyim çalışması olarak sunmuşlardır [19]. Tek bir yöntemin tüm yönleriyle iyi olduğu bir durumun tespit edilemediği raporlanmıştır. Tüm bu çalışmalarda, otomatik kod üretiminin farklı seviyelerde ve teknolojilerle gerçekleştirildiği görülmektedir. Yapılan çalışmada ise görsel programlama ve otomatik kod üretiminin, son kullanıcı yazılım mühendisliği açısından etkin bir yaklaşım olduğu örnek durum çalışmasıyla ortaya konulmuştur.

## 3. Yöntem

Alana özel modelleme araçları ve dilleri (domain specific languages-DSL), sistem kalitesinin iyileştirilmesine, tasarım ve analizin etkin olmasına imkan verirken, otomatik kod üretimini de sağlamaktadır [20]. DSL kullanımıyla birlikte; UML tabanlı kod üreten sistemlere göre daha kapsamlı kodların üretilebilmesi mümkün olabilmektedir. Ancak küçük ve orta ölçekteki projeler için uygulama oluşturmadaki zorluklara ek olarak, bakımın maliyetli oluşu, UML tabanlı kod üreten yaklaşımların tercih edilmesini yaygınlaştırmaktadır.

Geliştirilen genel rezervasyon sistemi birçok sektördeki ihtiyaçları karşılayacak şekilde UML diyagramlarından kod üreten bir araç olarak tasarlanmıştır.

Çalışma konusu sistem, bazen bir diş hekiminin ofisi için uygulama oluştururken, yeri geldiğinde daha komplike bir otelin rezervasyon sistemi olarak yapılandırılabilmektedir. Önerilen sistemdeki temel amaç, eğitim seviyesi ve bilgiden bağımsız olarak, herhangi bir bireyin ihtiyacı olan rezervasyon sistemini kendi başına kurarak ihtiyaçları doğrultusunda yapılandırabilmesidir. Bu amaca bakıldığında, son kullanıcı geliştirme veya son kullanıcı programlama yaklaşımlarıyla çok büyük oranda benzerlik gösterdiği açıkça görülmektedir.

Durum çalışması ve gerçekleme aşamasında; geliştirme ortamı olarak Visual Studio 2015 ve MS SQL Server 2012 kullanılmıştır. Yazılım geliştirme metodolojisi olarak çevik yazılım geliştirme yaklaşımı saldırgan (scrum) yönetim modeli altında kontrol edilmiştir. Üçer haftalık altı adet evre (sprint) sonunda önerilen sistemin prototipi kullanılabilir seviyeye ulaşmıştır.

Sistemin geliştirilmesi ve testlerinde kullanılan 5 temel yapı bulunmaktadır: Modeller, Kurallar, Görsel Yardımcılar, Şablonlar ve İçerik. Bunlardan sırası ile ilk dördü sistemin geliştirilmesi sırasında kullanılacak olan kaynakları ifade ederken, içerik, testler sırasında sistemin girdisi olarak kullanılmaktadır.

- **KodModel** – Sistemin kod üretiminde kullandığı ana sınıftır.
- **ModelKural** – Otomatik kod üretimi sırasında izlenecek kurallar dizisinin kontrolünü sağlayacak sınıftır.
- **GörselYardımcı** – Otomatik rezervasyon sistemi üretimi görsel öğeler yardımı ile yapılacağından bu kullanıcıya kolaylık sağlayacak animasyon, resim ve renklerin kontrolünü sağlayan JavaScript ve css ile görsel öğelerin bulunduğu klasör.
- **Şablon** – Veri tabanı, Asp.Net sayfası, sınıf, JavaScript ve stil dosyalarının her birinin üretimi sırasında kullanılacak olan şablonlar topluğunu içerir.
- **İçerik** - İçerik bilgisi son kullanıcı tarafından sağlanması gereken ve sistemle ilgili metin, resim ve teknik bilgi içeren bir kaynaktır.

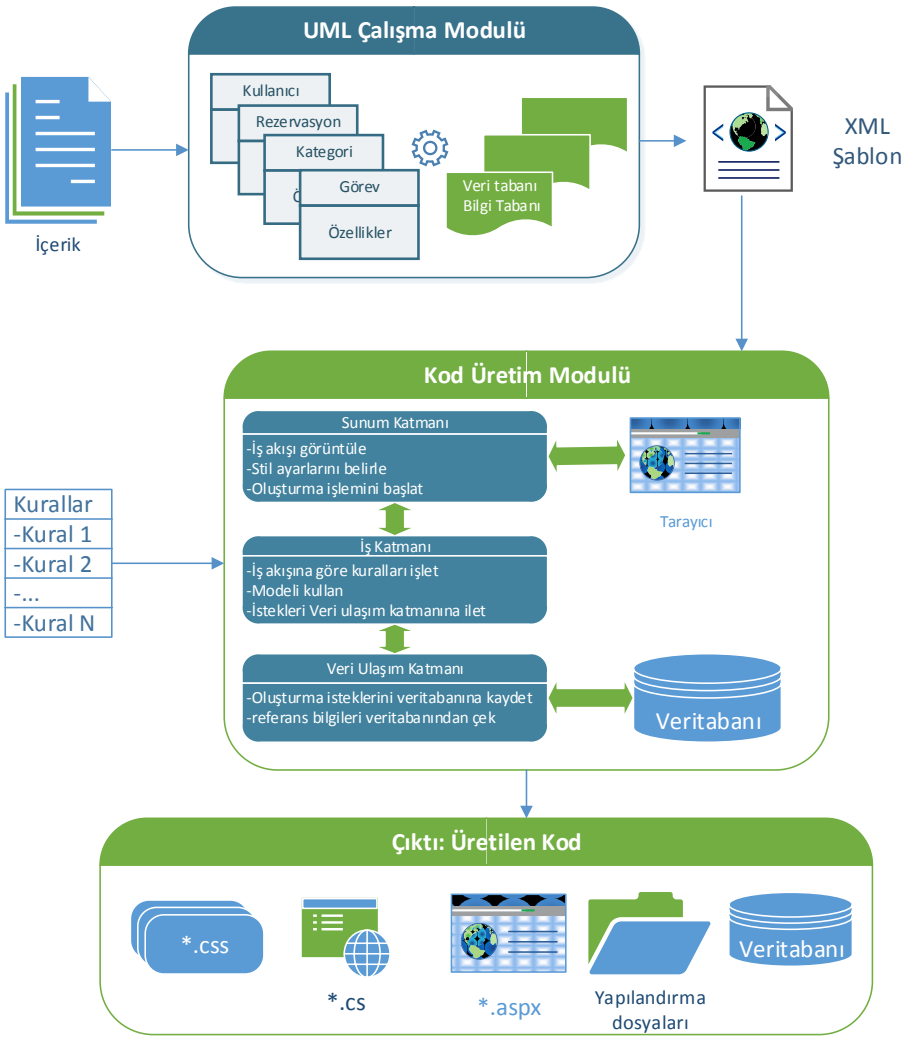
Otomatik kod üretim süreci Eşitlik 1'deki gibi formüle edilmiştir:

$$( M + Ş + K )D1 * ÇD1,D2 => MD2 \qquad (1)$$

M: Model
Ş: Şablon
K: Kural
Ç: Çeviri
D: Dil

Birinci dilden (D1) ikinci dile (D2) olan dönüşüm Model (M), Şablon (Ş) ve Kurallar (K) yardımı ile yapılmaktadır. Dil 1 geliştirme yapılan ortamı, dil 2 ise dönüştürülecek olan sistemin dilini temsil etmektedir. Gerekli girdileri sağlayan sistem çıktı olarak Rezervasyon sistem modelini üretmektedir.

Sistemin çalışma metodolojisi olan Şekil 3'e bakacak olursak, iki ana bileşeni bulunmaktadır. Bu bileşenler; UML'den çalışma modelinin oluşturulması ve bu modelden kod üretimi olarak ifade edilebilir.




**Şekil 3.** Rezervasyon Sistem Mimarisi

### 3.1. Uml Çalışma Modeli

Rezervasyon sistemi tasarımında ilk adım bir UML aracı kullanılarak kurulum senaryosuna konu olan projenin ihtiyaçlarına dönük tasarımın yapılmasıdır. Bu tasarım sonraki aşamalar için şablon olarak kullanılacak ve kod üretiminin referans alacağı yegane kaynak olacaktır. Şablon oluşturulması sırasında 4 ana varlık belirlenmiştir. Bunlar sırasıyla; Kullanıcı, Rezervasyon, Görev ve Kategoridir. Her bir varlığın içeriği oluşturulacak olan sistemin teknik alt yapısını karşılayacak şekilde belirlenmiştir. Çalışma modelinin detayları xml uzantılı bir dosya içerisinde saklanmaktadır. UML'in endüstri tarafından benimsenmiş olması [21] en önemli tercih sebebimizdir

### 3.2. Çalışma Modelinden Kod Üretimi

Çalışmanın ikinci adımında UML kullanılarak üretilen model aracılığı ile veri tabanı ve ara yüzler oluşturulmaktadır. Kod üretimini detaylıca ele aldığımızda, gerçekleşen işlemler aşağıda sunulmaktadır:

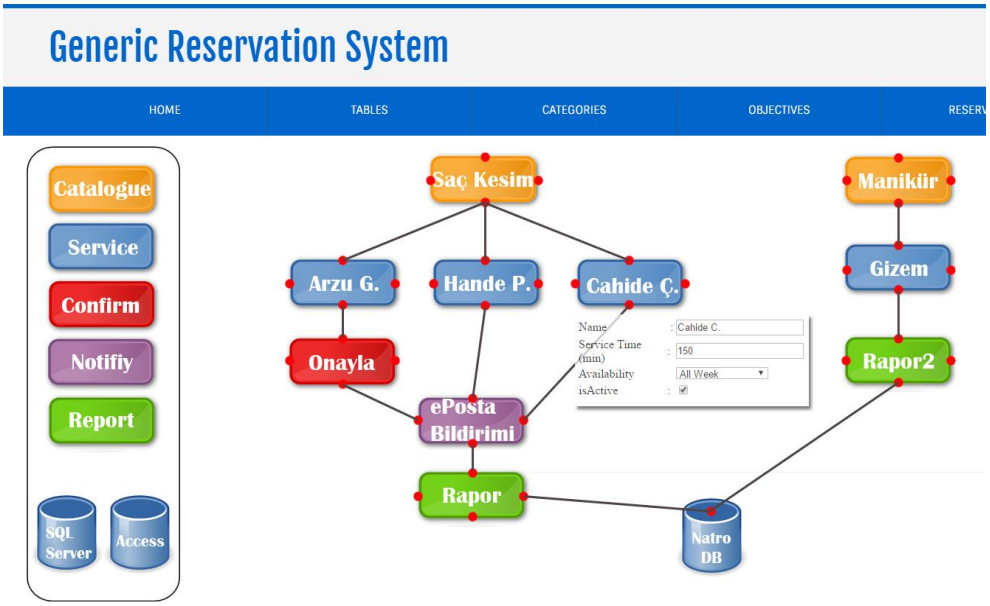
**Veri tabanı Kurulumu**: Sistemin veri tabanını oluşturmak için hangi sektörde hizmet vereceği, kaç ayrı hizmetinin bulunduğu, rezervasyon detayları ve çalışacak operatör sayısının belirlenmesi ile referans şablondan yararlanılarak tercih edilen veri tabanı platformunda otomatik olarak tablolarının oluşturulması işlemi gerçekleştirilir.

**Veri Ulaşım Katmanı Gömülmesi**: Özelleştirilmiş olarak oluşturulan veri tabanı detayları kullanılarak sistemin veri ulaşım katmanı oluşturulur. Bu katmanda yer alan bilgiler; bağlantı detayları (veri tabanı adı, yeri, vb.), varlıklar, varlıkların özellikleri, varlıkların işlevleridir.

**Web Ara yüzlerinin Oluşturulması**: Altyapısı tamamlanan rezervasyon sisteminin kullanıcı ara yüzlerinin oluşturulması adımı işletilir. Bu bölümde tüm kural, model ve şablon aracılığı ile ara yüzler iki bileşen halinde oluşturulur. İlk kısım görsel öğeleri barındıran HTML ve CSS çıktıları ile oluşurken ikinci kısımda sistemin çalışması için gerekli olan prosedür ve özellikleri barındıran sayfalar üretilir.

**Stil Dosyalarının Eklenmesi**: Hazırlanan ara yüzlerin istenilen sektöre adaptasyonunda en önemli unsurlardan bir tanesi sitenin genel stilinin de otomatik üretilerek entegrasyonun tamamlanmasıdır. Bunun için şablon olarak harici bir stil dosyası oluşturulmuş; renk, boyut, yazı tipi, sayfa düzeni, vb. seçeneklerin kurulum seçimleri doğrultusunda ayarlanarak sistem içerisinde kullanılacak stil dosyasının oluşturulması sağlanmıştır.

İş akışının görsel olarak tanımlandığı ara yüzler Şekil 4'de gösterilmektedir. Kullanıcı; sistemin kullanacağı veri tabanını SQL Server veya Access olarak seçebilmekte, sol tarafta sunulan kontrolleri sürükle/bırak ile kullanarak tasarımını gerçekleştirebilmektedir. Nesnelerin arasındaki bağlantılar seri veya paralel olarak tanımlanabilmektedir. İş akışının temel kuralları kontrol edildikten sonra, çalışma modeli konfigürasyon dosyasına kayıt edilir.




**Şekil 4.** İş Akışının Modellendiği Ekran Görüntüsü

Bu yaklaşım sayesinde, rezervasyon sistemi son kullanıcı tarafından kolaylıkla oluşturulmakta ve programlama eklentisi yapmasına gerek kalmamaktadır. Son kullanıcı geliştirme kapsamında kolaylıkla kullanılabilecek olan bu yaklaşımda, tüm işlemler görsel öğeler kullanılarak gerçekleştirilmekte ve otomatik kod üretimi arka planda kullanıcıdan bilgi almadan çalıştırılmaktadır. Geliştirilen bu sistemde kullanılan yöntem ve prensiplerin, diğer son kullanıcı geliştirme ihtiyaçları için de uygulanabileceği, farklı durum çalışmalarıyla yöntemin genişletilebileceği değerlendirilmektedir.

Bu sistemin başarımını test etmek üzere örnek bir kuaför için bir uygulama geliştirilmesi sağlanmıştır. Kuaförün ihtiyaçlarına özel olarak çalışma modeli tasarlanmış ve bu doğrultunda web uygulaması yapılandırılmıştır. Web uygulamasının sunduğu tüm fonksiyonlara ait ara yüz testlerinin yapılması sonucunda, üretilen uygulamanın hatasız çalıştığı tespit edilmiştir. Mevcut iş modeline, yeni çalışanlar eklenerek yapılan güncellenmeler sonrasında da, oluşan kodlar ara yüz testlerinden başarıyla geçmiştir. Uygulamanın programlama bilgisine sahip olmayan son kullanıcılar tarafından kolaylıkla kullanılabildiği görülmüştür.

## 4. Sonuç

Son kullanıcıların yazılım geliştirebilmesi veya ihtiyaçlarına bağlı olarak bir yazılımı uyarlayabilmesi, değiştirebilmesi veya yeni eklentiler sunabilmesi halen üzerinde çalışılan bir araştırma alanıdır. Son kullanıcı geliştirme olarak ifade edilen bu alanda, çok sayıda yöntem ve yaklaşım önerilmiş, halen yeni yöntemler ortaya konulmaya devam edilmektedir. Birleştirme teknolojilerinden örnek tabanlı programlamaya, servis yönelimli mimarilerden görsel programlamaya kadar çok sayıda yaklaşım bu amaçla kullanılabilmektedir. Bu çalışmada; görsel programlama ve otomatik kod üretimi tekniklerinden yararlanılarak, son kullanıcı uygulamalarının etkin şekilde geliştirilip geliştirilemeyeceği değerlendirilmiştir. Örnek bir vaka çalışması olarak rezervasyon sistemleri ele alınmıştır. Görsel programlama ve otomatik kod üretiminin birlikte kullanıldığı durumda; son kullanıcı geliştirme işlemlerinin oldukça kolaylaştığı ve hedeflenen gereksinimlerin kolaylıkla karşılanabildiği görülmüştür. Yazılım uzmanından çok daha fazla sayıda iş ortamında kendi günlük işlerini programlar vasıtasıyla kolaylaştırmaya çalışan kullanıcılar olduğu düşünülürse, programlama bilgisinin minimum seviyede olduğu altyapıların oldukça kritik olduğu görülecektir.

Bu çalışmada; UML tabanlı kod üretimi yapan sistemlerin kullanımının, uygulama geliştirme süresini büyük ölçüde azalttığı görülmüştür. Bir aktivite diyagramından, hızlıca ve kolaylıkla hedeflenen yazılımın üretilebildiği somut bir durum çalışmasıyla ortaya koyulmuştur. Kullanıcıların teknik bilgiye sahip olmadan kendi başlarına kurulum yapabilmeleri, kullanımı yaygınlaştırmaktadır. Öte yandan, otomatik olarak üretilen kodun, geleneksel geliştirme yöntemlerine göre daha az optimize olduğu ve mevcut sistemler ile gereken entegrasyonun ek çaba gerektirmesi olumsuz yönler olarak değerlendirilmektedir. Son kullanıcı programlama veya geliştirmenin esasında, çok zor çözülen bir problem olmadığı, ilgili alanda yapılacak iyi bir analizle hedefe ulaşılabileceği görülmüştür. Gelecekte; geliştirilmiş olan altyapının rezervasyon sistemleri dışında daha farklı alanlara uygulanması planlanmakta olup, detaylı

kullanılabilirlik testleri o alanda görev alan kişilerle birlikte yapılacaktır. Alınacak geri bildirimlere göre geliştirilmiş olan yazılımın güncellenmesi planlanmaktadır. Ayrıca, diğer yöntemler kullanılarak benzer uygulamanın son kullanıcı geliştirme kapsamında gerçeklenmesi ve gerek geliştirme süresi gerekse geliştirme kolaylığı açısından karşılaştırmalı bir çalışma yapılması hedeflenmektedir.

**Kaynakça**


[1] Paternò, F., (2013). "End user development: Survey of an emerging field for empowering people". ISRN Software Engineering, 2013.

[2] Togay, C., Akkus, V., Dogru, A.H, (2014). "Son Kullanıcı Yönelimli Yazılım Geliştirme Aracı", UYMS 2014, Kuzey Kıbrıs Türk Cunhuriyeti, Eylül 08-10, 2014.

[3] Dogru, A.H., Togay, C., (2012). "Son Kullanıcı Geliştirme Ortamı için Aksiyomatik Tasarım Esinli Mimari", Ulusal Yazılım Mühendisliği Kongresi 2012 (UYMK 2012), İzmir, Turkey, Aralık 7-8, 2012.

[4] Long, H., (2012). "End user development of digital collection mash-ups: A survey to assess the suitability of current infrastructure". OCLC Systems & Services: International digital library perspectives, 28(4), pp.199-207.

[5] Herrington, J. (2003). Code generation in action. Manning Publications Co..

[6] Vavilina, E., & Gaigals, G. (2015, November). Improved LabVIEW code generation. In Information, Electronic and Electrical Engineering (AIEEE), 2015 IEEE 3rd Workshop on Advances in (pp. 1-4). IEEE.

[7] Li, S., Li, D., Li, F., & Zhou, N. (2015). CPSiCGF: A code generation framework for CPS integration modeling. Microprocessors and Microsystems, 39(8), 1234-1244.

[8] Mehmood, A., & Jawawi, D. N. (2013). Aspect-oriented model-driven code generation: A systematic mapping study. Information and Software Technology,55(2), 395-411.

[9] Tolvanen, J. P., Djukić, V., & Popovic, A. (2015). Metamodeling for Medical Devices: Code Generation, Model-debugging and Run-time Synchronization Procedia Computer Science, 63, 539-544.

[10] Solís-Martínez, J., Espada, J. P., García-Menéndez, N., G-Bustelo, B. C. P., & Lovelle, J. M. C. (2015). VGPM: Using business process modeling for videogame modeling and code generation in multiple platforms. Computer Standards & Interfaces, 42, 42-52.

[11] Kim, B., Phan, L. T., Sokolsky, O., & Lee, I. (2013, September). Platform-dependent code generation for embedded real-time software. In Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2013 International Conference on (pp. 1-10). IEEE.

[12] Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., & Fasih, A. (2012). PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. Parallel Computing, 38(3), 157-174.

[13] Graichen, C., & D'Amato, F. (2011, May). Adding code generation to develop a simulation platform. In Systems, Applications and Technology Conference (LISAT), 2011 IEEE Long Island (pp. 1-6). IEEE.

[14] Wang, Y., Ma, D., Zhao, Y., Zou, L., & Zhao, X. (2012, July). Automatic RT-Java code generation from AADL models for ARINC653-based

avionics software. In Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual (pp. 670-679). IEEE.

[15] Morkoc, C., Onal, Y., & Kesler, M. (2014, September). DSP based embedded code generation for PMSM using sliding mode controller. In Power Electronics and Motion Control Conference and Exposition (PEMC), 2014 16th International (pp. 472-476).

[16] Wang, J., Wang, P., & Li, Y. (2012, August). Research and Implementation of the Code Generation System Based on Production Rules. In Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on (Vol. 1, pp. 224-227).

[17] Lachgar, M., and Abdali, A. (2015). DSL and Code Generator for Accelerating iOS Apps Development. Third World Conference on Complex Systems (WCCS), Marrakech, Morocco, 2015, pp. 1-8.

[18] Bernardi, M. L., Cimitile, M., & Maggi, F. M. (2016, April). Automated development of constraint-driven web applications. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 1196-1203). ACM.

[19] Umuhoza, E., Ed-douibi, H., Brambilla, M., & Cabot, J. Automatic Code Generation for Cross-platform, Multi-Device Mobile Apps: Some Reflections from an Industrial Experience. Transformation, 2, M2T.

[20] P. Mohagheghi and V. Dehlen, "Where is the proof? — A review of experiences from applying MDE in industry," in MDA Foundations & Applications, 2008.

[21] Malavolta, I., Lago, P., Muccini, H., Pelliccione, P. and Tang, A., (2013). "What industry needs from architectural languages: A survey". IEEE Transactions on Software Engineering, 39(6), pp.869-891.