# İş Gereksinimi Odaklı Kaynak Kod Üretme Sistemi

**Ökkeş Emin Balçiçek[1], Büşra Tokgöz[2] , Havva Çetiner Altıparmak[3]**
[1] Kuveyt Türk Katılım Bankası , Ar-Ge Merkezi, Servis Yöneticisi, Konya
[2] Kuveyt Türk Katılım Bankası , Ar-Ge Merkezi, Yazılım Mühendisi, Konya
[3] Kuveyt Türk Katılım Bankası , Ar-Ge Merkezi, Kıdemli Yazılım Mühendisi, Konya

emin.balcicek@kuveytturk.com.tr , busra-tokgoz@kuveytturk.com.tr, havva.altiparmak@kuveytturk.com.tr,

**Özet:** Bu bildiride büyük ölçekli kurumsal firmalarda uygulanabilir, iş gereksinimi odaklı kaynak kod üretme modeli önerilmiş ve içermesi gereken özellikler anlatılmıştır. İş gereksinimlerini ekran tanımı, tasarımı, ekrandan alınacak veri tipleri, iş akışı, parametre tanımları ve merkezi sistem entegrasyon tanımlarının yapılabileceği tanım tabanlı bir araç aracılığı ile alınabileceği ifade edilmiştir. Önerilen sistem, iş gereksinimleri tanımları referans alınarak, gerekli uygulama sunucusu ve kullanıcı arayüzü kaynak kodları, veritabanı nesneleri, fonksiyonel analiz dokümanı ve test senaryo dokümanı üretebilmektedir. Bu sayede kurala bağlı üretilebilen bütün çıktılar üretilmiş olup, yazılım mühendisi, sistem analisti gibi pahalı kaynakların yerinde ve verimli kullanılması hedeflenmiştir. Önerilen sistem bu çıktıları üretmek için güçlü, oturmuş, olgunlaşmış bir kurumsal mimariye ihtiyaç duymaktadır.

**Anahtar Sözcükler:** Kaynak Kod Üretimi, İş Gereksinimi Odaklılık, Yazılım Geliştirme Aracı

**Business Oriented Source Code Generation System**

**Abstract:** In this paper, business oriented source code generation system in large-scale enterprises, is proposed and explained in detail. Business requirements enter the system via user interface definition, user interface design, data types, workflow definition, parameter definiton and centralized system integration definition. Proposed system, generates user interface side, application server side and database item source code automatically by using business requirements which is gathered from system analyst role by a CASE tool. This system can also produce functional requirement and test scenario documents. By this means, important part of source code, which able to automatize, is produced and big-time of software engineering and business analysis time is gained. This system needs robust, structured, matured and well-designed enterprise architecture to produce these outputs.

**Keywords:** Source Code Generation, Business Oriented, CASE Tools

## 1. Giriş

Kod üretimi farklı manalarda kullanılan bir terimdir. Bir modelden veya yapısal bir formdan kaynak kodu üretiliyorsa o zaman kaynak kodu üretimi, bayt kodu veya asembler gibi kodlar üretiliyorsa orta seviye kod üretimi ve op-code yani çalıştırılabilir kod(executable code) üretiliyorsa düşük seviyeli kod üretimi manasında kullanılır. Orta seviyeli ve düşük seviyeli kod üretimi işlemine derleyicilerin bir parçası olarak bakılabilir. Ayrıca bir koddan aynı seviyede başka bir koda dönüştürmek de kod üretimi olarak bilinir.

Bu bildiride sadece iş gereksinimi odaklı kaynak kod üreticisi üzerinde durulacak ve geliştirilen kaynak kod üreticimizden ve geniş çaplı yazılım projelerinde kaynak kod üretiminin avantajlarından söz edilecektir.

Derleyicilerle birlikte sunulan standart uygulama geliştirme ortamları (UGO), kullanılan bileşenlere ilişkin temel kaynak kodu üretme imkanı sunmaktadırlar. UGO'lar ilişkisel yazılım geliştirmede yetersiz kalmakta ve bu işlemler için yeni geliştirme ortamlarına ihtiyaç duyulmaktadır.

Kaynak kod üretme sistemleri, genellikle büyük ölçekli sistemler geliştirilirken içerisinde, tekrar tekrar üretilmek zorunda kalınan, saklı yordamlar (stored procudures), kullanıcı arayüz kodları ve uygulama sunucusu kodları gibi yazılım mühendisinin vaktini alan kod parçalarını dinamik olarak üreten programlardır. Özellikle kurumsal büyük ölçekli kurumlarda kullanılan katmanlı mimariye[6] sahip yazılım sistemleri için önerilmektedir. Bu tür yazılımlara bankacılık sistemleri, telekominikasyon yazılımları, SGK gibi devlet kurumlarının kullandığı büyük ölçekli otomasyonlar örnek verilebilir. Binlerce kullanıcıya hitap eden bu tür sistemlerde bütün ekranların adeta tek bir usta elinden çıkmış gibi görünmesi ve davranması beklenmektedir.

Bu büyüklükte sistemlerin sahip olması gereken en büyük özelliklerden birisi ise yazılım üretim bandıdır. Yani kod geliştirme standartlarının, sanal ve fiziksel katmanların geliştirilmiş ve olgunlaştırılmış olması gerekmektedir. Bu yazılım üretim bandı sayesinde geliştirilecek bir kaynak kod üretimi sistemi aracılığı ile yazılım kodları en az emekle ve tek usta elinden çıkmış kadar standart ve okunabilir bir şekilde üretilebilir. Böylelikle, bakımı kolay ve standartlara uygun uygulamalar geliştirmek, daha az zaman ve emek harcanarak gerçekleştirilmiş olmaktadır.

Daha önceki Source Code Generator for Large Scale Applications [4] isimli bildirimizde, veritabanında bulunan tablo, görünüm(view) veya saklı yordam nesnelerini referans alarak; saklı yordam, orta katman kodları ve kullanıcı arayüzü kodları üreten bir model anlatılmıştı. Bu bildiri ile ekran tasarımlarında yer alan bileşenler ve aksiyonlar da referans alınarak tablo, görünüm, saklı yordam, orta katman uygulama kodları ve ekran tasarım kodlarının da üretilebileceği daha gelişmiş bir sistem anlatılacaktır. İş gereksinimlerini giriş olarak aldığı için, fonksiyonel analiz ve test senaryosu dokümanları da otomatik olarak üretilebilecektir.

İlerleyen bölümlerde yukarıda bahsedildiği üzere, iş gereksinimi odaklı kaynak kod üretimi yapan bir modelden ve bu modelin kullanılması ile oluşturulan bir uygulama dizaynından ve özelliklerinden bahsedilecektir. Bundan sonraki başlıklar şu şekilde organize edilmiştir. 2. bölümde bu çalışma ile ilgili daha önceki akademik çalışmalar taranmış ve ilgili yönleri tartışılmıştır, 3. bölümde bütün kurumların kendi ihtiyaçlarına göre iş gereksinimli kaynak kod üretici üretebilecekleri örnek bir model ortaya konulmuştur. 4. ve son bölümde ise bu modeli kullanım önerileri anlatılmıştır.

## 2. İlgili Çalışmalar

Kaynak kod üretme sistemleri, ileri mühendislik çalışmalarını destekleyen çok önemli araçlardır. Bu araçlar, yazılım nesneleri tanım bilgilerini okur ve çok farklı programlama dilleri için kaynak kod üretir. [1]

Kod üretme sistemleri, bir çok farklı alanlarda kullanılmış ve halen de kullanılmaya devam eden çok fonksiyonel bir araçtır. Daha önceden yapılan çalışmalar incelendiğinde çok farklı alanlar için kod üretme sistemleri tasarlandığı görülür. Mesela; [1]'de UML CASE araçları için XML tabanlı kod üretme sisteminin dizaynı

ve içeriği anlatılmaktadır. XML tabanlı olması bizim yaklaşımımızla ortak noktasıdır. Fakat bu modelde şablonlar kullanılmamıştır. Onun yerine UML sınıfları ve farklı programlama dilleri arasındaki ilişkileri adreslemek için kural tanımlayıcılar kullanılmıştır. Bu şekilde bir yaklaşım büyük ölçekli yazılımlar için tam istenilen esnekliği sağlamamaktadır. Sistem analistleri UML modelleme araçları ile beklenen somutlaştırmayı sağlayamadıkları için bu bildiride, tablo tanımı, prosedür tanımı, ekran dizaynı gibi daha somut uygulama nesneleri üzerinden kod üretilmesi önerilmiştir.

Stevica ve Dragan[2]'a göre Object Relational Mapping (ORM) sistemlerinden NHibernate gibi araçlar, kaynak kod üretimine veritabanı katmanında alternatif bir yaklaşım olarak kullanılmaktadır. Bu bildiride bu araçların performans sıkıntılarından dolayı Statik ORM yaklaşımı önerilmektedir. ORM tabanlı NHibernate vb. araçlardaki dinamik yapı dolayısı ile performans kaybı yaşanmaktadır. Statik ORM yaklaşımı ile kodlar derleme seviyesinde üretilererek performansı daha yüksek tekrar kullanılabilen kod parçaları kolayca elde edilebilmektedir. Buna ek olarak üretilen koda manuel müdahale yapılabilmesi ayrıca büyük bir esneklik sağlamaktadır.

Daha önceden yayınlanmış makalelerde önerilen bir başka modelde ise, yazarın kendi geliştirdiği bir sistem olan SCT (Specification-Configuration-Templates) dinamik çerçeve tabanlı bir kaynak kod üretme sistemi[3] , Python programlama dilinde geliştirilmiştir. Bu uygulamada Python'un uygulama dili olarak tercih edilmesinin sebebi, nesne yönelimli herhangi bir script dili ile esnek bir şekilde çalışabilmesidir. Gerçekleştirilen bu uygulamanın en büyük avantajı, tam konfigüre edilebilir olması, kod üretim maliyetini düşürmesi ve portatif olmasıdır. Fakat dinamik çerçeve kullanılması performans açısından dezavantaj sağlamıştır. Biz bu yaklaşım yerine Statik ORM yaklaşımını modelimizde uygulayarak büyük ölçekli yazılımlarda karşılaşılabilecek performans sorununa da çözüm bulmuş oluyoruz.

Yaklaşık 300 şubeli bir katılım bankasında uygulanan ve Source Code Generator for Large Scale Applications [4] bildirimizde anlatılan model büyük ölçekli kurumsal uygulamalarda kullanılabilmektedir. Bu çalışmamız tablo, saklı yordam veya tablo görünümünden referans alınan veriler ile gerekli kaynak kodların üç katmanlı mimari için, belirlenmiş standartlarla her katman için kod üretimi yapabilen bir modeli anlatmaktadır. Bu bildirimiz ile bu altyapı kullanılarak iş gereksinimi odaklı bir kod üretimi sağlanacak, gerçek dünyaya bir seviye daha yaklaşılabilecektir.

Bu kısımda bahsedeceğimiz son modelde [5] ise düzenli ifadeler (regular expression) kullanılarak bir kaynak kod üretme sistemi geliştirilmiştir. Regular expression sistemleri anlaşılması açısından oldukça karmaşık olması nedeni ile gündelik hayatta kullanımı oldukça zordur. Bu nedenle insan seviyesine yakın dillerden kullanılması önerilmektedir.

## 3.İş Gereksinimi Odaklı Kaynak Kod Üreticisi Modeli

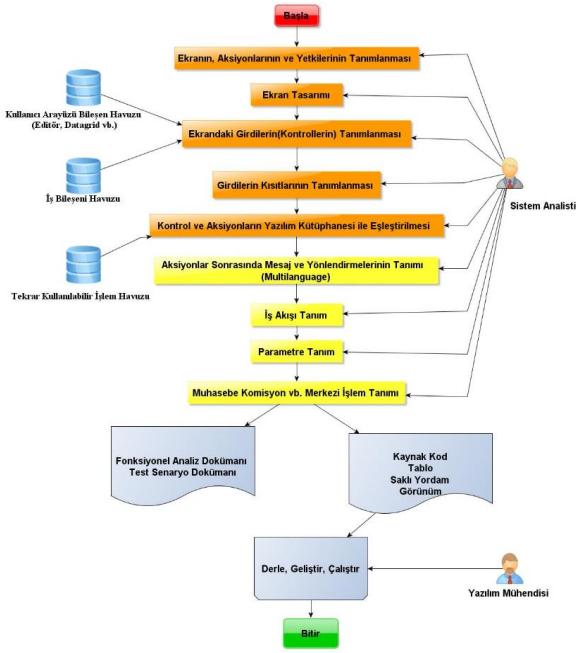
Yazılımların üretilmesinin temel amacı olan iş gereksinimlerinin, girdi olarak otomatik kaynak kod üretiminde kullanılabilmesi, kod üretimini nihai koda yaklaştırıcak ve gereksinimlerle uygulamanın arasındaki bağı kuvvetlendirecektir. Bu kapsamda, girdi olarak kullanılabilecek iş gereksinimlerini; ihtiyaç duyulan ekran tasarımı, ekrandan alınacak veri bilgileri ve kullanıcı arayüz kontrolleri, aksiyonlar, aksiyon sonrasındaki yönlendirme ve mesajlar, muhasebe ve komisyon gibi merkezi işlemler ve son olarakta kullanıcılar arası onay ve işlem akışını içeren iş akışı şeklinde sıralayabiliriz.

Önerilen modelde daha önceki bildirimizde anlatılan Source Code Generator for Large Scale Applications (SCG) [4] modeli temel alınmış ve yukarıda bahsedilen kapsamdaki iş gereksinimi odaklı yeni bir yaklaşım ortaya konulmuştur. Tablo, görünüm ve saklı yordam girdilerinin yanında iş gereksinimleri de girdi olarak alınabilmekte ve bu girdiye göre kod üretimi gerçekleştirilebilmektedir. Bununla beraber veritabanı, orta katman ve kullanıcı arayüzü kodlarının yanı sıra, fonksiyonel analiz ve fonksiyonel test senaryosu dokümanları da otomatik olarak üretilebilecektir.

Bu sistemde kod üretimi için oluşturulması gereken ilk girdi, ihtiyacın netleşmesi ile ortaya çıkan ekranın bu sistem üzerinde tasarlanması ile başar. Bu ekran tasarımı sistem analistinin bu modele istinaden geliştirilecek olan bir tasarım aracı üzerinde gerçekleştireceği ekran çizimi olarak düşünülebilir. Ekran üzerindeki alanlar, aksiyonların gerçekleşmesi için konulmuş butonlar ve alanların veri tipi gibi bilgiler kod üretimi için oldukça değerli girdilerdir. Referans alınan bu girdiler sayesinde bu iş gereksinimleri için gerekli veritabanı tablo yapısı için gerekli kaynak kodu kolayca üretilebilir. Bu tablo üretildikten sonra gerekli katman kod üretimleri daha önce yayınlanmış olan SCG [4] bildirimizde anlatılan, tablodan kod üretme seçeneği aynen uygulanarak üretilebilir.

Modelin gerçeklenebilmesi için yazılım üretim bandının olması, dolayısı ile yazılım geliştirme standartlarının ve tasarım kalıplarının oluşturulmuş ve oturmuş olması oldukça önemlidir. Böylelikle daha efektif, kontrolü ve yönetilebilirliği kolay kod parçalarının kod üretici tarafından üretilebilmesi mümkün olmaktadır.

Sistem analistinin iş gereksinimlerini toparlayarak, varolan sistemi analiz ederek ve yeni modülü kurgulayarak bu bilgileri sisteme girdi olarak uygulama aracılığı ile tanıtabilmelidir. Analistler bu araç ile ekran tasarımı, ekrandan alınacak veri bilgileri ve kullanıcı arayüz kontrolleri, aksiyonlar, aksiyon sonrasındaki yönlendirme ve mesajlar, muhasebe ve komisyon gibi merkezi işlemler ve son olarakta kullanıcılar arası onay ve işlem akışını içeren iş akışı gibi bütün tanımlamaları yapabilmelidir. Bu tanımlamaların yapılmasından sonra sistemden çıktı olarak; veritabanı tablosu, veritabanı görünümü, saklı yordamlar, orta katman uygulama kodları, kullanıcı arayüzü kodları, fonksiyonel analiz ve fonksiyonel test dökümanları verilebilmelidir.




**Şekil 1.** Kod Üretim Modeli Süreci

İş gereksinimlerinin kodsal karşılığını adresleyebilmek, daha önce çözümlenmiş gereksinim kodlarını tekrar kullanabilmek adına yazılım nesne ve fonksiyonların yer aldığı bir yazılım kütüphanesi oluşturulması modelin başarımını artıracaktır. Sistem analisti tasarım anında kullanıcı bileşenleri ve aksiyonlar ile bu kütüphanedeki fonksiyon ve metodları eşleştirmek sureti ile yazılım

mühendisine ilgili fonksiyonları hazır ve bağlı halde sunabilecektir.

Uygulamanın, fiziksel ve sanal yazılım katmanları için üretmiş olduğu kodlar kurum içerisinde kullanılan uygulama geliştirme ortamı ve uygulama geliştirme standartları dahilinde üretilmiş olarak, derlenebilir bir halde yazılım mühendisine aktarılır. Yazılım mühendisi bu kod ve dokümanları temel alarak otomatize edilemeyen talepleri karşılamak amacı ile kendisine aktarılan özelleştirmelerini yaparak geliştirme işlemini tamamlar. Tüm süreç şekil 1 'de gösterilmektedir.

İş gereksinimli kaynak kod üretim süreci aşağıdaki bölümlerde daha ayrıntılı olarak anlatılmıştır.

### 3.1.Ekranın, Aksiyonlarının ve Yetkilerin Sisteme Tanıtılması

Büyük ölçekli kurumlar için geliştirilmiş sistemlerde, kullanıcı arayüzleri, tanım ekranından sisteme tanıtılmalıdır. Her bir ekranın hangi menü altında görüneceği, hangi aksiyonlar içereceği ve bu aksiyonlara kimlerin erişim yetkisinin olabileceği belirtilmelidir. Bu aksiyonların işlem çeşidi daha başından belirlenebilmeli, ekranın davranışları bu tanımlara göre yönlendirilebilmelidir. Bu ilk adımla ekranın ne olduğu ifade edilmiş ve tasarlanması için hazır bir ortam sağlanmış olur.

### 3.2.Ekran Tasarımı ve Kontrollerinin Yerleştirilmesi

Ekranın aksiyonlarının ve yetkilerin sisteme tanıtılmasının ardından, ekranın tasarım ortamında dizayn edilmesi yani ekranda bulunması istenilen bileşenlerin daha önceden sisteme tanımlanmış olan bileşen havuzlarından (kullanıcı arayüzü bileşen havuzu, iş bileşen havuzu vb.) seçilerek sürükle - bırak yöntemi ile ekrana yerleştirilmelidir. Ekrana yerleştirilecek olan kontrollere örnek olarak; metin/sayısal alanlar, seçilebilir listeler (combobox), radio butonlar verilebileceği gibi bileşenlere de müşteri bileşeni, personel bileşeni, adres bileşeni örnek olarak verilebilir.

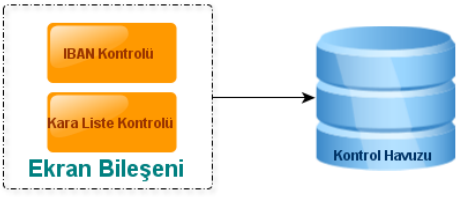
Sistem analisti tarafından yapılacak olan ekran tasarımında yer alan bileşenler ve kontroller iş gereksinimi odaklı kaynak kod üretiminin en temel girdileridir. Bu girdiler ile öncelikle veri tabanındaki tabloların alanları ve veri tipleri belirlenecektir. Aşağıdaki tabloda bu ilişki birkaç örnek üzerinden gösterilmektedir.

| Bileşen/ Kontrol Adı | Bileşen/ Kontrol Tipi | Tablo Alan adı | VeriTipi |
|---|---|---|---|
| EFT Tutarı | Numeric Editor | AmountOfEFT | numeric |
| EFT Tarihi | Datetime Editor | EFTDate | datetime |
| Komisyon Bilgileri | Commission Component | CommissionSerializeId | int |

**Tablo 1.** Bileşen/Kontrol tanımları ve oluşacak tablo alan bilgileri

### 3.3. Girdilerin Kısıtlarının Tanımlanması

Bir önceki adımda ekrana yerleştirilmiş olan bileşen ve kontroller genel kullanım amacıyla tasarlandığından, yerleştirildiği ekranın iş gereksinimlerini sağlayacak doğrultuda özelleştirilmelidir. Kod üretim sürecinin bu adımında özelleştirme için gerekli tanımlamaların yapılması sağlanmaktadır.



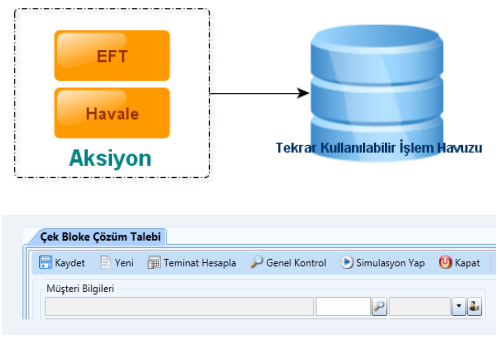

**Şekil 2.** Bileşen – Kontrol Havuzu eşleşmesi

Ekranda yer alan bileşenlere veri girişleri yapılırken kullanıcının bazı kriterlere dikkat etmesi istenilebilir. Örneğin, TC Kimlik No alanına karakter girilememesi (Karakter Kontrolü), tutarın 0 dan küçük bir değer

almaması (Negatif Değer Kontrolü) vb. bu tür kontroller Kontrol Havuzu'nda tanımlanmış ve kodsal karşılığı adreslenmiş olmalıdır.

### 3.4. Kontrollerin ve Aksiyonların Yazılım Kütüphanesi ile Eşleştirilmesi

Girdi kısıtları tanımlanmış olan bileşen ve kontroller ile ekran tasarımı tamamlanmış olmaktadır. Bu aşamadan sonra, ekran tanım aşamasında sisteme daha önceden tanımlanmış olan ekran aksiyonlarının (EFT, Kaydet, Güncelle, Sil vb.) yazılım kütüphanesinde yer alan tekrar kullanılabilir işlem havuzundaki kodsal karşılıkları ile eşleştirmelerinin yapılması gerekmektedir. Bu aksiyonlar ekrandaki araç çubuğu üzerinde bulunan ve ana işlemleri yapan ana düğmeler (buton) olarak düşünülebilir. Aksiyonlar Şekil 3 ile bu eşleştirmeye verilebilecek örnekler gösterilmektedir.




**Şekil 3.** Aksiyon ve Kontrollerin Kullanılabilir İşlem Havuzu ile Eşleştirilmesi

Havuzda bulunan tekrar kullanılabilir metod ve fonksiyonlar, mümkün olduğunca parametrik ve dinamik bir yapıda hazırlanmış olmalıdır. EFT aksiyonu ele alındığında, bu aksiyona yani düğmeye basıldığı zaman hangi işlemin çalışması gerektiği tasarım anında parametrik bir şekilde belirlenebilmelidir. Böylelikle ekran tasarımından sonra, daha önceden hazırlanmış kullanıcı arayüz ve orta katman kodları aksiyonların arkasındaki olay (event) ve metodlara otomatik olarak eklenebilecek, yazılım mühendisini kod parçası veya fonksiyonu aramak zahmetinden kurtaracaktır. Bu sayede gerekli kütüphane referanslı da projeye varsayılan olarak eklenmiş bir halde gelecektir. İşlem oldukça net ve kurala bağlı ise hiç kod yazmaksızın çalıştırılabilecektir. İşlem müdahele gerektiriyor ise eklenen standart kod parçaları revize edilmek sureti ile kod geliştirme tamamlanabilecektir.

### 3.5. Ekran Mesaj ve Yönlendirmelerin Tanımı

Ekranlarda kullanıcıyı bilgilendirmek ve yönlendirmek amacı ile verilen mesajlar ekran tasarım anında tanıtılabilmelidir. Bu mesajlar senaryo bazlı girilebilmeli, mesajı okuyan kullanıcının kararına göre nasıl hareket edilmesi gerektiği tasarım anında kararlaştırılabilmelidir. Sisteme statik olarak tanıtılan bu mesajlar diğer ekranlarda da kullanılabilmelidir. Mesajlar birçok dilde tanımlanarak dil desteğinin üst seviyelere taşınması sağlanabilir.

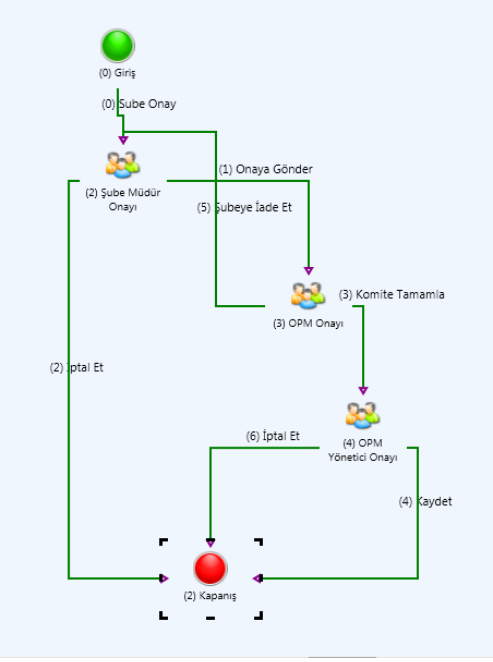
### 3.6. İş Akışı Tanımı

Ekranda yer alması gereken iş akışları bu aşamada tanımlanır. Kullanıcılar arası onay ve işlem akışını içeren iş akışı tasarımı, kurumsal mimari içerisinde daha önceden hazırlanmış olan iş akışı tasarım aracı sayesinde çizilebilmeli, ekranlarla ve aksiyonlarla bağlanabilmeli, gerekli fonksiyonlar ve kullanılacak veri tipleri belirlenebilmelidir.

Örnek bir senaryo üzerinden gidecek olur isek, bankacılık sistemlerinde bulunan fon kullandırım modülü incelendiğinde anlatılmak istenen daha iyi anlaşılacaktır. Bu süreçteki iş akışı şöyledir: müşterinin talebi şubede görev alan uzmanın girmiş olduğu veriler ile şube müdürünün onayına gönderilir. Daha sonrasında onaylanması ve reddedilmesi aksiyonunda hangi durumlara gönderileceği iş akışı tasarlama aracı ile tasarlanır. Şekil 4 ile örnek verilen bu sürecin tasarımı yer almaktadır. Operasyon uzmanına gönderilen talep operasyon

uzmanınca eklenecek verilerin girilmesiyle operasyon yöneticisinin onayına gönderilmektedir. Operasyon yöneticisinin karar alacağı aksiyon ile akış tamamlanmış ve tasarımı çizilen iş akışı süreci de gerçekleştirilmiş olmaktadır.

Ekran tasarım anında iş akışı da tasarlanabilmeli, ekran, aksiyon ve iş akışı arasındaki bağ kurulabilmelidir.




**Şekil 4.** İş akışının tasarlanması

### 3.7. Parametre Tanımı

İçerisinde liste barındıran ekran bileşenlerinde kullanılabilecek statik tanımlamalar parametre tanımlama işlemi ile gerçekleştirilebilir. Bu sayede liste bileşenlerinin içerisine hangi verilerin doldurulacağı daha tasarım anında karar verilmiş, kodu üretilmiş olacak, yazılım mühendisine bu konuda geliştirme kalmayacaktır. Kurala bağlı üretildiği için de standart kod ve standart davranışlar otomatik olarak elde edilmiş olacaktır.

### 3.8. Komisyon, Muhasebe ve Merkezi İşlemlerin Tanımı

Muhasebe, komisyon gibi merkezi işlemlerin ürettikleri satırlar ve sütunlar devlet ve kurum prosedürlerine bağlı olduğu için oldukça standart ve kurala bağlı olabilmektedir. Bu bağlamda ekrandan üretilmiş olan girdilerin bu merkezi sistemlerde hangi alana eşleştirilebileceği önceden tanımlanabilen senaryolardandır.

Ekran tasarımı anında bu merkezi modüllere yapılacak olan entegrasyonların tanımlanması ve ekranla ilişkilendirilebilmesi oldukça mümkün görünmektedir. Bu vesile ile her bir yazılım mühendisi tekrar tekrar bu merkezi yapıları öğrenmek zorunda kalmayacak, gerekli kod parçaları çalışma zamanında üretilmek sureti ile sistem otomatize edilebilecektir. Yine standart kod, standart ekran ve sistem davranışı bu yöntemle sağlanabilecektir.

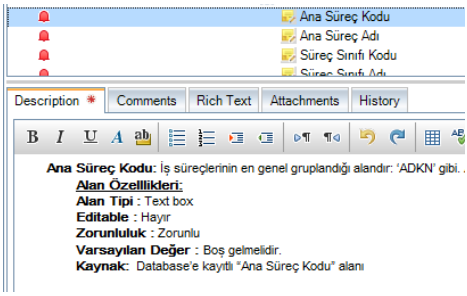
### 3.9. Doküman Çıktılarının Oluşturulması

İş gereksinimlerinin ekran tasarım aracı ile sisteme tanımlanması dolayısı ile döküman çıktıları ihtiyaç duyulan ekran tasarımları, ekranda yer alan iş gereksinimleri, kontroller, iş akışı tanımları, parametre ve merkezi sistem entegrasyon tanımları zaten elimizde bulunmaktadır. Bu girdilere göre fonksiyonel analiz dökümanı, test senaryosu dökümanları otomatik olarak üretilebilinecektir. Bu sayede analiz ve test süreci girdileri de bir üretim bandına ve standartlara bağlanmış olmaktadır.

Yazılım test süreci için gerekli olan test senaryolarının belirlenmesinde ekranda yer alan bileşenlerde belirlenen özellikler, analistin eşleştirmiş olduğu fonksiyonlar ve etki alanları da havuzda yer aldığından test senaryo dokümanının oluşturulmasında büyük katkı sağlamaktadır.

Kod üreticide tanımlanan alanlar ve özellikleri, aksiyonlar ve fonksiyon

eşleştirilmeleri birleştirilerek doküman oluşturulur. Şekil 5 de ekranda yer alan bileşenlerin özellikleri belirlenmiştir.




**Şekil 5.** Bileşenlerin özelliklerinin tanımlanması

Şekil 6'da ise tanımlanan özelliklerin dokümante edilmiş hali yer almaktadır.

Ekran üzerinde sürecin seçileceği 'Süreç Bilgileri' alanında yer alacak alanlar:

- **Ana Süreç Kodu:** İş süreçlerinin en genel gruplandığı alandır: 'ADKN' gibi. Ana Süreç Adı seçilince database'deki ilgili kayıtlı değeriyle otomatik olarak gelmelidir.

  **Alan Özelllikleri:**
  **Alan Tipi :** Text box
  **Editable :** Hayır
  **Zorunluluk :** Zorunlu
  **Varsayılan Değer :** Boş gelmelidir.
  **Kaynak:** Database'e kayıtlı "Ana Süreç Kodu" alanı

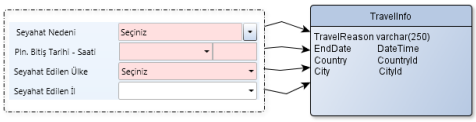
- **Ana Süreç Adı:** Kullanıcı tarafından üzerine riskin tanımlanacağı Ana Süreç bu alandan seçilmelidir.

  **Alan Özelllikleri:**
  **Alan Tipi :** Combo box

**Şekil 6.** Bileşen bilgilerinin dokümana aktarımı

### 3.10.Tablo, Saklı Yordam ve İş Gereksinimli Kaynak Kod Üretimi

Ekran tasarlanmasıyla belirlenen alanlar ve veri tipleri ile tüm tablo tasarımları kod üretici tarafından üretilir. Text alanların belirlenmesi ile tabloya alfanümerik (string) türde veri alanları, numerik alanların belirlenmesiyle tabloya integer türde veri alanları tanımlanması buna örnek olarak verilebilir. Şekil 7 de tablo ve bileşen eşleştirmesi gösterilmektedir.

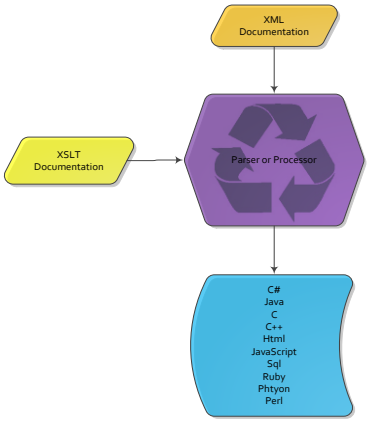



**Şekil 7.** Bileşen tablo eşleşmesi

Her bir tablo için kaydetme, güncelleme, silme ve listeleme iş gereksinimleri saklı yordam ve kaynak kod olarak üretilmektedir. Tanımlanan her bir ekran için bir proje kaynak kod dizininin oluşturulması sağlanır. Yukarıda anlatılmış tüm bu sistem xslt teknolojisi ile gerçekleştirilir.

#### 3.10.1. Kaynak Kod Üretimi

Xslt herhangi bir Xml içeriğini farklı bir Xml, Html, Csv (Comma Seperated Values) veya text formatına dönüştürme işlemi ile ilgili materyalleri sağlayan bir işaretleme dilidir[4].




**Şekil 8.** XML'den XSLT 'ye dönüşüm şeması [4]

Kullanıcıdan alınan verilere göre xslt teknolojili şablonlar ile gerekli kodlar oluşturulmaktadır. En önemli husus ortadaki parser ve XSLTransform işlemidir. Aşağıdaki şekilde xslt uzantılı şablon dosyasının içeriği verilmiştir.

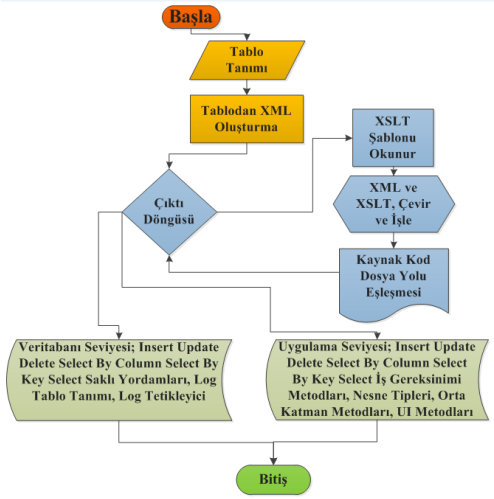
```
<?xml version="1.0" encoding="utf-8"?>
<!-- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"></xsl:output>
  <xsl:template match="/Tablo"> -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">
    <xsl:output method="text" indent="yes"/>

  <xsl:include href="GeneratorInformation.xslt"/>
  <xsl:include href="BusinessObjectForTable.xslt"/>

    <xsl:template match="/Table">

  <xsl:variable name="FileName"><xsl:value-of select="@ClassName"/>.cs</xsl:variable>

<xsl:call-template name="GeneratorInformation">
  <xsl:with-param name="FileName" select="$FileName"></xsl:with-param>
  <xsl:with-param name="SqlOrCs" select="'Cs'"></xsl:with-param>
</xsl:call-template>

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using BOA.Common.Types;
using BOA.Base;
using BOA.Base.Data;
using BOA.Types.<xsl:value-of select="@ProjectName"/>;

namespace BOA.Business.<xsl:value-of select="@ProjectName"/>
{
  /// <summary>
    /// Auto generated Business Object class of
    "<xsl:value-of select="@Schema"/>.<xsl:value-of select="@Name"/>" table.
    /// </summary>
  <xsl:call-template name="BusinessObjectClassForTable">
  </xsl:call-template>
}
  </xsl:template>
</xsl:stylesheet>
```

**Şekil 9.** Xslt uzantılı şablon dosyasının içeriği[4]

### 3.10.2.Tablodan Kaynak Kod Üretimi

Arayüz tasarımlarında tanımlanacak kontroller ve içerisindeki veri tipleri, seçilmiş olan tablonun kolonlarındaki veri tiplerine göre belirlenmelidir. Böylelikle kod üretim sürecinde gerekli tüm adreslemelerin yapılacağı veri tipleri oluşturulabilir.

Kod üretimi için, şekil 10'da bulunan akış diyagramında da belirtildiği üzere öncelikle tablo tanımı girdi parametresi olarak alınmalıdır. Tablo tanımında yer alan kolonlara ve yazılım mühendisinin yaptığı seçeneklere göre XML dosyası oluşturulmalıdır. Oluşturulan XML dosyası ve belirtilten XSLT şablonu işlenerek istenilen veri tabanı nesneleri kodları, orta katman kodları ve UI kaynak kodları oluşturulmalıdır. Veri tabanı nesnesi olarak; Insert, Update, Delete, Select, SelectByKey, SelectByColumns saklı yordamları ile log tablo oluşturma scriptleri ve tetikleyiciler(trigger) oluşturulmadır.




**Şekil 10.** Tablodan kaynak kod üretme algoritması[4]

Uygulama seviyesinde ise, veri tabanı nesnesi olarak oluşturulan saklı yordamların veri erişim katmanı metodları ve iş katmanı metodları için kaynak kodlar üretilmelidir. Bunlara ilave olarak, katmanlar arası veri taşıma işlevini görecek olan veri paketleri sınıfları da otomatik olarak üretilmelidir.

Sonuç olarak üretilen kodlar hem ilgili dosyalara yazılmalı hem de yazılım mühendisinin kopyalayarak kendi yazılımına ekleyebileceği bir sonuç penceresinde gösterilmelidir. Yazılım mühendisi isterse belirtilen dosyaları oluşturulduğu klasör içerisinden sürükle – bırak yöntemi ile projesine ekleyebilmelidir.

### 3.11.Derleme, Geliştirme ve Çalıştırma

Yazılım mühendisine kadar ulaşan kaynak kodlar, kod dizini yapısıyla oluşmuş durumdadır. Kod dizininin derlenip çalışmasıyla kaynak kod üretici süreci sonlandırılmış olur.

## 4. Sonuç ve Öneriler

Bu bildiride bir kaynak kod üreticisine iş gereksinimlerini de kazandırmanın nasıl olması gerektiğinden ve çok katmanlı yazılım mimarileri için sistem analisti, yazılımcı ve test mühendisleri pozisyonlarına sağladığı avantajlardan bahsettik. Önerilen bu sistem, analiz aşamasında tasarım ve dokümante kolaylığı sağlayarak, ortaya çıkabilecek analiz uygulama uyuşmazlığını da minimuma indirecektir.

Modelin büyük ölçekli kurumsal firmalarda veri katmanları, bileşen havuzları ve fonksiyon havuzları oluşturarak kullanılması ayrıca bir standardizasyon ve beraberinde bakım kolaylığı sağlayacaktır.

Kurumun tüm uygulamalarını kapsayabilecek fonksiyonel bir revizasyon için fonksiyon havuzunda gerçekleştirilebilecek ufak bir değişiklik ile dinamik yapının sağlamış olduğu avantajlardan yararlanılabilir.

Ortaya çıkan ürünlerden en önemlisi olan iş gereksinimlerini de içeren kaynak kodların, kaynak kod dizininde derlenip çalıştırılarak yazılımcının standartlaşmış gereksinimler için daha az ve efektif gereksinimler için daha verimli iş gücü harcamasına katkı sağlayarak oluşturulmasıdır.

## 5. Kaynaklar


[1] D. H. Park, and S. D. Kim, "XML Rule Based Source Code
Generator for UML Case Tool", Sangd-dong, Dongjak-Ku, Seoul,Korea, 0-7695-1083-3/01 2001 IEEE.

[2] S. Cvetkovic, and D. Jankovic, "A Comparative Study of the Features and Performance of ORM Tools in a .Net Environment" , **Serbia, Objects and Databases Lecture Notes in Computer Science** vol. 6348, pp. 147-158, 2010.

[3] D. Radoševi, and I. Magdalenic, "Python Implementation of Source Code Generator Based on Dynamic Frames", **Universitiy of Zagreb, Faculty of organization and informatics**, Varaždin, Croatia, MIPRO 2011, May 23-27, 2011.

[4] Altıparmak Ç.H., Tokgöz B., Balçiçek Ö.E., Arslan A. and Özkaya A., "Source Code Generation for Large Scale Application", **IEEE International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)**, 404 - 410 (2013).

[5] M. C. Franky, and J. A. Pavlich-Mariscal, "Improving Implementation of Code Generators: A Regular-Expression Approach", Pontificia Universidad Javeriana Bogotá, Colombia, 978-1-4673-0793-2/12/ ©2012 IEEE.

[6] M. C. Kus Khalilov, and M. Gundebahar, "Çok KatmanlıMimariler için Senkronize ve Bütünleşik Veri Sözlüğü Yapılanması", 4. Ulusal Yazılım Mimarisi Konferansı(UYMK'12), December 7-8, 2012