Aurora Yazılım Üretim Bandı

N. İlker ALTINTAŞ[1], Mehmet SURAV[2], Oğuz KESKİN[3], Semih ÇETİN[4]

[1, 2, 3,] Cybersoft Enformasyon Teknolojileri Ltd. Şti.
Ataşehir Bulvarı, Ata Plaza 3/3, 34758, Ataşehir, İstanbul
[4] Cybersoft Enformasyon Teknolojileri Ltd. Şti.
Silikon Blok, No:18, 06531, ODTÜ Teknokent, Ankara

{[1]ilker.altintas, [2] mehmet.surav, [3] oguz.keskin, [4] semih.cetin}@cs.com.tr

Özet

Bu bildiride "Aurora" olarak bilinen; çok katmanlı ve Web temelli mimarilerde yazılım geliştirme sürecini hızlandıran bir Yazılım Üretim Bandı (YÜB) anlatılmaktadır. Aurora; temelde Zengin İnternet Uygulamaları ve Kurumsal İnternet Uygulamaları modellerini bünyesinde barındırması sayesinde sadece Web temelli uygulamalar için eksiksiz bir yol haritası sunmakla kalmayıp; YÜB'ını yazılım geliştirme süreç yönetimi, WYSIWYG tasarım ve geliştirme ortamları, yazılım yaşam döngüsü yönetim teknikleri ve kalite yönetim araçları bağlamında hayata geçirebilmektedir. Ayrıca bu bildiride "çekirdek varlıkların saptanması", "ürün geliştirilmesi", "ürün yönetimi" gibi temel YÜB süreçleri tartışılmakla birlikte Aurora kullanılarak hayata geçirilen projelerden elde edilen tecrübeler ve örnekler de aktarılmaktadır.

Abstract

In this paper, an in-house Software Product Line (SPL), so-called "Aurora", is introduced as a platform independent multi-tier Web development environment including the core infrastructure based on Rich Internet Applications (RIA) and Enterprise Internet Applications (EIA) models. Besides Aurora provides a complete roadmap to enterprise scale Web-based applications, it also embodies the SPL within the context of software process management methodology, WYSIWYG (What You See Is What You Get) design and development environments, software lifecycle management techniques and quality management tools. Essential SPL activities – core asset development, product development, and product management – have been also discussed in addition to the brief description of real life cases implemented on Aurora.

1. Giriş

Web temelli yaklaşımların tüm dünya tarafından benimsenmesinin dizginlenemeyişi özellikle kurumsal ölçekli yazılım geliştirme süreçlerini zorlamaya başlamaktadır. Web temelli yaklaşımlar; zenginleştirilmiş sunum teknikleri (HTML, RIA ve PowerUser kavramı), farklı erişim kanalları, küçümsenemeyecek oranda bütünleşik gereksinimler ve birçok farklı geliştirme ortamının aynı anda kullanılmak zorunda olması gibi birçok açılımı da beraberinde getirmektedir. Bunun yanı sıra, zaman içinde yazılım süreçlerinin Nesneye Yönelik Geliştirme'den Bileşen Temelli Geliştirme'ye, Servis Temelli Mimari'ye ve en sonunda Yazılım Üretim/Ürün Bandı (YÜB) yaklaşımına dönüşmekte olduğu da yadsınamaz bir gerçektir.

Aurora ve beraberinde gelen yazılım süreç yönetimi yaklaşımı, yazılım mühendisliğinde yaşanan bu gelişmelere paralel olarak kurumsal bilgi sistemi gereksinimleri üzerine inşa edilmiştir. Bu nedenle Aurora temelde RIA ve EIA modellerini öngören, platform bağımsız bir Web uygulama mimarisi

sunmakla birlikte etkin ve yeniden kullanılabilir ölçütlerde uygulama geliştirebilmek için de sistematik bir yaklaşım getirmektedir. Aurora, özellikle geliştirme döneminin tamamlanmasından hemen sonra büyük bir genişleme sürecine girmiş ve farklı müşteri istekleri sonucunda çekirdek bileşenler, etkin yazılım geliştirme ve yönetim süreçleri gibi kalıcı özellikleri bünyesine katmıştır.

Bildiri içerisinde bir sonraki bölüm YÜB yaklaşımını özetlemekte ve yazılımda yeniden kullanım yöntemlerinin YÜB olarak değerlendirilebilmesi için gereken temel kriterleri tanımlamaktadır. Üçüncü bölümde benzer kaygılarla geliştirilen Aurora yaklaşımı özellikle mimari açıdan detaylı olarak anlatılmaktadır. Dördüncü bölümde ise Aurora yaklaşımının neden bir YÜB olduğu açıklanmaktadır. Son olarak, Aurora üzerinde gerçekleştirilmiş projeler ve bunlardan edinilen tecrübelerin değerlendirilmesi ile bildiri sonuçlandırılmaktadır.

2. Yazılım Üretim/Ürün Bantları

Bir *Yazılım Üretim/Ürün Bandı*; özel bir hedefe yönelik gereksinimleri sağlayan, paylaşılabilir ve yönetilebilir olmanın yanı sıra önceden tarif edilmiş yöntemlerle genel olarak *temel varlıklar (core assets)* kümesinden üretilebilen yazılım sistemleri bütünüdür [4]. Bir ürün bandının kapsamı; ürün bandını meydana getiren ürünlerin veya bir ürün bandının neleri üretebileceğini tarifleyen kabiliyetlerin tanımlanmasından ibarettir. Bu kapsam bünyesinde gereksinimler, tasarımlar, test durumları ve buna benzer diğer yazılım bileşenleri gibi temel varlıkların düzenli olarak yeniden kullanılabilirliği yazılım geliştirme maliyetlerini çarpıcı bir şekilde düşürecektir.

Yazılım süreçlerinde sistematik olarak yeniden kullanılabilirliği hedefleyen YÜB yaklaşımı bilgi sistemlerini modellemek için iki temel kavramdan yararlanmaktadır: *genelleme* ve *çeşitlilik*. Bunu hedeflerken de genellemeye odaklanmakta, çeşitliliği ise etkin bir şekilde yöneterek, benzer yazılım sistemlerine yönelik diğer ürün bantlarının ortaya konması ve sürdürülebilirliği için gerekli olan zamanı, çabayı, maliyeti ve karmaşıklığı en aza indirgemektedir. Böylelikle temel varlıklar ve genellemenin etkin yönetimi saysinde mevcut projelerdeki performans yetersizliklerinin ortadan kaldırılması, maliyetlerin en aza indirgenmesi, yönetilebilir ve sürdürülebilir ürün çeşitliliğinin sağlanması ile farklı müşteri ve pazar beklentilerine hızlı cevap verebilme gereksinimleri gibi karmaşık problemleri de daha kolay ele alabilmektedir.

Bu tür karmaşık problemlerin daha kolay ve etkin bir şekilde çözülebilmesindeki temel etmen ise belli bir ortak yapıya sahip olmakla birlikte bilinen ve yönetilebilir çeşitlilikte olan ürünlerin ortaya konabilmesi için gereken yazılım bileşenlerinin tanımlanması ve yeniden kullanılabilirliğine yönelik bir *ürün bandı mimarisini* şekillendirebilmesidir [25]. Dolayısı ile YÜB yaklaşımı şu tür tipik yeniden kullanılabilirlik yaklaşımlarından farklılaşmaktadır: kopyala-bırak şeklindeki yeniden kullanım, yeniden kullanım ile tek bir bilgi sistemi geliştirme, salt bileşen temelli uygulama geliştirme, konfigürasyon parametreleri ile yeniden şekillendirilebilen mimari kullanım, uygulama çatıları (frameworks) ile yazılım geliştirme, bir ürünün yeni sürümlerini otomatik olarak ortaya koyma ve birtakım standardlar kümesine dayanma.

Bir ürün/üretim bandının işletilebilmesi, temel varlıkların geliştirilmesi ile bu temel varlıkları kullanarak ürün geliştirilmesi olarak bilinen ve her ikisi de hem teknik hem de idari yönetim kademeleri tarafından sahiplenmesi gereken süreçler içermektedir. Buna yönelik olarak YÜB yaklaşımı üç temel işlev öngörmektedir [14,20]:

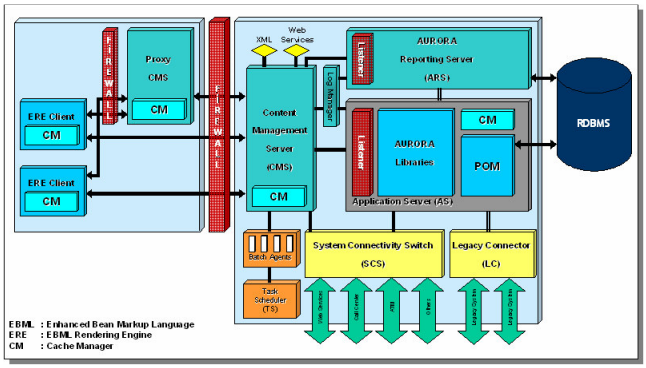
- *Temel Varlıkların Geliştirilmesi* üretim bandına yönelik temel varlıkların saklandığı bilgi bankasının güncel tutulması için süregelen bir işlevdir. Bu işlevin beklenen çıktıları ise ürün ailesinde kullnılan *temel varlıklar listesi* ile bu temel varlıkların bir ürüne dönüştürülmesi için ne şekilde kullanılması / uyarlanması gerektiğini açıklayan bir *üretim planıdır*.

- *Ürün Geliştirilmesi* ürün planında ifade edildiği şekliyle temel varlıklardan yararlanarak ürünün kendisinin geliştirilmesine yönelik tanımlı bir mühendislik sürecidir.
- *Ürün Yönetimi* ise eğer tam anlamıyla gerçekleştirilemezse ürün bandının başarısız olmasına kadar gidebilecek öneme sahip olan teknik ve idari yönetim aktivitelerinin bütünüdür.

Temel varlıkların geliştirilmesi ile ürün geliştirilmesi süreçleri birbirlerini karşılıklı olarak tetikleyebilir: temel varlıklardan yeni ürünler ortaya konabildiği gibi mevcut ürünler incelenerek temel varlıklara da ulaşılabilir. Bu nedenle temel varlıklar ile ürünlerin birbirlerine geri beslemede bulunabildikleri sürekli bir döngü mevcuttur. Yeni ürünler geliştirildikçe temel varlıklar da güncellenirler. Bu arada yeni temel varlıklar saptanır ve bilgi bankasına aktarılır, dolayısıyla bu sürecin sonuçları temel varlıkların geliştirilmesi işlevini doğal olarak beslemektedir. Ayrıca, temel varlıkların katma değeri, bu değerlerden meydana gelen ürünler üzerinden tespit edilir. Sonuçta temel varlıklar, daha sonra geliştirilecek potansiyel ürünler gözönüne alınarak genelleştirilir.

3. Aurora Yazılım Üretim Bandına Genel Bakış

Aurora [2] kurumsal Web uygulamalarının RIA ve EIA modelleri esas alınarak geliştirilebilmesi için uygun yazılım süreç yönetimi, WYSIWYG tasarım ve geliştirme ortamları, yazılım yaşam döngüsü yönetim teknikleri ve araçları da içeren platform bağımsız bir YÜB'dır. Yazılım yaşam döngüsündeki tüm gereksiz kodlama işlevlerini kaldırarak bunu çerçeve yaklaşıma yönlendirebilmek için Web uygulamalarının geliştirilmesi sırasında her katmanda ihtiyaç duyulabilen yöntem ve araçları sunmaktadır. Çok katmanlı mimari Şekil 1'de gösterilmiştir.



Şekil 1. Aurora Çok Katmanlı Mimari Altyapısı

RIA yaklaşımı etkin ve hızlı cevap verebilen görsel tasarımlar için istemci tarafındaki tüm kaynakların seferber edilmesini öngörmektedir [7,15,13]. Bu nedenle de *masaüstü yeteneklerinin*, *Web*'in ve *iletişimin* en üst düzeyde bütünleşmesini hedeflemektedir [5]. Aurora sunum katmanı, istemcide *Sıfır Kodlama ve Yükleme Modeli*'ni öngören bir yaklaşımla özellikle ince istemcilerin (thin clients) avantajlarını da ortadan kaldırmayacak şekilde görsellik anlamında zengin Web uygulamalarını geliştirebilmek için tasarlanmıştır.

Aurora sunum katmanı *User Interface Markup Language (UIML)* yaklaşımından esinlenmiştir. UIML kullanıcı arayüzlerinin görselleştirme ortamından ve teknolojilerinden bağımsız olarak XML formatında ifade edilebilmesini sağlayan bir standard kümesidir [16,22]. Çok katmanlı Web mimarileri açısından bakıldığında UIML sadece sunum katmanını tariflemektedir. Aurora UIML kavramını benimsemiş, yüksek etkileşimli Web sunum katmanı için yeniden düzenlemiş, JavaBean ve ActiveX teknolojileri ile güçlendirmiş, gelişmiş görselleştirme ögeleri ile zenginleştirmiş ve sonuçta oluşan "markup" lisanına *Enhanced Bean Markup Language (EBML)* adını vermiştir.

Gelinen noktada EBML; yeniden kullanılabilen ekran bölgelerinin tanımlanabildiği, aritmetik ve mantık kurallarının ifade edilebildiği ve çalıştırılabildiği, istemcide ve sunucularda servislerin çağrılabildiği, yapısal parçacıkların ayrı ayrı versiyonlandığı ve önbelleklendiği, parametrik bilgilerin detaylı sorgulanmasına yönelik ayrı görsel öğelerin yönetilebildiği, istemci tarafında tek satır kod yazmadan aynı içeriğin farklı lisanlarda sunulabildiği bir yetkinliğe ulaşmıştır.

EBML sunum ve içerik yönetimi katmanları arasında kavramsal bir köprü görevi görmektedir. EBML ile ifade edilen tanımların görüntülenmesi ve görsel etkileşimin sağlanması amacıyla istemci tarafında; platform bağımsız istemciler için genel bir Java applet ve Windows ortamında azami performans ve etkin yükleme yaklaşımı için de C++ ATL temelli iki farklı bileşen geliştirilmiş ve genel olarak da buna *EBML Rendering Engine (ERE)* adı verilmiştir.

Kurumsal Web uygulamaları istemciler ve içerik sunucuları arasında oluşan büyük bir veri (ekran tanımları, rapor tanımları, parametrik sorgular, markup özellikleri ve işlevsel veri gibi) trafiğine sahiptir. Dolayısıyla toplam uygulama performansının artırılması için Web uygulamalarında altın kural olan önbellekleme yeteneklerinin en üst seviyeye getirilmesi kaçınılmazdır. Bu açıdan bakıldığında EBML, HTML ile karşılaştırılamayacak yetkinlikte yapısal tanım parçacıklarının işlevsel verilerden ayrıştırıldığı tasarımsal ve işletimsel özelliklere sahiptir. Bu kadar yüksek kalibrasyona sahip bir veri bölümlenmesi sayesinde Web uygulamalarının ihtiyaç duyduğu etkin önbellekleme yaklaşımları ve konfigürasyonlarının kullanımı kolaylaşmıştır. Ayrıca, HTML proxy sunucularına benzer önbellekleme yeteneklerine sahip, EBML parçacıklarının sunulabildiği EBML proxy sunucuları da geliştirilmiş ve parametrik olarak kullanılabilir hale getirilmiştir.

Kurumsal uygulamalar aynı zamanda etkin raporlama ve listeleme kabiliyetlerini kaçınılmaz hale getirmiştir. Rapor içeriğinin oluşturulması, raporun farklı formatlarda (PDF, Doc, XML, HTML, etc.) gösterimi ve baskı yönetimi de RIA yaklaşımlarında çok büyük önem taşımaktadır. Aurora'da etkin raporlama için *Universal Reporting Markup Language (URML)* adıyla ayrı bir "markup" lisanı tasarlanmış ve raporların görselleştirilmesinden nokta-vuruşlu yazıcılarda yazdırılabilmesine kadar uzayan detaylı gereksinimler bu şekilde karşılanabilmiştir.

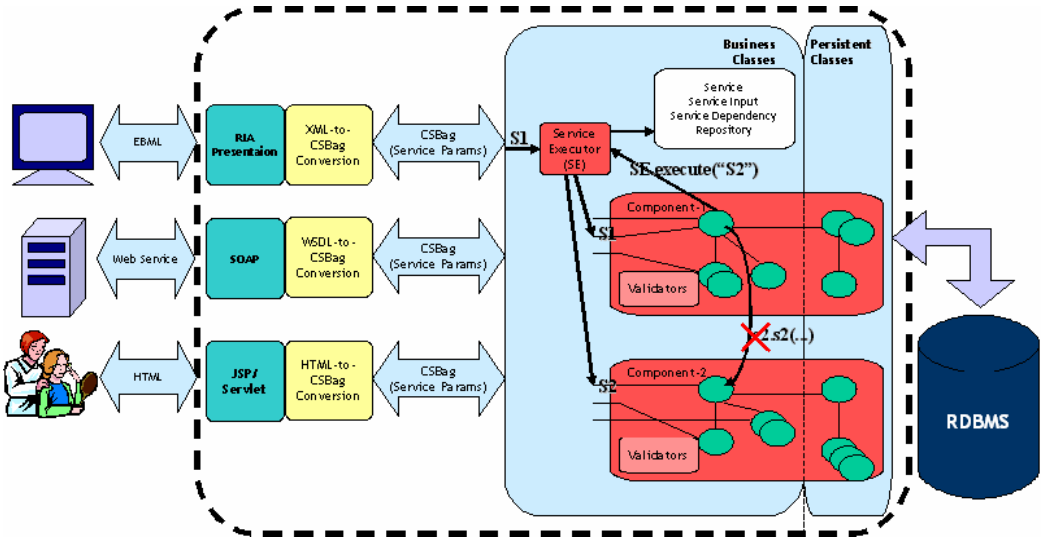
İçerik yönetimi katmanı; sunum katmanından uygulama katmanına uzanan ve anlık sogulamalar ile raporlama taleplerine de karşılık verebilen bir köprü görevi görmektedir. Aurora *İçerik Yönetim Sunucusu (Content Management Server - CMS)* tarafından desteklenen ve aşağıda belirtilen prensipler, aynı zamanda Aurora YÜB yaklaşımındaki temel varlıkları da işaretlemektedir:

- CMS altyapısı, etkin önbellekleme yapmak ve uygulama bakım maliyetlerini indirgemek amacıyla yapısal parçacıkların işlevsel veriden ayrıştırılması prensibine dayanmaktadır.
- CMS *kişisel belirleme (authentication)* ve *yetkilendirme (authorization)* için genel bir rol-tabanlı yaklaşım sunmakla birlikte bu yaklaşımın projelerin özel isterlerine uygun şekilde yeniden tanımlanabilmesine de olanak vermektedir.
- CMS *oturum yönetimi (session management)* için oturum bilgilerinin (session information) ve ilgili bağlamların (session context) sunucu tarafında tutulan genel bir model (server-side persistent session store) ile yönetilebilmesini sağlamaktadır. Bununla birlikte yine CMS'in sağlamış olduğu programlama arayüzleri ile bu genel modelin prensipleri gerekli olduğu noktalarda özel isteklere göre şekillendirilebilmektedir.
- CMS, ERE bileşeni ile farklı kriptolama ve sıkıştırma yöntemleri üzerinden konuşabilmektedir.
- İstemcide önbelleklenebilecek tüm yapılar (EBML, URML, parametrik bilgiler vb.) CMS'in kontrolünde ürünlerin değişen taleplerine göre yönetilebilmektedir.
- XML temelli sunum mekanizması çoklu dil ve kişiselleştirme desteği vermektedir.
- Gelişmiş görsel öğelerin kullanımını sunarak belli kayıtların anlık sorgulanması, sonuçlarının önbeleklenmesi ve sayfa sayfa (streaming desteği) gösterilmesi gibi Web için kritik olan özellikleri uygulama geliştirmeden bağımsız olarak sağlamaktadır.
- CMS rapor içeriğinin hazırlanması ve sunulması için de uygulama geliştirmeden bağımsız etkin yapılar içermektedir. Bu amaçla raporlama taleplerine yönelik servislerin çalıştırılması, sonuçların önbelleklenmesi ve asenkron iletişim kanalları ile istemciye ulaştırılması işlevleri yine programcıdan bağımsız, altyapısal olarak sağlanmaktadır.

- Aurora ile tüm orta katman servisleri ek bir uygulama geliştirme maliyeti gerektirmeden Web servisi olarak dış dünyaya sunulabilmektedir. Web servislerine yönelik WSDL [9] tanımları zaten *Servis Temelli Mimariye (Service-Oriented Architecture - SOA)* [6,11] sahip olan Aurora servis altyapısının doğal bir uzantısıdır ve orta katman servislerinin Web servisi olarak sunulması sadece bir tanımlama işlevidir.

Aurora uygulama modeli iş tanımlarının içerik sunumu ve veri tabanı süreçlerinden tamamen ayrıştırılmasını öngörmektedir. Bu yaklaşım ve bileşen temelli mimari ile Aurora'da iş tanımları çok uygun ölçekte ve tamamen birbirinden bağımsız bileşenlere bölümlendirilebilmekte, böylelikle bileşenler arası etkileşim (coupling) en aza indirgenirken bileşen içi tutarlılık (cohesion) ise en üst seviyeye çekilebilmektedir [12,19].

Şekil 2'de iş tanımı bileşenlerinin detaylı yapılandırılmasını gösteren Aurora orta katma mimarisi SOA uyumludur. Bu şekil temelde Aurora orta katman bileşenlerinin çalışma prensiplerini üst seviyede resmetmektedir. Aurora'da *Bileşen Nesne Modeli (Component Object Model)* iki ana bölümde ele alınmaktadır: *İş Tanımları Nesne Modeli (Business Object Model – BOM)* ve *Saklama Nesne Modeli (Persistent Object Model – POM)*. Birbirinden tamamen ayrılmış olan bu iki model yine Şekil 2'de *Servis Sağlayıcı (Service Executor – SE)* olarak belirtilen bir arayüz (facade) ile yönetilmekte ve böylelikle bu iki modelin birbirinden bağımsız olarak geliştirilmesi sağlanmaktadır.



Şekil 2. Aurora Servis Temelli Mimari

Bileşenler *servis* olarak adlandırılan arayüzler üzerinden etkileşimde bulunurlar ve bu arayüzler bileşenlerin dış dünyaya ve Aurora çatısına açıldıkları bir tür sözleşmedir. Böyle bir sözleşme bir bileşenin yönetmek zorunda olduğu *genel kısıtları (invariants)*, bileşen kullanıcılarının ihtiyaçlarını karşılayan *ön koşulları (pre-conditions)* ve bileşenlerin sonuçları algılayabileceği *art koşulları (post-conditions)* içermektedir. Şekil 2'de gösterilen *doğrulayıcılar (validators)* ise bu ön koşulların kodlanmış görünümleridir. Aurora servislerinin adları, girdi ve çıktı parametreleri merkezi bir bilgi bankasında (service repository) tutulmaktadır. Aurora servislerinin çalıştırılması için karmaşık EJB çatıları yerine *Basit Java Sınıfları (Plain Old Java Objects – POJO)* yaklaşımı yeterlidir.

Şekil 2'den de görüleceği üzere servis temelli bir sistem uygun adaptörlerin kullanılması sonucu EBML, XML, HTML ve WSDL seçeneklerinden herhangi birisi ile sunum katmanına içerik sağlayabilmektedir. Aurora SOA yaklaşımını Web servislerinden ibaret görmek yerine onun doğal bir uzantısı olarak modellemektedir [18].

Yakın zamanda geliştirilen bir başka yaklaşım sayesinde Aurora uygulama katmanı daha etkin bir bölümleme ile *Temel İş Mantığı (Core Business Logic – CBL)* ve *İş Kuralları (Business Rules - BR)* olarak modellenmiştir [1]. Bu model *(RUle Based Model for Basic Aspects – RUMBA)*; önerilen özgün *Reflective Aspect* ve *Reflective Rule* şablonları sayesinde kural temelli bir uygulama katmanını YÜB bünyesine bütünleştirmiştir. Reflective Aspect ve Reflective Rule şablonları; bytecode uyarlama modeli kullanarak *kod üretimine (generative)* dayanan ve temel iş tanımlarının *basit görünümler (basic aspects)* olarak tasarlandığı mimari şablonlardır. Bu şablonlar sayesinde *Uyarlanabilir Nesne Modeli (Adaptive Object Model)* [23,24] kullanılarak değişen iş kurallarının görsel ekranlar aracılığıyla *kural (rule)*, *kural kümeleri (rule-sets)* ve *olgular (facts)* olarak tanımlanabildiği dinamik ve kodlama gereksiniminin en aza indirgendiği bir model geliştirilmiştir. Böylelikle dinamik ürün yapılarının BT bölümleri dışında gerçekten işin sahipleri tarafından tanımlanabilmesi hedeflenmiş, bu tanımlar yapılırken YÜB'deki genellemelerin basit görünümler, çeşitliliklerin ise iş kuralları şeklinde modellenebilmesi sağlanmıştır.

Son olarak Aurora veri katmanı; veri ve servis kaynaklarının uygulama katmanına uygun bir şekilde bütünleştirilebilmesine yönelik olarak modellenmiştir. Aurora'da veri tabanı servisleri platformun sağladığı saydam bir *O2R eşleşme (Object to Relational mapping)* arayüzü üzerinden verilmektedir. *POM Manager (Persistent Object Model Manager)* olarak bilinen bu arayüz kendi bünyesinde bir eşleşme yapısı içerdiği gibi endüstriyel olarak O2R eşleşmesi için kabul gören Hibernate [8] türü yaklaşımların kullanılmasına da izin vermektedir.

4. Aurora'yı Bir Yazılım Bandı Olarak Nitelendirmek

Bu bölümde Aurora yaklaşımı YÜB temel aktiviteleri [14] kapsamında tartışılmaktadır. Temel varlıkların geliştirilmesi, ürün geliştirilmesi ile teknik/idari yönetim aktiviteleri ve bunların Aurora ile nasıl gerçekleştirildiği açıklanmakta, yeri geldikçe YÜB ölçüm kriterleri [25] ile tartışma zenginleştirilmektedir. Bu ölçüm kriterleri iki ana gruba ayrılmaktadır: *YÜB yönetimi ölçütleri* ve *temel varlık geliştirme yönetimi ölçütleri*. YÜB yönetim ölçütleri, YÜB'ün genel performans ve uygulanabilirliğini ölçmek için tasarlanmıştır. Bunun yanı sıra temel varlık geliştirme yönetimi ölçütleri ise varlıkların yönetimi, projelerin/ürünlerin temel varlık üretimine katkıları ve temel varlıkların ürün geliştirme projeleri açısından değerlerini ölçmek için kullanılmaktadır.

4.1. Varlık Geliştirme Açılımı

İkinci bölümde belirtildiği gibi temel varlık geliştirme ile ürün geliştirme süreçleri arasında kuvvetli bir geri besleme döngüsü vardır. Aurora varlıkları başlangıçta bir temel bankacılık uygulaması içinde geliştirilmiştir. Temel varlıklar kurumsal bankacılık sisteminden özümsenmiş, diğer sistemleri de mümkün kılacak şekilde genişletilmiş ve Aurora YÜB kapsamı oluşturulmuştur.

Aurora YÜB kapsamı; zengin istemci arayüzü sağlayan platform bağımsız bir Web uygulama mimarisi, istemci gücünün en üst düzeyde kullanımının sağlanması, ince istemci yaklaşımı, istemci tarafında kural işleme motoru, istemci tarafında asenkron rapor görselleştirme, verimli önbellekleme mekanizmaları, uyarlanabilir güvenlik modelleri, etkili entegrasyon yetenekleri, orta katmanda iş tanımlarının SOA ile geliştirilmesi, anlık Web Servisi üretimi ve entegrasyonu, doğru ölçeklendirme ve platform bağımsızlığı olarak tanımlanmaktadır.

RIA ve EIA modellerini mümkün kılan SOA bazlı mimari yaklaşım Aurora YÜB kapsamında en temel varlıktır. Mimari, Aurora YÜB ile üretilebilecek ürünlerin ve temel varlıkların yapısını belirleyen birleştirici bir faktördür [25]. Diğer önemli varlıklar arasında kullanıcı doğrulama ve yetkilendirme modelleri, seans yönetimi, bileşen temelli geliştirme modeli, genişleyebilir işlem

modeli, POJO bazlı iş nesneleri, O2R eşleşme modeli, PKI, akıllı kartlar ve uyarlanabilir şifreleme modelleri içeren güvenlik altyapısı, geliştirme bilgi bankası bulunmaktadır.

Aurora mimari yaklaşımı ile sağlanan yazılım kalite öğeleri de temel varlık kümesi içinde yeralmaktadır. Bu kalite öğeleri arasında yatay ve dikey ölçeklenebilirlik, etkin Web anahtarlama ve yük dengeleme, doğrusala yakın ölçeklenebilirlik (%98.8), 7x24 çalışabilirlik ve etkin alternatif dağıtım kanalları entegrasyonu sayılabilir.

4.2. Ürün Geliştirme Açılımı

Tipik bir YÜB yaklaşımında ürün geliştirme aslında bir birleştirme sürecidir: temel varlıklarla başlar ve ürünün birleştirilmesi ile son bulur. Bu açıdan bakıldığında Aurora YÜB bu tanıma uymanın yanı sıra ürün geliştirmeye metodolojik bir açılım da getirmektedir. Ürün geliştirme aktiviteleri şu sorunun yöneltilmesiyle başlar: bu ürün Aurora YÜB kapsamında mıdır? Bu soruyu cevaplamak için öncelikle ürün kapsamı belirlenir ve ardınan Aurora temel varlıkları resmin içine yerleştirilmeye çalışılır. Sonuçta kapsamın karşılanması cevabı da getirecektir: eşleşme ne kadar büyükse ürün de o kadar Aurora YÜB kapsamındadır denilebilmektedir.

Ürün hakkında böyle bir pozitif belirlemenin ardından ürün geliştirme süreci Aurora geliştirme metodolojisine uygulanır. Bu metodoloji süregelen (iterative) ve bileşen temelli bir yaşam döngüsüdür. Mevcut temel varlıklar farklılık analizi için eldeki ilk bileşenlerdir. Buna göre temel varlıklar istenilen ürüne yönelik olarak uyarlanıp, iyileştirilir. Bazen yeni temel varlıklara da ihtiyaç duyulur ve bu durumda onlar için de yeni bir *alan analizi (domain analysis)* başlatılır.

Analiz yaklaşımı nesneye yönelik tekniklerden alınmış ve çok ince bir UML gösterimine uyarlanmıştır. Tasarım süreci bileşenler, servisler ve uygulama bölümlenmesi etrafında devam etmektedir. Bileşenler belirlenir, *servis* diye nitelendirilen bileşen arayüzleri tanımlanır ve bileşen seviyesinde uyarlanmış bir *sıralama diyagramı (sequence diagram)* oluşturulur. Bu arada Aurora'nın ERE ve EDS yetenekleri kullanılarak etkin bir prototipleme süreci de başlatılır.

Kodlama sırasında süregelen yaklaşım devam ettirilir, servis modeli ve bu modelin arkasındaki bileşenler sürekli sisteme dahil edilirler (continuous integration). Bu arada gözden geçirmeler ve kod kontrolleri ile temel varlıkların çerçevesi yeniden uyarlanır, gerekiyorsa yeni temel varlıklar ortaya konur.

Konfigürasyon ve sürüm yönetimi Aurora yaklaşımının vazgeçilmez bir parçasıdır. Bu amaçla CVS temelli bir bilgi bankasından yararlanılır. Veri sözlüğü oluşturulur ve bileşenler, servisler, sınıflar, metotlar, ekranlar, ekransal bölgeler ve raporlar yaşam döngüsü boyunca bu sözlüğe tanımsal olarak konulur. Etki (impact) ve bağımlılık (dependency) analizi çok önemli bir yer tutmaktadır ve veri sözlüğü ile bu iki temel işlev kolaylıkla yönetilebilmektedir.

Unutulmaması gereken önemli süreçlerden birisi de test yönetimidir. Aurora test yönetimi temel varlıklardan hareketle ve onların genişletilmesiyle bileşen temelli olarak yürütülmektedir. Aurora test süreci test ekibinin tanımlanmasından; ekip üyelerinin rol ve sorumluluklarının tarifine, test senaryolarının yönetimine, testlerin yapılış tarihçelerine kadar aktiviteleri öngörmektedir. Test süreci bünyesinde entegrasyon, yük, stres ile regrasyon testlerini de içeren ve mekanik olarak tekrarlanabilir aktiviteler vardır. Başlı başına test yönetim süreci bile Aurora YÜB'nın en önemli temel varlıklarından birisidir.

4.3. Yönetimsel Açılımlar

Yönetimsel açıdan bakıldığında Aurora YÜB; üretim maliyetinin indirgenmesini, üretkenliğin artırılmasını, çıktıların zamanında oluşturulmasını, yaşam döngüsü işlevlerinin layıkıyla kontrolünü,

temel varlıklar ve ürünlerdeki hata sayısının azaltılmasını, şirket misyonuna odaklanılmasını, mimari uyumluluğu, süreç uyumluluğunu, toplam kaliteyi, pazar ve müşteri memnuniyetini amaçlamaktadır. Buna yönelik yürütülen yönetim aktiviteleri; konfigürasyon yönetimi, süreç yönetimi gibi teknik yönleri içermekle birlikte proje kurulumu, proje organizasyonu, kaynak ve zaman planlaması gibi idari yönleri de öngörmektedir.

"Kaydedilemeyen hiçbir şey ölçülemez, ölçülemeyen hiçbir şey yönetilemez, yönetilemeyen hiçbir şey de mühendislik değildir" mottosu Aurora yönetim yaklaşımının temelini oluşturmaktadır. Bu nedenle tüm yazılım geliştirme süreci tekrarlanabilir ve yönetilebilir bir şekilde tanımlanmaya çalışılmıştır. Metodoloji; temel bir yazılım süreç yönetimi bilgi bankası etrafında şekillendirilmiştir. Bu bilgi bankası geliştirme sürecini tariflemekte, yol göstermekte ve ekip üyelerinin rol ve sorumluluklarını yönlendirmektedir. Web temelli olan bilgi bankasına erişim, uygulama geliştirme sürecine yardımcı olmaktan çok etkin bir ölçme ve değerlendirme yaklaşımı için geliştirme sürecinin vazgeçilmez bir parçası haline getirilmektedir.

Yazılım süreç yönetimi bilgi bankası sonuçta kalite yönetimi için de tek başvuru kaynağıdır. Ürün bandının etkinliğini artırabilmek için geliştirilen *Lighthouse* [21]; Web temelli bir kalite ve iletişim yönetim sistemidir. Lighthouse ile kağıt ortamındaki iletişimin azaltılması, bürokrasinin ortadan kaldırılması, uygun süreçteki kişilerin etkin iletişimi için altyapıların oluşturulması, temel varlıklara yönelik yenilik ve iyileştirilmelerin ekip üyelerine duyurulmasının sağlanması ve sonuçta süreçlerin iyileştirilmesi adına ölçme ve değerlendirmelerin yapılabilmesi için metrik bilgilerin toplanması hedeflenmektedir. Lighthouse yaklaşımı ISO 15504 (SPICE) [10] modeli ile birebir uyumlu kalite değerlendirme süreçlerinin kullanılabilmesini mümkün kılmaktadır.

5. Aurora Uygulamaları

Bu bölümde Aurora YÜB kullanılarak geliştirilen ürün ve projelerden bazılarına kısaca değinilmektedir. Bu projelerin tipik özellikleri; coğrafik olarak dağıtık 1000+ kullanıcısı olan, karmaşık ve çok çeşitli iş tanımları içeren, zor modellenen geniş veri hacmine sahip, Web temelli kullanımın avantajlarını zengin istemci avantajlarıyla birleştiren, 7x24 kullanıma açık, yoğun işlem hacmine sahip şeklinde sıralanabilir.

- *Temel Bankacılık Sistemi* yaklaşık iki yılda ve 3600 adam aylık bir emek ile tamamlanmıştır. Java temelli bu Temel Bankacılık Sistemi halen iki yıldır üretim ortamında sorunsuz çalışmaktadır. Bu uygulamada 20'den fazla dış sistemle XML/Web Servisi temelli adaptörler üzerinden entegrasyon sağlanmıştır. Bankacılık modeli ürün ve işlem kavramlarına odaklanmış, Aurora YÜB temel varlıklar kütüphanelerine müşteri ilişki yönetimi, ürün bazlı sistemler, işlem merkezli yaklaşım, dinamik organizasyon yönetimi, güçlü bir rol bazlı yetki, onay ve delegasyon mekanizması, kanıtlanmış bir XML-bazlı RIA modeli ve bunun üzerinde çalışma felsefesi eklenmiştir.
- *Merkezi Kayıt Kuruluşu, Kaydi Sistem* bir buçuk yıllık ve yaklaşık 400 adam-ay'lık bir emek ile geliştirilmiş, 2005 yılı ilk çeyreğinde ise devreye alınmıştır. Sistem Aurora üzerinde geliştirilmiş ve çeşitli finansal sistemlerle Web Servisi bazlı entegrasyon sağlanmıştır. Finansal işlemler asenkron olarak başlatılırken sonuçların mesaj bazlı bildirimlerle dönülmesi üzerine çalışan bir mekanizma kurulmuştur. Bu sistem sermaye piyasalarına Türkiye'de ilk kez kullanılan Straight Through Processing (STP) [17] açılımını getirmektedir. Yüksek işlem hacmi ve yoğun kullanımın tam olarak kestirilemediği yapısıyla sistemin temel başarı faktörleri devamlılık, dayanıklılık ve güvenirlik olarak tespit edilmektedir. Sistemde, ileri düzey güvenlik politikaları PKI altyapısı kullanılarak yerleştirilirken, Smartcard ve sayısal imzanın kullanımı bu açıdan da sisteme Türkiye'de bir ilk olma özelliği getirmektedir.

- *Sigortacılık Altyapısı* çok yeni bir proje olmakla birlikte RUMBA [1] katmanının kullanıldığı ilk örnektir. Sigortacılık alanında dinamik ve sürekli değişen iş tanımları, karmaşık iş kuralları ve işlevsel kısıtlar bu tür uygulamaların geliştirilmesini ve bakımını oldukça zorlaştırmaktadır. Benzer sorunlara çözüm sağlamak için geliştirilen RUMBA yaklaşımı, sigortacılık alanındaki karmaşıklığı modelleyecek ve yönetecek seçim olmuştur. Bu yaklaşımda temel iş tanımları basit görünümler ile geliştirilirken, iş kuralları da temel kural ve kural kümeleri üzerinden tanımlanmaktadır.

Aurora YÜB üzerinde uygulama geliştirme yaklaşımı BT organizasyonlarını da dramatik olarak etkilemektedir. Aurora YÜB bazlı uygulama geliştirme iş akışına ek olarak özelleşmiş gereksinim yönetimi, değişim yönetimi, konfigürasyon ve sürüm yönetimi, test yönetimi ve kalite yönetimi ekiplerinin işlevlerini yeniden tanımlamaktadır. Diğer yandan hala uygulama geliştiren ekipler kendi uzmanlık alanlarında, bu kez daha etkin olarak çalışmaya devam etmektedir.

6. Sonuç

YÜB kavramı yazılım mühendisliğinde sıkça tartışılan konular arasında yeralmaktadır. Aurora, bu gelişmeler ışığında tasarlanmış olan bir YÜB'dır. Aurora YÜB; çok katmanlı ve Web temelli çatıya ek olarak misyon kritik ve yüksek devamlılık gerektiren kurumsal sistemler için temel varlıkları da tanımlamaktadır. Aurora mimarisinde temel varlıklara ve uygulama geliştirme yaşam döngüsüne mimari anlamda özel bir önem verilmektedir.

Bu bildiri özellikle Yazılım Üretim Bandı işlevlerini tariflemekte ve Aurora'nın bir YÜB olarak nitelendirilebilirliğini üç farklı açıdan ele almaktadır. Öncelikle temel varlıkların geliştirilmesi penceresinden bakıldığında Aurora kapsamı net olarak tanmlanmış ve temel varlıklar kalite belirteçlerini de içine alacak şekilde bu tanımlara dahil edilmiştir. Aurora'da ürün geliştirme mimariyle bütünleşik bir süreç olarak tanımlanmış ve tüm yazılım yaşam döngüsü işlevlerine her katman için bağlanmıştır. Son olarak, ürün bandının yönetimi ekip yapılandırmasını içine alacak şekilde tanımlanmış, projenin kurulum aşamasından; kaynak planlaması, zaman planlaması, süreç yönetimi ve hatta ölçümlemeye kadar gerekli kriterler ortaya konmuştur.

References

[1]. Altintas, N. Ilker & Cetin, Semih. Integrating a Software Product Line with Rule-Based Business Process Modeling (http://www.cs.com.tr/ar-ge/pdf/spl-bpm-fullpaper.pdf). VLDB-TEAA-2005 (http://www.teaa.formcharts.org/). (2005)

[2]. AURORA Software Product Line, Web Site, http://www.aurora.com.tr

[3]. AURORA Software Product Line, Blue Paper, http://www.cs.com.tr/pdf/aurora-bluepaper-en.pdf

[4]. Clements, P. & Northrop, L. Software Product Lines: Patterns and Practice. Reading, MA: Addison Wesley. (2001)

[5]. Duhl, J., The Business Impact of Rich Internet Applications. IDC White Paper. (2003)

[6]. Erl, Thomas. Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall. (2004)

[7]. Grosso, W., Laszlo: An Open Source Framework for Rich Internet Applications. (2005) Web Address: http://today.java.net/pub/a/today/2005/03/22/laszlo.html

[8]. Hibernate. http://www.hibernate.org/

[9]. Java Technology and Web Services. http://java.sun.com/webservices/index.jsp

[10]. ISO/IEC TR 15504:18 (E) Information Technology- Software Process Assessment.

[11]. Krafzig, D., Banke, K., & Slama, D. Enterprise SOA : Service-Oriented Architecture Best Practices (The Coad Series). Prentice Hall PTR. (2004)

[12]. Linthicum, D. S., Coupling Versus Cohesion: When to Leverage Services. Service-Oriented Architectures Web Address: http://www.ebizq.net/hot_topics/soa/features/4688.html (2004)

[13]. Mullet, K., The Essence of Effective Rich Internet Applications. Macromedia Experience Design. (2003)

[14]. Northrop, L. M., A Framework for Software Product Line Practice, v4.2. http://www.sei.cmu.edu/productlines/framework.html

[15]. O'Rourke, C., A Look at Rich Internet Applications. Oracle Magazine. (2004)

[16]. Phanouriou, Constantinos: UIML: A Device-Independent User Interface Markup Language. Ph.D. dissertation. http://scholar.lib.vt.edu/theses/available/etd-08122000-19510051/ (2000)

[17]. Samtani, G., Sadhwani, D.: Web Services and Straight Through Processing. Web Services Architect White Papers. http://www.webservicesarchitect.com/content/articles/samtani06.asp (2002)

[18]. Stevens, M., Service-Oriented Architecture, in Java Web Services Architecture, McGovern, J., Tyagi, S., Stevens, M., and Mathew, S., Morgan Kaufmann Publishers. (2003)

[19]. Stevens, W., Myers, G., and Constantine, L. Structured design. IBM System J. 13, 2. (1974) 115–139

[20]. SoftwareProductLines.com community web site. http://www.softwareproductlines.com/

[21]. Tufekci, O., LIGHTHOUSE: A Practical Quality Management Information System. Impact of Software Process on Quality Workshop, Izmir, 2004.

[22]. UIML, http://www.uiml.org/.

[23]. Yoder J. W., Balaguer F. & Johnson R. "Architecture and Design of Adaptive Object Models" Intriguing Technology Presentation at the 2001 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '01), ACM SIGPLAN Notices, ACM Press, (2001).

[24]. Yoder J. W., Balaguer, F. & Johnson, R. "Adaptive Object Models for Implementing Business Rules" Position Paper for Third Workshop on Best-Practices for Business Rules Design and Implementation, OOPSLA (2001)

[25]. Zubrow, D. & Chastek, G. Measures for Software Product Lines, Technical Note CMU/SEI-2003-TN-031. (2003)