EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

BAĞLI VERİ ORTAMINDA UYGULAMA GELİŞTİRMEK İÇİN BİR YÖNTEM TASARIMI VE LOJİSTİK KONTEYNER TAŞIMACILIĞI İÇİN UYGULANMASI

Pınar GÖÇEBE

Tez Danışmanı : Prof. Dr. Oğuz DİKENELLİ

Bilgisayar Mühendisliği Anabilim Dalı

Bilim Dalı Kodu : 619.01.00
Sunuş Tarihi : 07.05.2015

Bornova-İZMİR

2015

Pınar GÖÇEBE tarafından YÜKSEK LİSANS tezi olarak sunulan "**BAĞLI VERİ ORTAMINDA UYGULAMA GELİŞTİRMEK İÇİN BİR YÖNTEM TASARIMI VE LOJİSTİK KONTEYNER TAŞIMACILIĞI İÇİN UYGULANMASI**" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 07.05.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri: **İmza**

Jüri Başkanı **: Prof. Dr. Oğuz Dikenelli**

Raportör Üye **: Prof. Dr. N. YaseminTopaloğlu**

Üye **: Yrd. Doç. Dr. Tuğkan Tuğlular**

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi / Doktora Tezi olarak sunduğum "**BAĞLI VERİ ORTAMINDA UYGULAMA GELİŞTİRMEK İÇİN BİR YÖNTEM TASARIMI VE LOJİSTİK KONTEYNER TAŞIMACILIĞI İÇİN UYGULANMASI**" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

07 / 05 / 2015

İmzası

Pınar GÖÇEBE

ÖZET

BAĞLI VERİ ORTAMINDA UYGULAMA GELİŞTİRMEK İÇİN BİR YÖNTEM TASARIMI VE LOJİSTİK KONTEYNER TAŞIMACILIĞI İÇİN UYGULANMASI

GÖÇEBE, Pınar

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Prof. Dr. Oğuz DİKENELLİ
Nisan 2015, 56 sayfa

Bağlı veri ile veb, sayısı her geçen gün artan birbirine bağlı ve dağıtık veri kümelerinden oluşan büyük ve tek bir bilgi ağı haline gelmektedir. Bu bilgi ağı üzerinde istenilen bilgiyi ve bu bilginin ilişkili olduğu diğer bilgileri kullanıcıya sunan uygulamalar geliştirmek mümkündür. Bu tez çalışması kapsamında bağlı veri üzerinde uygulama geliştirmek için hangi aktivitelerin hangi sırayla gerçekleştirilmesi gerektiğini belirten bir yöntem önerilmiş ve bu yöntem lojistik konteyner taşımacılığı alanında bir uygulamada uygulanmıştır. Önerilen yöntem, çevik pratikler ve ilkeler çerçevesinde şekillenmiş, çevik mimari ve çevik analiz yaklaşımlarından esinlenerek uygulama geliştirme ve bağlı veri üretme süreçlerini harmanlamıştır.

Anahtar sözcükler: Bağlı Veri Geliştirme Yöntemi, Çevik Analiz, Çevik Mimari, Lojistik Konteyner Taşımacılığı

ABSTRACT

DESIGNING A METHODOLOGY TO DEVELOP APPLICATIONS IN LINKED DATA ENVIRONMENT AND IMPLEMENTATION OF IT FOR LOGISTICS CONTAINER TRANSPORTATION

GÖÇEBE, Pınar

MSc in Computer Eng.
Supervisor: Prof. Dr. Oğuz DİKENELLİ
April 2015, 56 pages

With the linked data, web is became into huge and single knowledge graph which consists of increasing number of interconnected and distributed datasets. It is possible to develop applications over this knowledge graph to provide required and related information to the users. In the context of this thesis, a methodology that defines activities and orders of these activities to develop applications over the linked data is proposed. Also, this methodoloy is implemented in a key study in the logistics container transportation domain.The proposed methodology is shaped around the agile practices and principles and it is inspried by the agile analytics and agile architecture approaches while collating the linked data and application development life cycles.

Keywords: Liked Data Development Methodology, Agile Analytics, Agile Architecture, Logistics Container Transportation

TEŞEKKÜR

Eğitimim ve tez çalışmam sürecinde bana verdiği destekten dolayı Prof. Dr. Oğuz Dikenelli ’ye, her durumda eğitimimi destekleyen ve manevi desteğini esirgemeyen aileme, dostlarıma, bana sağladığı çalışma ortamı için Galaksiya Bilişim Teknolojileri ve Danışmanlık Şirketi ’ne ve çalışma arkadaşlarım Erdem Eser Ekinci, Zehra Gül Çabuk, Tayfun Gökmen Halaç, Berkay Akdal ve Uğur Üntürk' e, SEAGENT araştırma grubu üyeleri Burak Yönyül, Enes Bulut ve Bahtiyar Erden ’e, yüksek lisans tez çalışmamı birlikte yürüttüğüm BİMAR Bilgi İşlem Hizmetleri' ne ve çalışanlarına, ayrıca yüksek lisans tezim boyunca bana burs sağlayan Bilim, Sanayi ve Teknoloji Bakanlığı, Bilim ve Teknoloji Genel Müdürlüğü' ne teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

BLOB	A Methodology to **B**ring Agility into **L**inked **O**pen Data Development for **B**usinesses
XP	Extreme Programming
UGK	Uygulama Gereksinim Kartı
İGB	İterasyon Gereksinim Belirleme
CDC	Changed Data Capture
RDF	Resource Description Framework
R2RML	RDB to RDF Mapping Language
SOM	Servis Odaklı Mimari
EDI	Electronic Data Interchange
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport
STEP	Standard for the Exchange of Product Model Data
AnsiX12	American National Standards Institute X12
XML	Extensible Markup Language
URI	Uniform Resource İdentifier
HTTP	Hyper-Text Transfer Protocol
UP	Unified Process

UML	Unified Modelling Language
SOM	Servis Odaklı Mimari

1 GİRİŞ

Bağlı veri, veb üzerindeki bilginin birbirine bağlanarak bütünleştirilmesini ifade etmektedir ve Tim Berners-Lee' ye göre Anlamsal Veb' in gerçekleşebilmesi için önce veb üzerindeki verilerin bağlı olması gereklidir (Bizer et al, 2009). Böylelikle veb üzerindeki bilgi sadece insanlar tarafından değil aynı zamanda makineler ve yazılım uygulamaları tarafından da anlamlandırılabilir, anlaşılabilir ve yorumlanabilir hale gelecektir. Bu amaçla, Tim Berners-Lee tarafından bağlı veri yayınlamak için 4 temel ilke belirlenmiştir (Bizer et al., 2009);

1. Her kaynak URI ile tanımlanmalıdır.
2. Hem insan hem de makinelerin kaynakları okuyabilmesi için HTTP URI' ler kullanılmalıdır.
3. Kaynaklara erişildiğinde RDF/XML gibi standartlara bağlı bir biçimde bilgi sunulabilmelidir.
4. Kaynaklar veb üzerindeki diğer ilgili kaynaklara bağlı olmalıdır.

Bu bağlı veri ilkeleri uygulanarak elde edilen bağlı veri bulutu 2011 yılındaki verilere göre 295 veri kümesi, 31 milyar üçlü ve 500 milyon bağ içermektedir (Jentzsch et al., 2011). Bağlı veri ile veb, sayısı her geçen gün artan birbirine bağlı ve dağıtık veri kümelerinden oluşan büyük ve tek bir bilgi ağı haline gelmektedir.

Lojistik gemi acentesi, liman/gemi operatörü, karayolu, demiryolu ve havayolu taşıma şirketleri gibi birçok farklı rolün bir konteyneri son noktaya ulaştırmak için birlikte çalıştığı karmaşık bir endüstridir. Bu şirketler tüm taşıma sürecini sorunsuz bir şekilde devam ettirebilmek için bir iletişim altyapısına ihtiyaç duyarlar. Bu iletişim altyapısı, tüm alt-taşımaları yönetebilmek ve gerektiğinde sürece yeni şirketleri kolayca ekleyebilmek veya çıkarabilmek amacıyla taşıma sürecine katılan şirketlerin bilgi sistemleri için bir bütünleştirme ortamı sağlamalıdır. Bilgi sistemlerinin bütünleştirilmesine ek olarak, sürecin etkili bir şekilde yönetilebilmesi için sürecin izlenmesi de kritiktir. Örneğin; kara taşımanın tamamlanması, limanda boşaltma işleminin başlaması gibi olaylar sürece katılan roller için çok kritiktir.

Son yıllarda, EDI tabanlı standartlar (EDIFACT, RosettaNet, STEP, AnsiX12), XML standartları ve Servis Odaklı Mimari (SOM) yaklaşımları lojistik sektörünün bütünleştirme problemlerini çözmek için kullanılmıştır (Nurmilaaskso,

2008; Harleman, 2012). Bu standartlar veri gösterimi için ortak bir format sağlamaktadır. SOM, farklı şirketler arasında veb servisleri aracılığıyla bir uygulama bütünleştirme altyapısı sunar. EDI tabanlı standartlarda ise önceden tanımlanmış bir zamanda organizasyonlar arasında mesajlar gönderilir. Daha sonra mesajı alan ve gönderen taraflarda iletişimi sağlayabilmek için bu mesajlar tarafların kullandığı uygun formata dönüştürülür. Fakat, bu teknolojiler büyük şirketlerdeki bütünleştirme zorluklarını tamamen çözmek için yeterli değildir. SOM yaklaşımında, en büyük problem bağlanabilirliktir (Loutas, 2013). Veri tabanlarında saklanan veri belirleyiciler (ID), tanımlandıkları sistem içerisinde tanınırlar ve diğer sistemlerde anlamlarını kaybederler. Belirleyici ile çağırılan bir veb servis operasyonunu bulma uygulama mantığına gömülür. Lojistik sektörünün karmaşıklığı düşünüldüğünde bu durum uygulama mantığını karmaşıklaştırır. EDI tabanlı standartlar ise gerçek zamanlı uygulamalar için uygun değildir, çünkü bilgi güncellendiğinde veri geçersiz hale gelebilir ve bu durum bir mesaj içerisinde doğrudan gönderilmez. Ayrıca, organizasyonlar farklı formatlardaki EDI standartlarını kullanıyor ise çok fazla dönüşüm yapılması gerekmektedir.

Bağlı veri altyapısı, oldukça esnek, genişletilebilir ve istendiğinde dışa açık hale getirilebilir bir bütünleştirme ortamı sağladığından dolayı lojistik sektörünün problemleri için uygun bir teknik çözüm gibi görünmektedir. Bağlı veri standartları ve altyapısı son yıllarda kurum bilgisini ve iş süreçlerini bütünleştirmek için sıklıkla kullanılmaktadır (Hu and Svensson, 2010; Frischmuth et al., 2012; Mihindukulasooriya et al., 2013). Bağlı veri tabanlı bütünleştirme ile belirleyiciler (URI) sadece sistemler çapında değil aynı zamanda bu URIler HTTP protokolü ile erişilebilir olduğundan veb ölçeğinde de tanınmaktadır. Bu sebeple, şirketlerdeki ve bağlı veri bulutu üzerindeki tüm veri kaynakları büyük bir bilgi tabanı oluşturacak şekilde bütünleşik bir bilgi tabanı oluşturur ve yazılım istemleri bu bilgi tabanını birbirinden bağımsız olarak kullanabilir. Bağlı veri teknolojileri genel olarak lojistik sektörünün ve sektörün iyi bilinen sorunlarından biri olan "Konteyner Yaşam Döngüsünün Gözlemlenmesi ve İzlenmesi" probleminin dinamik, dağıtık ve karmaşık doğasına yeni bir çözüm önermektedir. Çünkü, konteyner taşıma sürecinin gerçek zamanlı olarak izlenmesi ve gözlemlenmesi, farklı çalışma alanlarındaki farklı şirketler tarafından gerçekleştirilen alt-taşımaların farklı yazılım sistemlerinde paralel olarak gerçekleşebilecek olmasından dolayı zorlu bir mühendislik görevidir ve bağlı veri alt yapısının kullanımı için oldukça uygun bir alandır.

Bu tez kapsamında, bağlı veri ortamında uygulama geliştirmek isteyen geliştiricilere geliştirim süreci boyunca hangi aktivitelerin gerçekleştirilmesi gerektiğine dair rehber olacak bir yöntem önerilmektedir. Bu yöntem yinelemeli yazılım geliştirmeye dayanmakta, çevik ilke ve pratiklerden esinlenmektedir. Önerilen yöntem aşağıda listelenen maddeler ile bağlı veri geliştirim sürecine katkıda bulunmaktadır;

- Özellik/ Hikaye odaklı bağlı veri geliştirme.
- Bağlı veri geliştirme sürecine çevik mimari (Brown et al., 2010) yöntemini dahil etme.
- Bağlı veri geliştirme sürecinde önce test yaklaşımını uygulama.

Yöntem yukarıda listelenen katkılarından yola çıkılarak, BLOB (A Methodology to **B**ring Agility into **L**inked **O**pen Data Development for **B**usinesses) olarak isimlendirilmiştir. BLOB yöntemi, lojistik sektörünün "Konteyner Yaşam Döngüsünün Gözlemlenmesi ve İzlenmesi" probleminin çözümü için gerçekleştirilen bir uygulamaya uygulanmış ve bu uygulama geliştirimi süresince edinilen deneyimler ile son haline gelmiştir.

Tezin ilerleyen bölümleri şu şekilde düzenlenmiştir. Bölüm 2' de tez çalışması kapsamında incelenen ilgili literatür çalışmaları olan bağlı veri yaşam döngüleri, çevik mimari, çevik analiz, ontoloji geliştirme yöntemleri anlatılacaktır. Bölüm 3, önerilen yöntemin lojistik konteyner taşımacılığı için geliştirilen uygulama sırasında nasıl evirilerek son haline geldiğini ve "Konteyner Yaşam Döngüsünün Gözlemlenmesi ve İzlenmesi " uygulamasını içermektedir. Bölüm 4' te BLOB yöntemi hakkında detaylı bilgi verilmektedir. Son olarak Bölüm 5 tezi sonuçlandırmakta ve gelecekteki çalışma olanaklarından bahsetmektedir.

2 İLGİLİ ÇALIŞMALAR

2.1 Bağlı Veri Yaşam Döngüleri

Literatürde bağlı veri oluşturmak için çeşitli yaşam döngüleri öne sürülmüştür (Hyland and Wood, 2011; Hausenblas, 2011; Villazon-Terrazas et al., 2011; Auer et al., 2012). Bu yaşam döngüleri incelendiğinde bağlı veri oluşturmak için ortak ve net kurallar olmadığı halde benzer işlemlerin gerçekleştirildiği gözlemlenmiştir.

Hyland ve Wood (2011) bağlı veri yaratmak, modellemek ve yayınlamak için 6 adımlık "yemek kitabı" adını verdikleri bir yaşam döngüsü önermişlerdir. Bu çalışma World Wide Web (WWW) standartlarını ve ilkelerini göz önünde bulundurarak Veri Belirleme, Modelleme, Adlandırma, Yeniden Kullanma, Dönüştürme, Yayınlama ve Bakım aktivitelerinden oluşan bir yaşam döngüsü önermiştir.

Hausenblas' ın (2011) çalışmasında ise standart yöntemlerin veri şeması, veri, ve veri üretimi üzerinde tam kontrol sahibi olduğuna dikkat çekilmiş ve günümüzde dağıtık ve açık olan web altyapısının dinamizmine uyum sağlayacak yeni bir yöntemin gerekliliği vurgulanmıştır. Hausenblas' ın yaşam döngüsü Veri Farkındalığı, Modelleme, Yayınlama, Veri Keşfetme, Bütünleştirme ve Kullanım Durumu aktivitelerinden oluşmaktadır.

Bir diğer çalışmada ise Villazon-Terrazas et al. (2011) bağlı veri yayınlama sürecinin yazılım geliştirme süreçlerinde olduğu gibi sürekli gelişen, döngüsel ve arttırımlı bir süreç olması gerektiğini savunmutur. Bu yöntem; Veri Belirleme, Modelleme, Veri Üretme, Yayınlama ve Veriyi Kullanma aktivitelerini takip ederek ilerlemektedir.

Diğer yaşam döngülerinden farklı olarak Auer et al. (2012) daha detaylı bir çalışma gerçekleştirmişlerdir. Bu çalışmada diğer yaşam döngülerine ek olarak verinin kalitesi, saklanması ve zenginleştirilmesi ile ilgili de çalışmalar yapılmış ve süreç boyunca her aktivitede kullanılabilecek araçların bilgisi verilmiştir. Buna göre bağlı veri yayınlama süreci; Veri Elde Etme, Saklama, Veri Oluşturma, Bağ Oluşturma, Zenginleştirme, Gelişme, Keşfetme adımlarını takip etmektedir.

Bu yaşam döngülerinin çoğu genellikle bağlı veri yayınlama süreci ile ilgilenmiş, yayınlanan bağlı verinin nasıl tüketilmesi gerektiğine değinmemiştir.

Fakat, bağlı veri ortamı gibi yeni, sürekli gelişen ve dinamik bir ortamda bu veriyi tüketen uygulamaların da ortam, iş ve mimari değişikliklere hızlıca uyum sağlaması gerekmektedir. Bu noktada, bu tez çalışmasında önerilen metodolojide Çevik Mimari(Brown et al., 2010; Meier et al., 2008) ve Çevik Analiz (Collier, 2011) yaklaşımlarından faydalanılmıştır.

2.2 Çevik Mimari

Çevik Mimari yaklaşımı kısaca mimari bağımlılıkları belirleme, analiz etme ve bu bağımlılık bilincini duyarlı bir geliştirim modeli içerisine dahil etme yeteneği olarak özetleyebiliriz (Brown et al., 2010). Meier et al. (2008) önerdikleri çevik mimari yönteminde beş temel mimari tasarım aktivitesi olduğunu belirtmiştir. Bu aktiviteler aşağıdaki gibidir;

1. Mimari Hedefleri Belirlemek

Mimari hedefler, mimariyi ve tasarım sürecini şekillendiren, sürecin başlangıcını ve bitişini netleştiren temel amaçlar ve kısıtlardır. Bu amaçlar; mimari tasarımda doğru problemlerin çözülmesini sağlarken, mimariyi kullanacak kişilerin gereksinimlerini ve teknoloji, kullanım, dağıtım kısıtlarını anlamayı sağlar. Bu nedenle başlangıçta netleştirilmesi gerekmektedir.

2. Anahtar Senaryolar

Anahtar senaryolar, uygulamanın başarılı olması için gereken en önemli senaryoları içerir. Bu senaryolar, iş açısından kritik, çok kullanılması beklenen veya teknolojik risk içeren senaryolar olabilir. Bu senaryolar fonksiyonel ve fonksiyonel olmayan (kalite parametreleri) gereksinimlerin bir kesişimi olabilir. Ayrıca, son uygulamanın kabulü için önemli senaryolardır. Bu senaryolar, tasarımda neyin önemli olduğuna odaklanmaya ve aday mimarileri değerlendirmeye yardımcı olur.

3. Uygulama Özeti

Uygulama özeti, uygulama tamamlandığında neye benzeyeceğinin bir özetidir. Mimarinin daha gerçekçi olmasını ve uygulamanın gerçek dünya

kısıtları ve kararları ile bağlantısını sağlar. Uygulama özeti, uygulamanın tipi, dağıtım kısıtları, mimari desenler ve ilgili teknolojileri içerir.

4. Anahtar Darboğazlar

Anahtar darboğazlar, uygulama mimarisinde en çok sorun olması beklenen noktaların anlaşılmasını sağlar. Bu darboğazlar, kalite parametreleri göz önünde bulundurularak belirlenir.

5. Aday Çözümler

Anahtar darboğazlar belirlendikten sonra, ilk tasarım yapılır ve bu mimari detaylandırılmaya başlanır. Daha sonra anahtar senaryolar adımına geri dönürelek bu aday çözüm senaryolar ve tanımlanan gereksinimler göz önünde bulundurularak doğrulanır ve süreç tekrarlanarak devam eder.

Çevik mimari yaklaşımı ile geliştirimin her iterasyonda uygulama mimarisi, bu mimarinin dar boğazları, fonksiyonel gereksinimler ve kalite parametrelerini sağlayıp sağlayamadığı sürekli değerlendirilir ve böylece en iyi sonuca ulaşılmaya çalışılır. Bağlı veri uygulamaları açısından ise sadece tasarlanan mimarinin uygulamanın her iterasyonunda değerlendirilmesi yeterli değildir. Aynı zamanda geliştirim boyunca oluşturulan bağlı verinin, bağların ve ontolojilerinde doğrulanması gerekmektedir. Bu bağlamda, önerilen yöntemde Bölüm 2.4' te detayları anlatılan çevik pratiklerden ve çevik analiz (Collier, 2011) yaklaşımından esinlenilmiştir.

2.3 Çevik Analiz

Çevik analiz; veri ambarı, iş zekası ve analizi uygulamaları geliştirmek için çevik pratikler ve ilkeler üzerine kurulmuş bir uygulama geliştirme tarzıdır. Collier (2011) çalışmasında çevik analizin bir geliştirme tarzı olduğunu, uygulama geliştirme boyunca hangi adımda ne yapılacağını belirten bir yöntem olmadığını belirtmiştir. Çevik analiz yaklaşımının uygulama geliştirim sürecine kazandırdığı temel özellikler aşağıda listelenmiştir;

- **İteratif, arttırımlı, evrimsel**

Çevik yazılım geliştirmede olduğu gibi çevik analizde de uygulama geliştirme süreci bir ile üç hafta arasında gerçekleşen kısa iterasyonlar halindedir. Kısa iterasyonlar sayesinde sistem küçük arttırımlar ve kullanıcılardan alınan geribildirimler ile sürekli evrilir. Böylelikle her geliştirim hızlıca doğrulanmış olur.

- **Değer odaklı geliştirme**

Her iterasyonun temel amacı kullanıcı açısından değerli özelliklerin geliştirilmesidir. Uygulama geliştirmenin arka planında yapılan karmaşık mimariler veya sorgular kullanıcının ilgilenmediği, kullanıcı için bir şey ifade etmeyen özelliklerdir. Bu nedenle her iterasyonda en azından bir tane arayüz gibi kullanıcı açısından değerli özellik gerçekleştirilmelidir.

- **Ürün kalitesi**

Geliştirilen her yeni özellik iterasyonlar boyunca tamamen test edilmeli ve bulunan hatalar çözülmelidir. İterasyonların sonunda çıkan ürünler hiçbir hatası olmayan ve kullanıcı tarafından kabul edilmiş ürünler olmalıdır. Bu yüzden geliştirim sürecinin her aşamasında ilgili testler yapılarak her adımın doğruluğu o anda test edilmelidir.

- **Ucu ucuna yetişen süreçler**

Geliştirim süreci boyunca sadece gerektiği kadar gerçekleştirim yapılmalıdır. Gelecekteki aktiviteleri düşünerek fazla gerçekleştirim yapılmamalıdır. Yüksek kalitede, yüksek değerli, çalışan sistemler üretebilmek için diğer aktiviteler için gereken iş minimuma indirilmelidr.

- **Otomasyon, otomasyon, otomasyon**

Çevik geliştirmede en önemli adımlardan biri rutin işlemleri otomatikleştirmektir. Bunlardan en önemlisi test otomasyonudur. Geliştirilen sistemin bir demo ortamında testler otomatik olarak belirli aralıklarla çalıştılmalı ve sistemin hala beklenildiği gibi çalıştığı doğrulanmalıdır. Bu işlemi elle yapmak yerine otomatikleştirmek zaman kaybını engelleyerek daha çok yeni özellik geliştirmeye odaklanmayı sağlar.

- **İşbirliği**

Proje ekibi içerisinde kullanıcılar, ürün sahipleri, sponsorlar, teknik uzmanlar, yöneticiler ve geliştiriciler gibi birçok grup bulunmaktadır. Bu gruplar arasındaki iletişimin iyi olması başarı için kritiktir. Ayrıca çevik projeler için teknik kullanıcılar arasındaki günlük iletişim ve işbirliği projenin başarısı için en önemli gereksinimlerden birisidir.

- **Kendinden organize, kendini yöneten takımlar**

Çevik yazılım geliştirme sürecinde geliştirim takımı bir iterasyon boyunca ne kadar iş bitirebileceklerine kendisi karar verir ve bu taahhüdü gerçekleştirmek için çalışır. Proje yöneticisi ise geliştirim takımının çalışması için uygun ortamı sağlar ve takımın kullanıcı ve diğer proje üyeleri ile iletişimini kolaylaştırır.

Bağlı veri uygulamaları ve iş zekası uygulamalarının birbirine olan yakınlıkları düşünüldüğünde çevik analiz yaklaşımını bağlı veri ortamına uyarlayarak kullanmak ve böylece küçük iş adımları ile kullanıcı açısından değerli, kullanıma hazır ve tamamen test edilmiş uygulamaları hızlıca geliştirmek mümkündür. Fakat bu noktada önemli olan bir diğer konu verinin temsil edilme formatıdır. Bağlı veri ortamında verilerin hangi formatta olacağı ontolojiler aracılığıyla belirlenir. Bu nedenle bağlı veri oluşturulurken bir ontoloji geliştirmek veya var olan uygun ontolojiyi bulmak gerekmektedir. Fakat, ontoloji geliştirme süreci kendi içerisinde bir çok detay bulunduran, karmaşık bir süreçtir. Bölüm 2.4' te literatürdeki ontoloji geliştirme yöntemleri kısaca anlatılacaktır.

2.4 Ontoloji Geliştirme Yöntemleri

Ontoloji, yapısal ve yapısal olmayan alan bilgilerinin yapısallaştırılarak görsel ve mantıksal bir biçimde ifade edilebilmesini sağlar. Bunu yaparken de söz konusu alandaki kavramları ve bunlar arasındaki ilişkileri ortaya çıkarır. Ontoloji Geliştirim Yöntemi, ontoloji geliştirme sürecinde izlenecek etkinlikleri, etkinliklerin birbiriyle ilişkilerini ve üretilecek çıktıları tanımlamayı hedeflemektedir. Ontoloji oluştururken izlenecek çok değişik yöntemler olmasına rağmen bu süreçte ortak olan iki önemli kural vardır;

- Bir alan pek çok farklı şekilde modellenebilir. Her zaman dikkate değer farklı seçenekler bulunabilir. Bu durumda en iyi çözüm, ontolojiyi kullanacak olan uygulamaları dikkate alarak en uygun alternatifi seçmektir.

- Ontolojiyi oluşturan kavramlar (sınıflar) mümkün olduğunca söz konusu alandaki nesnelerle (fiziksel ya da mantıksal) ve bunlar arasındaki ilişkilerle örtüşmelidir.

Bu tez çalışmasında literatürdeki Methontology (Fernandez- Lopez et al., 1997), Ontology Development 101 (Noy and McGuinness, 2001), Diligent (Pinto et al., 2004), Upon (Nicola et al., 2009) ve NeOn (Suarez-Figueroa et al., 2012) yöntemlerinden esinlenerek çevik bağlı veri uygulamaları geliştirmek için önerilen yöntemin ontoloji geliştirme bölümü şekillendirilmiştir.

2.4.1 Methontology

Bu yöntem ontoloji geliştirim sürecinde hangi aktivitelerin yapılacağına odaklanmıştır fakat bu aktivitelerin bir sırasını vermez. Methontology yöntemi bir ontoloji geliştirim sürecinde aşağıdaki aktivitelerin bulunması gerektiğini önerir;

- Ontoloji geliştirmeye başlamadan önce yapılacak temel görevler planlanmalı ve bu görevlerin ne kadar zamanda ne kadar kaynakla gerçekleştirileceği planlanmalıdır. (Planlama)

- Ontolojinin amacı ve kapsamı belirlenmelidir. Bu işlem sırasında "Ontoloji neden geliştiriliyor?", "Ontolojinin kullanım amacı ne olacak?", "Ontolojiyi kim kullanacak?" gibi soruların cevabı verilir ve bu cevaplar ile bir gereksinim dökümanı hazırlanır. (Belirleme)

- Veri kaynaklarının bir listesi çıkarılmalı, süreçlerin nasıl gerçekleştirileceği, kullanılacak teknikler açıklanmalıdır. (Bilgi Edinme)

- Yeterince bilgi edindikten sonra alanın kavramsal bir modeli oluşturulmalıdır. (Kavramsallaştırma)

- Kavramsallaştırılmış model resmi bir modele çevrilmelidir. (Biçimlendirme)

- Ontolojiler yeniden kullanılmak için geliştirilmiştir. Alanda var olan ontolojiler araştırılarak mümkün olduğunca yeniden kullanılmaya çalışılır. (Bütünleştirme)

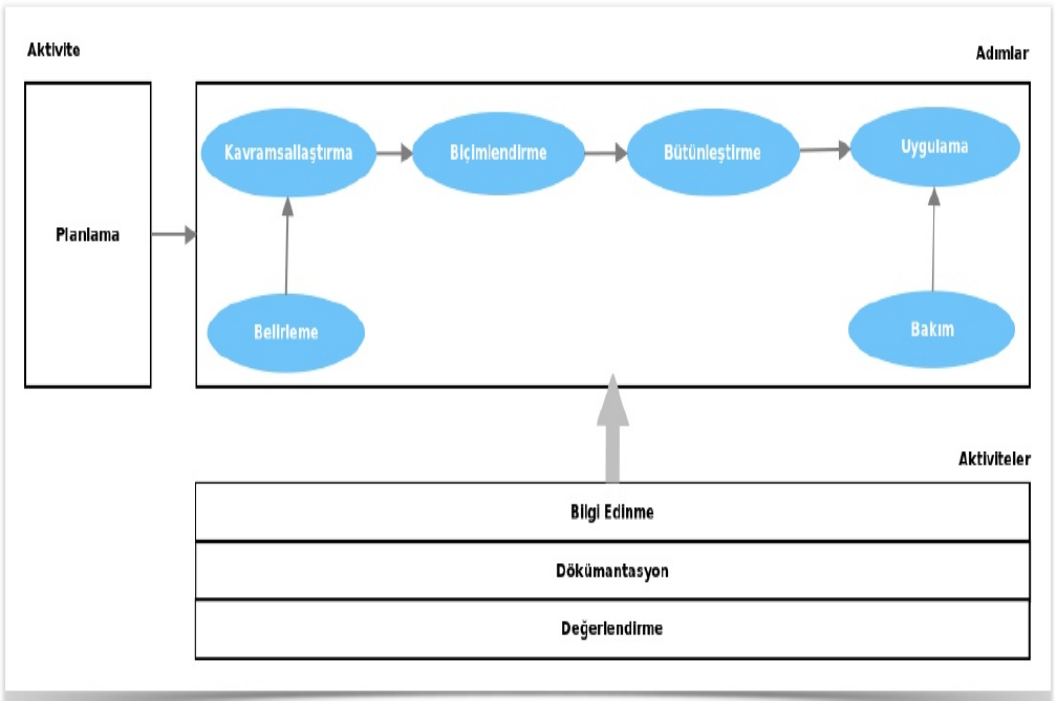
- Ontoloji resmi bir ontoloji dili ile kodlanır. (Uygulama)

- Ontolojide hata olup olmadığı, ontolojinin doğru olup olmadığı değerlendirilir. (Değerlendirme)

- Ontolojinin yeniden kullanılabilirliği için bir dökümantasyon olması önemlidir. Dökümantasyon hazırlanır. (Dökümantasyon)

- Ontolojiye yeni tanımlar eklemek ya da var olan tanımları düzenlemek gerekebilir. Bu açıdan ontolojinin bakımı önemlidir. Ayrıca bakım için bir rehber de gereklidir. (Bakım)

Yukarıda sıralanan aktiviteler bir ontoloji geliştirme sürecinde olması gereken aktivitelerdir. Planlama ve Bakım aşamalarının ilk ve son aktiviteler olacağı açıktır fakat diğer aktiviteler için biri birinden önce diğeri sonra demek pek doğru olmaz çünkü süreç içerisinde bu aktiviteler birbiri ile iç içe geçmiştir. Bilgi edinme, Dökümantasyon ve Değerlendirme adımları tüm ontoloji geliştirme sürecine yayılmıştır. Fakat bu aktivitelerin ağırlıkları süreç boyunca değişiklik gösterebilmektedir. Örneğin; Bilgi edinme aktivitesi Belirleme adımından diğer adımlara doğru azalarak yapılmaktadır. Dökümantasyon aktivitesi ise son adımlara doğru daha çok yapılmaktadır ve Değerlendirme aktivitesi de ilk aşamalarda daha çok yapılmaktadır. Şekil 2.1' de Methontology ontoloji geliştirme sürecinin adımları gösterilmektedir. Nasıl insan hayatı Çocukluk, Ergenlik, Gençlik, Yetişkinlik, Yaşlılık olarak ilerliyorsa ontoloji yaşam döngüsü de Belirleme, Kavramsallaştırma, Biçimlendirme, Bütünleştirme, Uygulama ve Bakım olmak üzere ilk üründen son ürüne doğru ilerlemektedir. Burada ilk ürün ontoloji geliştirmemize sebep olan gereksinimlerken son ürün de değerlendirilmiş, dökümantasyonu yapılmış, resmi bir dil ile gerçekleştirilmiş ontolojidir.



Şekil 2.1. Methontology ontoloji geliştirme yöntemi

2.4.2 Ontology Development 101

Ontology Development 101 çalışması net bir yöntem değildir, fakat ontoloji geliştirmeye giriş sayılabilecek bu çalışmada Noy and McGuinness (2001) daha önceki deneyimlerini derleyerek ontoloji geliştirme sürecinde olması beklenen aktiviteleri tanıtmıştır. Buna göre ontoloji geliştirim süresi aşağıda listelenen aktiviteleri içermelidir;

- Ontolojinin alanını ve kapsamını belirlemek

Bu adımda "Ontolojinin kapsayacağı alan nedir?", "Ontoloji ne için kullanılacak?", "Ontoloji hangi sorulara cevap vermeli?", "Ontolojiyi kim kullanacak ve ontolojinin bakımını kim yapacak?" gibi sorulara cevap verilir. Bu soruların cevapları ontoloji tasarım sürecini değiştirebilir, fakat herhangi bir zamanda ontolojinin kapsamını sınırlamak için yardımcı olacaktır. Ontolojinin kapsamını belirlemek için bir diğer yolda yeterlilik sorularıdır. Bu sorular ontolojinin cevaplamasını istediğimiz soruların listesidir. Bu sorular daha sonra ontolojinin test edilmesi için yardımcı olacaktır. Yeterlilik soruları birer karalama gibidir çok detaylı bilgi içermesine gerek yoktur.

- Mevcut ontolojilerin yeniden kullanılması

Ontoloji geliştirirken ontoloji geliştirilecek alan ile ilgili var olan kaynakları incelemek bu kaynakları genişleterek yeniden kullanımını düşünmek önemlidir. Eğer geliştirilen sistem başka uygulamalar ile etkileşim içinde çalışacaksa zaten alanında doğruluğu kabul edilmiş sözlükleri kontrol edilmiş bir ontolojiden yararlanmak, onu kullanmak bir ihtiyaç olabilir. Bir çok ontolojinin ontoloji geliştirme editörlerine dahil edilebilen bir elektronik formu bulunmaktadır.

- Önemli terimleri belirlemek

Ontolojide kullanılacak olan, hakkında konuşulan ya da kullanıcılara açıklama yaparken bahsedilen terimlerin bir listesi çıkarılır. İlk olarak kavramların, terimler arası ilişkilerin, kavramların sahip olduğu özelliklerin

çakışmasından ya da kavramların sınıf mı yoksa bir ilişki veya özellik olup olmadığı ile ilgili endişelenmeden kapsamlı bir liste çıkarılır. Daha sonra ise sınıf hiyerarşisi ve kavramların özellikleri tanımlanır. Sınıf hiyerarşisini ve kavramların özelliklerini tanımlamak birbiri ile iç içe geçmiş işlemlerdir birini diğerinden ayırmak pek mümkün değildir. Bu yüzden genellikle hiyerarşi içinde bir takım kavramların tanımları yapılır daha sonrada bu kavramların özellikleri tanımlanır. Bu adımlar ontoloji tasarım sürecinin en önemli adımlarıdır.

- Sınıf ve sınıf hiyerarşilerini tanımlamak

Bu adımda Uschold ve Gruninger ' in (1996) çalışmasından yararlanılmış ve yukarıdan-aşağıya, aşağıdan-yukarıya veya kombinasyon yaklaşımlarından biri ile sınıf hiyerarşilerinin oluşturulabileceği vurgulanmıştır. Bu yaklaşımlardan herhangi biri diğerinden daha iyi değildir. Geliştiriciler tercihlerine göre kendileri için en uygun olanı seçebilir. Eğer geliştiricinin elinde alanın sistematik yukarıdan-aşağı bir görüntüsü var ise o zaman yukarıdan-aşağıya yaklaşımını kullanmak daha kolay olabilir ya da önce genel bir sınıflandırma yapılmak isteniyorsa gene yukarıdan-aşağıya yaklaşımı kullanılabilir. Eğer geliştirici daha belirli örneklerle başlamak istiyorsa o zaman aşağıdan-yukarıya yaklaşımı kullanılabilir. Fakat hangi yaklaşım kullanılırsa kullanılsın öncelikle sınıflar tanımlanarak başlanır. Bir önceki adımda hazırlanan terimler listesinden nesneleri bağımsız olarak tanımlayan terimleri seçeriz. Bu terimler ontolojideki sınıflardır ve sınıf hiyerarşisinin temel noktalarıdır. Bu sınıfların bir sınıfın örneği olup olmadığına bakarak hiyerarşi organize edilir, çünkü her sınıf muhakkak bir sınıfın örneğidir.

- Sınıfların özelliklerini belirlemek

Sınıflar tek başına tanımlanan yeterlilik sorularını cevaplamak için yeterli bilgi içermez. Sınıflardan bir kısmı tanımlandıktan sonra kavramların içsel yapısını tanımlamalıyız. Daha önce hazırlanan terimler listesinden sınıfları seçerek sınıflar tanımlanmıştı. Bu seçim işleminden sonra geriye kalan terimler genellikle tanımlanan sınıfların özellikleridir. Bu özellikler

sınıflarla ilişkilendirilir. Listedeki her özellik için hangi sınıfı tanımladığına karar verilir. Özellikler mümkün olduğunca en genel olan ve diğer sınıfları kapsayan sınıflar ile ilişkilendirilmelidir.

- Özelliklerin değerlerini tanımlamak

Bu adımda özelliklerin alabileceği değerler, değer tipleri, değer sayısı ve bu özelliğin alabileceği diğer özellikler belirlenir.

- Örnekleri yaratmak

Son adım sınıfların örneklerini yaratmaktır. Bir örnek tanımlamak için şunlar gereklidir; bir sınıf seçmek, bu sınıfın örneğini yaratmak, özellik değerlerini girmek. Örneğin A sınıfının bir örneğini yaratmak istiyorsak A sınıfını seçeriz, bir örnek veri yaratır ve özellik değerlerini gireriz.

2.4.3 Diligent

Diligent yöntemi Oluşturma, Yerel adaptasyon, Analiz, Revizyon, Yerel Güncelleme olmak üzere beş temel aktiviteye ayrılmıştır. Diligent yönteminin diğer yöntemlerden temel farklı ontoloji geliştirme sürecini birbirinden farklı amaç ve gereksinimleri olan farklı katılımcılar ile farklı yerlerde dağıtık olarak geliştirilebilen bir süreç olarak düşünmesidir. Bu nedenle bu sürecin çevrimiçi olarak gerçekleştirilmesi gerektiğini savunur.

- **Oluşturma**

Oluşturma adımı alan uzmanları, kullanıcılar, bilgi mühendisleri ve ontoloji mühendislerinin katılımıyla ilk seviye ontolojinin geliştirilmesi işlemi ile başlar. Bu aşamada oldukça küçük ontolojiler üretilir ve üretilen bu ilk seviye ontolojinin alan açısından tam olmasına gerek yoktur.

- **Yerel Adaptasyon**

İlk seviye ontoloji ortaya çıkarıldıktan sonra kullanıcılar bu ontolojiyi kullanır ve kendi amaçlarına göre uyarlar. Kullanıcılar geliştirilen ontolojiyi kendi yerel ortamlarında istediği gibi düzenleyebilir fakat, kullanıcılar arasında paylaşılan ontoloji üzerinde direk olarak değişiklik yapamazlar. Paylaşılan ontoloji için yapılan değişiklik istekleri kontrol tahtası adı verilen bir ortamda saklanır.

- **Analiz**

Analiz adımında saklanan değişiklik istekleri ve yerel ontolojiler kontrol tahtası editörleri tarafından incelenerek bu ontolojilerdeki benzer noktalar analiz edilir. Bu analizler sonucunda hangi değişikliklerin paylaşılan ontolojiye yansıtılacağına karar verilir.

- **Revizyon**

Hangi değişikliklerin paylaşılan ontolojiye yansıtılacağına karar verildikten sonra paylaşılan ontoloji revize edilir. Bu işlem sık aralıklarla tekrar edilir ve böylece paylaşılan ontoloji ile yerel ontolojiler arasında büyük farklılıklar olmaması sağlanır.

- **Yerel Güncelleme**

Paylaşılan ontolojinin yeni bir versiyonu oluştuğunda kullanıcılar yerel ontolojilerini buna göre günceller. Eğer küçük değişiklikler var ise kullanıcılar yerel ontolojiyi güncellemek yerine direk olarak paylaşılan ontolojiyi kullanabilir.

2.4.4 Upon

UPON metodolojisi UP(Unified Process) ve UML' in (Unified Modelling Language) avantajlarından faydalanarak ontoloji geliştirmek için bir yaklaşım sunar. Bu kapsamda kullanıcı durumu odaklı, iteratif ve arttırımlı bir yapı

sunmaktadır. UPON döngülerden oluşmaktadır ve her döngü dört süreç içermektedir. Süreçler tamamlandığında ontolojinin yeni bir versiyonu ortaya çıkmış olur. Bu süreçler şöyledir;

- **Başlangıç süreci:** İhtiyaçları belirleme ve tam olmayan bazı kavramsal analizler yapma ile ilgilenir. Test ya da uygulama içermez.

- **Detaylandırma süreci:** Analiz tamamlanmış, temel kavramlar tanımlanmış ve kabataslak yapılandırılmış haldedir. Ontoloji iskeleti çıkacak kadar tasarım ve ön uygulama yapılabilir.

- **Geliştirim süreci:** Tasarım ve uygulama daha çok bu süreçte gerçekleştirilir. Ayrıca bu süreçte bir önceki süreçte gözden kaçan kavramların eklenmesi için bir miktar analize gerek duyulabilir.

- **Geçiş süreci:** Bu sürecin temel aktivitesi olan test etme gerçekleştirilir ve sonunda ontolojinin bir versiyonu çıkmış olur.

UPON süreçlerinin her biri alt iterasyonlara bölünmüştür ve her alt iterasyon boyunca aşağıda beş iş akışı yer alır. Şekil 2.2' de UPON metodolojisi gösterilmektedir.

- **İhtiyaçları Belirleme:** İlk toplantı boyunca bilgitabanı uzmanı ve alan uzmanı görüşmeleri ve inceledikleri alan dökümanları sonucunda ontoloji geliştirmek için bir rehber oluşturur. İlk görev kullanıcı bakış açısından ontolojinin amaçlarını tanımlamaktır. Bu amaçla, aşağıdaki maddeler gerçekleştirilir;
  İlgi alanı ve kapsamı belirlemek: İlgi alanı belirlenirken modellenecek olan ontolojinin ilgi alanı sınırlandırılır. Eğer alan genişse bir ya da iki alt alana ayırılabilir. Ontolojinin kapsamı belirlenirken sunulacak olan kavramların en önemlileri ve ontolojik bağları tanımlanır.

  İş amaçlarını tanımlamak: Ontoloji ne için kullanılacak ve hedef kullanıcılar kim olacak saptanır.

İş amaçları gerçekleştirecek kullanıcı hikayelerini yazmak: İş amaçlarının anlaşılması ile oluşan senaryoların yazılması.

Uygulama Sözlüğü(US) yaratmak: Alan uzmanının bilgilerinden, uygulamaya özel dökümanlar ve kullanıcı durumlarından bir terminoloji derlenir. Böylece ön bir uygulama sözlüğü elde edilmiş olur. Bu aşamada metin dökümanlardan bilgi çıkarmak için bir takım otomatik araçlarda kullanılabilir.

Yeterlilik Sorularını(YS) tanımlamak: Yeterlilik soruları ontolojiyi test süreci boyunca yapılan ontolojik bağlılıkların, ontoloji kapsamının ve derinliğinin değerlendirilmesi için kullanılacak olan sorulardır. Bu sorular ontolojinin kavramsal seviyede nelere cevap vereceğini belirlemek içindir.

İlişkili kullanıcı durumlarını modellemek: Yeterlilik sorularına karşılık gelen kullanıcı durumları modellenir ve bir öncelik sırasına koyulur.

Bu iş akışı sonucunda çıkan ürünler yeterlilik soruları, kullanıcı durumu modelleri ve uygulama sözlüğüdür.

- **Analiz:** Bir önceki iş akışında tanımlanan ontoloji ihtiyaçlarının yapılmasıyla ilgilenir. Ontolojik bağlılıklar mevcut kaynakların yeniden kullanılması ve kavram iyileştirme yoluyla ortaya çıkarılır ve bu kavramlar ile bir alan sözlüğü oluşturulur. Bir referans açıklayıcı sözlük oluşturmak için uygulama sözlüğü terim tanımları da eklenerek zenginleştirilir. Bu aşamada aşağıdaki adımlar izlenir;

  Alan kaynaklarını elde etme ve alan sözlüğü oluşturma: Alan sözlüğü mevcut dökümanlar, raporlar, teknik standartlar, sözlükler, alan ile ilgili mevcut ontolojiler ve bu alanda kullanılan terminolojilerden elde edilen terimler ile oluşturulur.

  Referans sözlüğü oluşturma: Uygulama sözlüğü ve alan sözlüğü seçici bir şekilde birleştirilerek elde edillir. Bu birleştirme süreci boyunca terimler 3 ana alana gruplanır. Bu alanlar kesişim alanı, alana özgü ve uygulama özgü olarak belirlenmiştir. Referans sözlüğü mutlaka kesişim alanındaki

tüm terimleri içermelidir.Ayrıca alan uzmanı ve kullanıcıların onayladığı iki farklı alandan gelen terimleri de içerir.

UML diyagramları ile kullanıcı hikayelerini modelleme: Kullanıcı hikayelerini modelleme, ihtiyaçları belirleme iş akışında belirlenen kullanıcı durumlarının aktivite ve sınıf diyagramlarını çizme bu aktivitenin hedefidir. Burada elde edilen UML diyagramları doğrulama için kullanılır. Modellenen tüm sınıfların, aktörlerin, aktivitelerin ontolojide kavram olarak bir karşılığı olmalıdır.

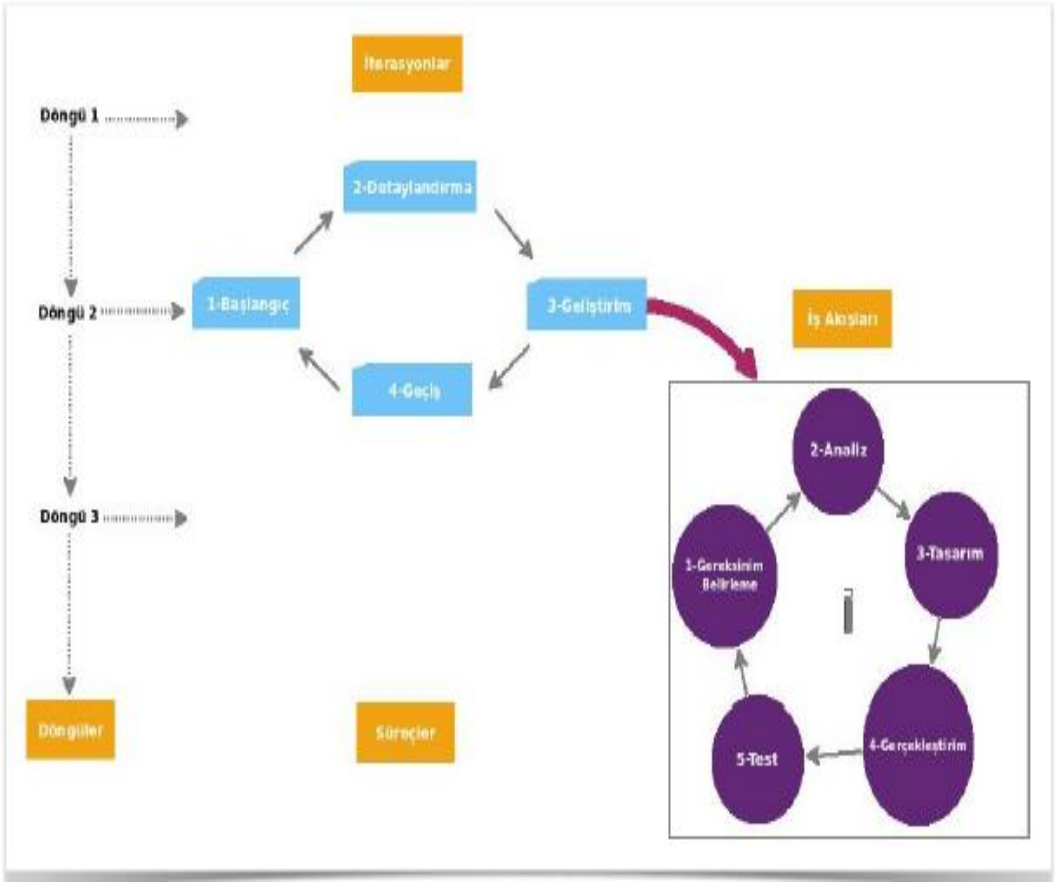
Referans açıklayıcı sözlük oluşturulması: Referans sözlüğü kullanılarak ve bilgilendirici tanımlamalar eklenerek referans açıklayıcı sözlüğünün ilk versiyonu oluşturulur.

Bu iş akışı sonunda alan sözlüğü, referans sözlüğü, UML sınıf ve aktivite diyagramları ve referans açıklayıcı sözlük oluşur.

- **Tasarım:** Bu iş akışının temel amacı açıklayıcı referans sözlüğündeki girdilere bir ontolojik yapı vermektir. Bu amaçla kavramlar kategorilere ayrılır ve kavramlar arası hiyerarşiler oluşturulur. Aktivite sonunda ontolojinin bir anlamsal ağ görüntüsü oluşur.

- **Uygulama:** Bu aşamanın amacı ontolojiyi bir dil ile kodlamaktır. Kodlanacak dil seçilirken anlatım gücüne, çıkarsamaları hesaplama karmaşıklığına, metotlarına ve alanda kabul görme seviyesine dikkat etmek gerekmektedir.

- **Test:** Ontolojinin kalitesi sözdizimsel, anlamsal, pragmatik ve sosyal olmak üzere dört ayrı açıdan değerlendirilir.



Şekil 2.2. UPON yöntemi

2.4.5 NeOn

Neon yöntemi, sabit ve kesin kurallardan oluşan bir yöntem değil ontolojinin gereksinimine göre şekillenen çeşitli yöntemler sunar. Yeniden yapılandırma, modüllere ayrıştırma, farklı dil ve kültürlere özelleştirme, ontoloji tasarım desenleri ve ontolojik olmayan kaynaklar ile bütünleştirme gereksinimi olan ontolojileri geliştirebilmek için 9 faklı senaryo önerir.

Çevrimiçi ulaşılabilen ontoloji sayısının gittikçe artması ile ontoloji geliştirim süreci yeniden kullanılabilirlik temelli olmaya başlamıştır. Yeniden kullanılabilirliğin seviyesi ise diğer ontolojilere, ontoloji modüllerine, ontoloji ifadelerine ve tasarım desenlerine ve ontolojik olmayan kaynaklara bağımlıdır. Bu bağımlılık ontoloji geliştirim sürecinin farklı organizasyonlardaki farklı insanlar ve farklı kaynaklardan oluşan bir süreç haline getirmiştir. Bu yeni yapı ile birlikte,

güçlü bir yöntem sunabilmek ve ontolojilerin ortak geliştirimini sağlayabilmek için yeniden kullanılabilirlik gittikçe önemli bir hale gelmektedir. Neon yöntemi, yukarıda bahsedilen dağıtıklık, ontolojinin dinamik geliştirimi, yeniden kullanılabilirlik gibi problemleri aşabilmek için senaryo tabanlı bir yöntem sunmaktadır ve aşağıda listelenen bileşenleri içerir;

- Ontoloji ağlarında bulunması muhtemel aktiviteler ve süreçlerin tanımlandığı bir sözlük (Neon Glossary).

- Ontoloji ve ontoloji ağı geliştirmek için 9 senaryo. Bu senaryolar bir üst adımda bahsedilen aktivite ve süreçlere ayrıştırılarak oluşturulmuştur.

- 2 adet ontoloji ağı yaşam döngüsü modeli. Bu modeller Neon Glossary içerisindeki aktivite ve süreçlerin nasıl düzenleneceğini belirler. Bu yaşam döngüleri, klasik yazılım geliştirme yöntemlerinebenzer olarak İteratif Arttırımlı Ontoloji Ağı Geliştirim Modeli ve Şelale Ontoloji Ağı Yaşam Döngüsü Modelidir.

- Süreç ve aktiviteler için metodolojik bir rehber**.**

Buna göre, ontoloji geliştiriciler kendileri için uygun olan senaryoyu neon senaryolarını inceleyerek belirler, daha sonra bu senaryo içerisinde belirtilmiş aktivite ve süreçleri Neon yönteminin önerdiği rehberi takip ederek gerçekleştirir. Bu senaryoları aşağıdaki gibi listeleyebiliriz;

- Tümüyle ontoloji geliştirme

- Ontolojik olmayan kaynakları yeniden kullanma ve yeniden mühendislik

- Ontolojik kaynakları yeniden kullanma

- Ontolojik kaynakları yeniden kullanma ve yeniden mühendislik

- Ontolojik kaynakların yeniden kullanılması ve birleştirilmesi

- Ontolojik kaynakların yeniden kullanılması, birleştirilmesi ve yeniden mühendislik
- Ontoloji tasarım desenlerini yeniden kullanma
- Ontolojik kaynakları yeniden yapılandırma
- Ontolojik kaynakların yerelleştirilmesi

3. YÖNTEMİN UYGULANMASI

Yöntemin uygulanması süreci bir SANTEZ projesi olan "BAKİ - Bağlı Veri Tabanlı Konteyner İzleme Sistemi" isimli proje kapsamında ARKAS holdingin bilişim şirketi olan BİMAR Bilgi İşlem Hizmetleri A.Ş. ile birlikte gerçekleştirilmiştir. Bu kapsamda, önerilen yöntem lojistik sektörünün iyi bilinen sorunlarından biri olan "Konteyner Yaşam Döngüsünün Gözlemlenmesi ve İzlenmesi" problemini çözmek için uygulanmıştır. ARKAS, deniz, karayolu, demiryolu, havayolu, gemi ve liman operasyonları gibi birçok farklı alanda hizmet vermektedir. Bu farklı çalışma alanlarında farklı roller tarafından gerçekleştirilen konteyner taşımasının izlenmesi ve gözlemlenmesi zorlu bir mühendislik görevidir. Bu bölümde, lojistik sektörünün bu probleminin çözümü için bağlı veri altyapısı kullanılarak geliştirilen mimari çözüm ve bu çözümü gerçekleştirirken izlenen yöntemin evrimi anlatılmaktadır. Başlangıçta önerilen yöntem uygulama geliştirimi boyunca birden çok veri kümesinin bağlı veri olarak açılması sonucu ortaya çıkan yeni problemlerin çözümüne imkan verecek şekilde güncellenmiş ve son haline ulaşmıştır. Uygulamanın ilk iterasyonunda önerilen yöntem Şekil 3.1' deki gibidir. Bu yöntem, yinelemeli yazılım geliştirmeye dayanmakta ve çevik mimari yaklaşımından esinlenmektedir.

"Konteyner Yaşam Döngüsünün Gözlemlenmesi ve İzlenmesi" problemi lojistik konteyner taşımacılığının dağıtık, dinamik ve karmaşık doğasından kaynaklanan bir problemdir. Lojistik endüstrisinde, müşterilerin yükleri bir başlangıç noktasından varış noktasına konteynerler içerisinde taşınır. Bu taşıma süresince konteynerler liman, depo karayolu, demiryolu ve havayolu gibi değişik çalışma alanlarında işlem görmektedir. Örneğin; Türkiye/Manisa' dan Almanya/Münih' teki müşterisine yük göndermek isteyen bir şirketi düşünelim. Bu şirket, taşıma için bir aracı şirket ile anlaşır. Bu taşıma, dört aşama olarak planlanır; Manisa' dan İzmir Alsancak limanına bir karayolu taşıması, İzmir Alsancak limanından Atina' nın Pire limanına bir denizyolu taşıması, Pire limanından İtalya'nın Venedik limanına bir denizyolu taşıması ve Venedik' ten Münih' e bir karayolu taşıması. Farklı yöntemlerle gerçekleşen bu taşımalar süresince müşteri yükünün ne durumda olduğu, nerede olduğu gibi bilgileri öğrenmek istemektedir (Ahn, 2005). Taşıma süresince, müşteriler sadece ARKAS holding bünyesindeki müşteri operasyon birimini telefon ile arayarak bilgi alabilmektedir. Tüm taşıma şirketleri kendilerine özgü bilgi teknolojileri ve altyapıları kullanmaktadır. Dahası, bu şirketlerin sistemleri arasında bir bütünleştirme ortamı bulunmamaktadır. Taşıma sürecindeki herhangi bir gecikme

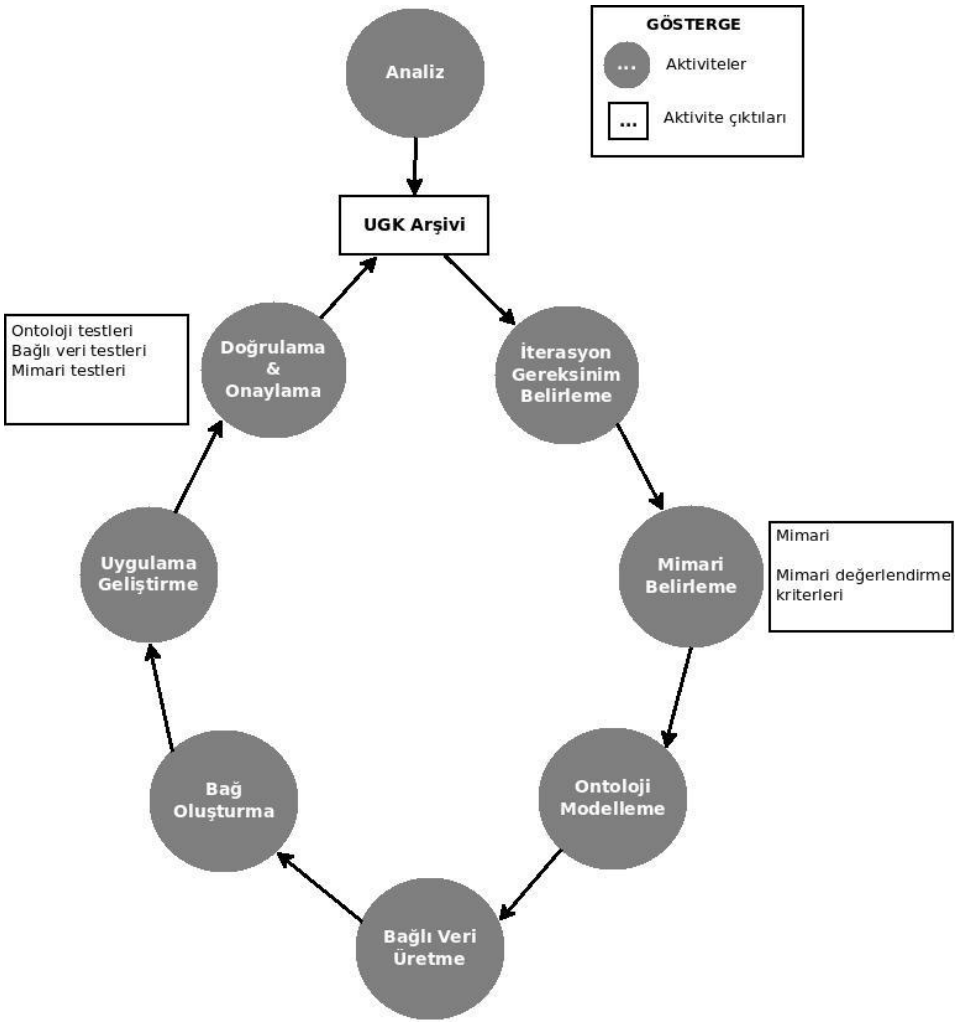
diğer tüm taşımaları etkilemektedir. Bu nedenle, taşıma sürece katılan şirketleri sürekli arayarak izlenmektedir ve bu durum müşteri operasyon birimine çok fazla operasyonel yük getirmektedir. Bu tez çalışmasında, lojistik sektörünün "Konteyner Yaşam Döngüsünün Gözlemlenmesi ve İzlenmesi" problemini bağlı veri altyapısı kullanarak çözmek amaçlanmaktadır. Bu amaçla, endüstrinin takip eden iki ana gereksinimi gerçekleştirilecektir;

- Bir Bütünleştirme Ortamı Sağlamak

  Farklı yazılım sistemlerine dağılmış olan konteyner taşıma tarihçesi bütünleştirilmeli ve gözlemlenebilmelidir.

- Konteynerleri İzlemek

  Farklı çalışma alanlarındaki oluşan konteyner taşıması olayları izlenmeli ve müşteri bu olaylar ile ilgili bilgilendirilmedlidir.

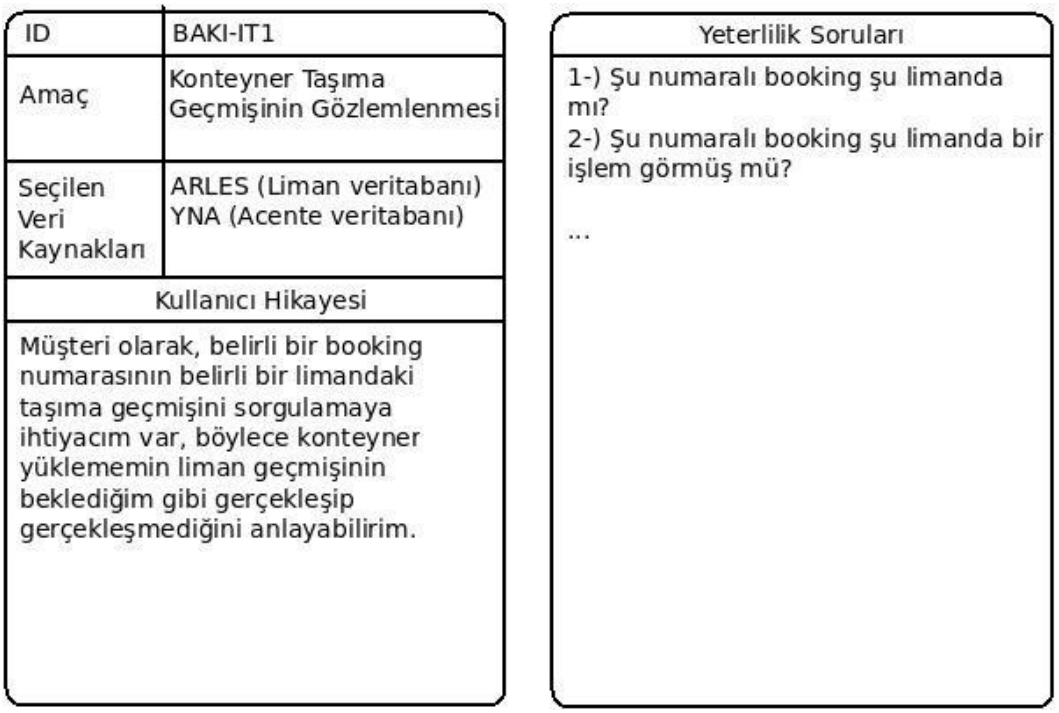


Şekil 3.1. İterasyon 1' de uygulanan yöntem

Uygulama geliştirimine başlamadan önce proje tarafları tarafından alan ve uygulama gereksinimleri analiz edilerek “Konteyner taşımasının izlenmesi” (İzleme amacı) ve “Konteyner taşıma geçmişinin gözlemlenmesi” (Gözlemleme amacı) olmak üzere iki temel amaç belirlenmiştir. Analiz sırasındaki tartışmalar boyunca sistemin birincil müşterisinin müşteri operasyon birimi çalışanları olduğu anlaşılmıştır. Bu noktada kullanıcı hikayeleri müşteri bakış açısından üretilmiştir. Gözlemleme amacı için kullanıcı hikayeleri genellikle konteyner taşıma yaşam döngüsünün alt-taşımaları ile ilgili bilgiye odaklanmıştır. Örneğin; bir kullanıcı hikayesi “Müşteri olarak, belirli bir rezervasyon numarasının belirli bir limandaki taşıma geçmişini sorgulamaya ihtiyacım var, böylece konteyner yüklememin liman geçmişinin beklediğim gibi gerçekleşip gerçekleşmediğini anlayabilirim.” şeklinde tanımlanmıştır. Bu hikayeler analiz edildiğinde, geliştirim takımı çoğu hikayenin birden çok veri kaynağındaki bilgiye gereksinim duyduğunu gözlemledi. Örneğin; bir rezervasyon numarası ile liman tarihçesini sorgulmak istediğimizde acente ve liman veri kaynaklarının bilgisine gereksinim olduğu görülmüştür. İzleme amacında ise hikayeler genellikle “Müşteri olarak, konteyner yüklememin içerisinde bulunduğu gemi limana yanaştığında bilgilendirilmeye ihtiyacım var, böylece liman sahasında yaşanabilecek gecikmelerden dolayı gerçekleşecek problemleri önleyebilirim.” örneğindeki gibi izleme kurallarını tanımlamaktadır. Böylece, amaçlara ait kullanıcı hikayeleri, kullanıcı hikayelerinin yeterlilik soruları ve kullanıcı hikayelerini ilgilendiren veri kaynakları belirlendikten sonra uygulama gereksinim kartları (UGK) tanımlanmış ve UGK arşivine eklenmiştir. Şekil 3.2' de iterasyon 1' e ait örnek UGK görülmektedir.

3.1 İterasyon 1' e Genel Bakış

İterasyon Gereksinim Belirleme (İGB) aktivitesinde, müşteri gözlemleme amacının günlük operasyonlar için daha acil olduğunu belirtti. Böylece, bu amaca ait UGK kartlarının daha öncelikli olduğu anlaşılmıştır. Geliştirim takımı liman ve acente veri kaynaklarının konteyner taşıma yaşam döngüsündeki büyük bir bölüme ait veriyi içerdiğinden dolayı kritik olduğunun altını çizmiş. Bu nedenle, bu kaynakları içeren UGK kartları bu iterasyona dahil edilmiştir.

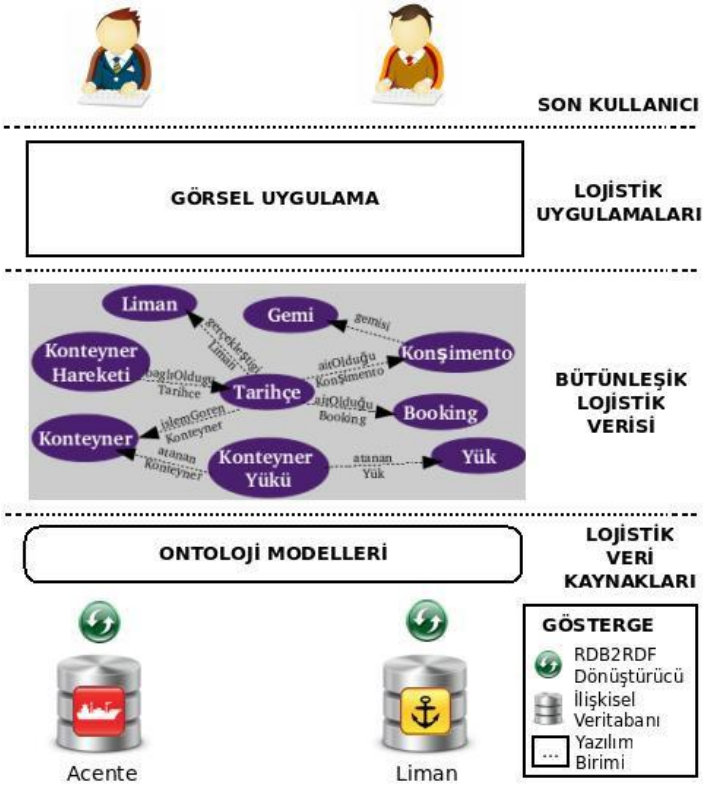


Şekil 3.2. İterasyon 1' e ait örnek UGK

Geliştirim takımı *Mimari Belirleme* aktivitesine geçmiş ve eskiyen veri problemi ile uğraşmamak için Sorgu Birleştirme desenini kullanmaya karar vermiştir. Acente ve liman veri kaynakları iki farklı ilişkisel veritabanında saklandığından dolayı geliştirim takımı bağlı veri üretmek için Şekil 3.3' de görüldüğü üzere bir RDB2RDF aracı kullanmaya karar vermiştir. Bu noktada, geliştirim takımı tarafından mimari değerlendirme için veri getirme kriterleri, kalite parametreleri ve bu parametrelerin sınırları belirlenerek test planı olarak tanımlanmıştır. Birleştirilmiş sorguların işletilme süresinin temel veri getirme kriteri olmasına karar verilmiş ve bu sürenin 0.5 sn' yi geçmememesi gerektiği kararlaştırılmıştır. Ayrıca, son uygulamanın performans ve ölçeklenebilirlik gibi kalite parametrelerinin değerlendirilebilmesi için 20 eş zamanlı kullanıcı ve kullanıcı başına maksimum 1sn cevap süresi temel alınarak yük testi yapılması gerektiği kararlaştırılmıştır.

Mimari Belirleme aktivitesinden sonra *Ontoloji Modelleme* aktivitesi başlamıştır. İlk seviye kavramsallaştırma çalışmalarından sonra, geliştirim takımı literatürdeki lojistik ontolojilerini araştırmaya başlamıştır. Bu çalışmalardan birtanesi askeri taşıma alanında farklı taşıma türlerini planlamayı amaçlamıştır (Becker and Smith, 1997). Fakat bu çalışma lojistik alanını derinlemesine incelememiş, daha çok planlama ve zamanlamaya odaklanmıştır. Lian et al. (2007) çalışmasında bir kurumun lojistik süreçlerinde meydana gelen hareketler ve olaylar

tanımlanmış, fakat taşıma süreci detaylandırılmamıştır. Hoxha et al. (2010), Scheuermann and Hoxha (2012) ve Preist et al. (2005)' in çalışmalarında ise lojistik servislerinin resmi gösterimi olan bir OWL ontolojisi tanımlanmış ve servis tabanlı lojistik operasyonları için OWL-S kullanımı örneklendirilmiştir. iCargo[1] ve CASSANDRA[2] projeleri farklı veri kaynaklarından gelen verileri tümleştirmeye çalışmış ve bu veri üzerinde taşıma risk değerlendirilmesi ve enerji korunması gibi gereksinimler gerçekleştirilmiştir. Fakat, geliştirilen ontolojiler genel kullanıma açılmamıştır. Tüm bu ontolojiler konteyner taşıma sürecine odaklanmadığından dolayı direk olarak bizim çalışmamızda kullanılmaya uygun değildir. Bu yüzden geliştirim takımı gerekli ontolojileri sıfırdan üretmeye karar vermiştir. Bu bağlamda liman ve acente kaynaklarını temsil eden ontolojiler üretilmiştir.



Şekil 3.3. İterasyon 1'e ait mimari

Bağlı Veri Üretme aktivitesinde veri kaynakları bir RDB2RDF aracı kullanılarak üretilen ontolojilere göre RDF formatına dönüştürülmüş ve bir sunucuda yayınlanmıştır. Bu dönüşümler Sequeda et al. (2012) çalışmasında

[1] http://i-cargo.eu/ (Erişim tarihi: 24 Mart 2015)

[2] http://www.cassandra-project.eu/ (Erişim tarihi: 24 Mart 2015)

önerilen eşleşme desenleri uygulanarak gerçekleştirilmiştir. Bağlı veri üretiminden hemen sonra geliştirim takımı *Bağ Oluşturma* aktivitesine geçmiştir (kaynakları tek tek test etmeden). Acente ve liman sistemlerinde konteyner, rezervasyon ve konşimento numarası gibi ortak ve tek alanlar bulunduğundan dolayı bağ oluşturma aktivitesinde otomatik bir bağ oluşturma aracı kullanılmamıştır. Bu kaynakların bağlanması ontolojik seviyede URIlerinin aynı yapılması ile gerçekleştirilmiştir.

Mimari Belirleme aktivitesinde temel veri getirme kriteri olarak birleştirilmiş sorguların işletilme süresi seçildiğinden dolayı geliştirim takımı bağ oluşturma aktivitesinden sonra, *Doğrulama ve Onaylama* aktivitesinde üretilen veriyi test etmek için bu iterasyonda ele alınan UGKların yeterlilik soruları üzerinde yeniden çalışmaya başlamış, var olanları revize etmiş ve yeni sorular eklemiştir. Örneğin; Bu iterasyonda Şekil 3.1'de detayları görülen UGK ele alınmış, yeterlilik soruları revize edilip, yeni sorular eklenerek Şekil 3.4' teki hale getirilmiştir. Yeterlilik sorularının genişletilmesinin temel amacı oluşturulacak olan testlerin geliştirilen ontolojinin doğruluğunu ve yeterliğini test edebilmek için gerekli soruları sormasını sağlamaktır. Daha sonra bu yeterlilik soruları SPARQL sorgularına çevrilmiş. Fakat, bu çevrim işlemi sırasında sorguları yazabilmek için ontolojide değişiklikler yapılması gerekmiştir. Şekil 3.5' te iterasyon 1' e ait SPARQL sorgusu örneği görülmektedir. Bu örnekteki sorgunun yazılabilmesi için ontolojiye ve çalıştırılabilmesi için R2RML eşlemelerine "yapildigiYer" ve "aciklama" ilişkilerinin eklenmesi gerekmiştir. Bu düzenlemeler yapıldıktan sonra, oluşturulan SPARQL sorguları üretilen bağlı veri üzerinde çalıştırılmıştır. Ne yazık ki, mimari daha önce *Mimari Belirleme* aktivitesine belirlenen veri getirme kriterlerini sağlayamamıştır. Geliştirim takımı sorguların işletim süresini Ultrawrap[3], D2RQ[4] ve Oracle Spatial and Graph[5] gibi farklı araçlar kullanarak iyileştirmeye çalışmıştır. Ayrıca ontoloji modelleri karmaşık eşlemeleri kolaylaştırabilmek için değiştirilmiştir. Fakat, sorgu performansı beklenilen 0.5 sn değerine hala yaklaşamamıştır ve ontolojilerin yapısı alan bakış açısından gerçeği yansıtmayan bir duruma gelmeye başlamıştır. Bu yüzden geliştirim takımı bu iterasyonu sonlandırmaya ve yeni bir iterasyona başlamaya karar vermiştir.

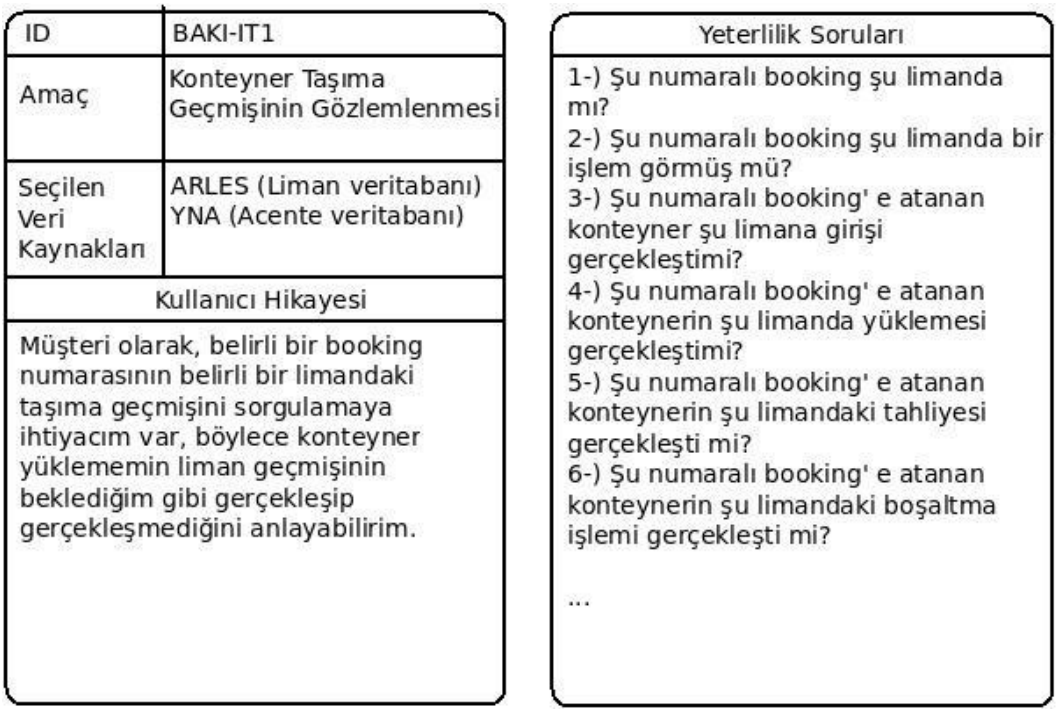
3.1.1 İterasyon 1' in değerlendirilmesi

[3] http://capsenta.com/#section-ultrawrap (Erişim tarihi: 18 Mart 2015)

[4] http://d2rq.org/ (Erişim tarihi: 18 Mart 2015)

[5] http://www.oracle.com/technetwork/database/options/spatialandgraph/overview/index.html (Erişim tarihi: 25 Mart 2015)

- Bağlı veri geliştirme sürecini öğrenmek zaman aldığından ve aktivitelerin düzgün bir biçimde akmasını engellediğinden dolayı geliştirim takımına ontoloji modelleme, R2RML eşlemeleri, SPARQL gibi bağlı veri teknolojileri ile ilgili eğitim planlanmalıdır.



Şekil 3.4. İterasyon 1 sonunda örnek UGK'nın evrimi

- Üretilen bağlı verinin testi Bağ Oluşturma aktivitesinden sonraya bırakıldığından bazı SPARQL sorgularını oluşturabilmek için ontolojilerde değişiklik yapılması gerekmiştir. Ayrıca bu değişiklik R2RML eşlemelerini de etkilemektedir. Bu durum tekrar Ontoloji Modelleme aktivitesine geri dönülmesine neden olmuştur. Bu yüzden, genel bir kural olarak test durumlarının oluşturulması ve testlerin çalıştırılması gerekli bilgi hazır olduğunda yapılmalıdır. Bu durumda, *Ontoloji Modelleme* aktivitesi sonunda yeterlilik soruları revize edilebilir ve SPARQL sorguları oluşturulabilir. Ayrıca, bu test durumları O*ntoloji Modelleme* aktivitesinde tek bir veri kaynağını ilgilendirenler, birden fazla veri kaynağını ilgilendirenler olarak gruplanmalı, *Bağlı Veri Üretme* ve *Bağ Oluşturma* aktivitelerinde ilgili test durumları çalıştırılmalıdır. Bu sebeple, yöntem bu gözlemlere göre revize edilmiştir.

- Mimari belirlemeyi ve belirlenen mimarinin değerlendirilmesini geliştirim yaşam döngüsü içerisinde yapmanın iyi bir fikir olduğu görüldü.

YETERLİLİK SORUSU

Şu numaralı booking' in şu limanda başından geçen hareketlerin açıklaması ve tarihi nedir?

SPARQL SORGUSU

```
SELECT ?hareket ?aciklama ?tarih
WHERE {
  ?tarihce   genel:aitOlduguBooking       <http://.../genel/booking/B10047400>.
  ?tarihce   genel:islemGorenKonteyner    <http://.../genel/ekipman/ARKU4320533>.
  ?tarihce   genel:yapildigiYer           "Liman".
  ?tarihce   arles:yapildigiLiman          ?liman.
  ?liman     genel:adi                    "IZMIR ALSANCAK".
  ?hareket   genel:bagliOlduguTarihce     ?tarihce.
  ?hareket   rdf:type                     genel:KonteynerHareketi.
  ?hareket   genel:aciklama               ?aciklama.
  ?hareket   genel:hareketTarihi          ?tarih.
}
```

DOĞRULAMA VE ONAYLAMA AKTİVİTESİ

Şekil 3.5. İterasyon 1' e ait örnek SPARQL sorgusu

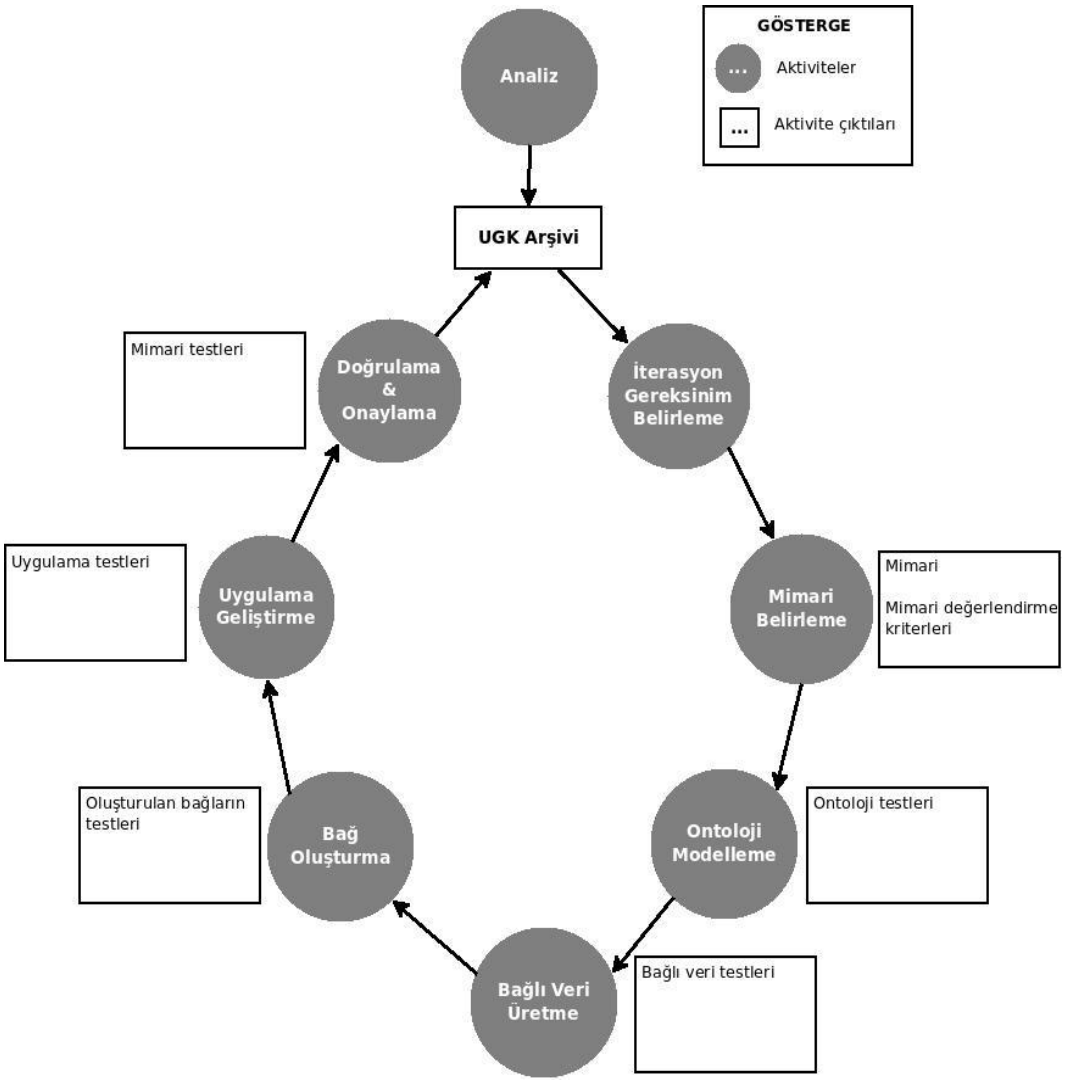
3.2 İterasyon 2' ye Genel Bakış

Uygulamanın ilk iterasyonu sonucunda Şekil 3.2' de görülen yöntem alınan derslere göre revize edilmiş, çevik analiz ve önce test yaklaşımlarından esinlenilerek Şekil 3.6' daki hale gelmiştir. Uygulamanın bu iterasyonu yöntemin bu revize edilmiş versiyonu uygulanarak gerçekleştirilmiştir. Sorgu birleştirme deseninin kısıtları görüldükten sonra geliştirim takımı bu iterasyona bağlı veri mimarisinin değiştirilmesi gibi net bir amaç ile başlamıştır. Geliştirim takımı bu iterasyona gözlemleme amacı için depo veri kaynağını içeren yeni bir UGK dahil etmiştir. Şekil 3.7' de depo kaynağını içeren bu UGK görülmektedir. Bu seçimin amacı önerilen yöntemi edinilen derslere göre yeniden uygulamak ve deneyimi arttırmaktır.

Mimari Belirleme aktivitesinde geliştirim takımı gözlemlenen performans problemlerini çözmek için *Tarama* desenini kullanarak kurumun iç veri kaynaklarını tarayan bir mimari oluşturmaya karar verilmiştir. Bu noktada tarama deseninin eskiyen veri sorununu çözmek bir problem haline gelmiştir. Bu amaçla geliştirim takımı bir Değişen Verileri Yakalama (CDC - Change Data Capture)[6] aracı kullanmaya ve CDC aracından gelen her mesajı ontolojilerin RDF

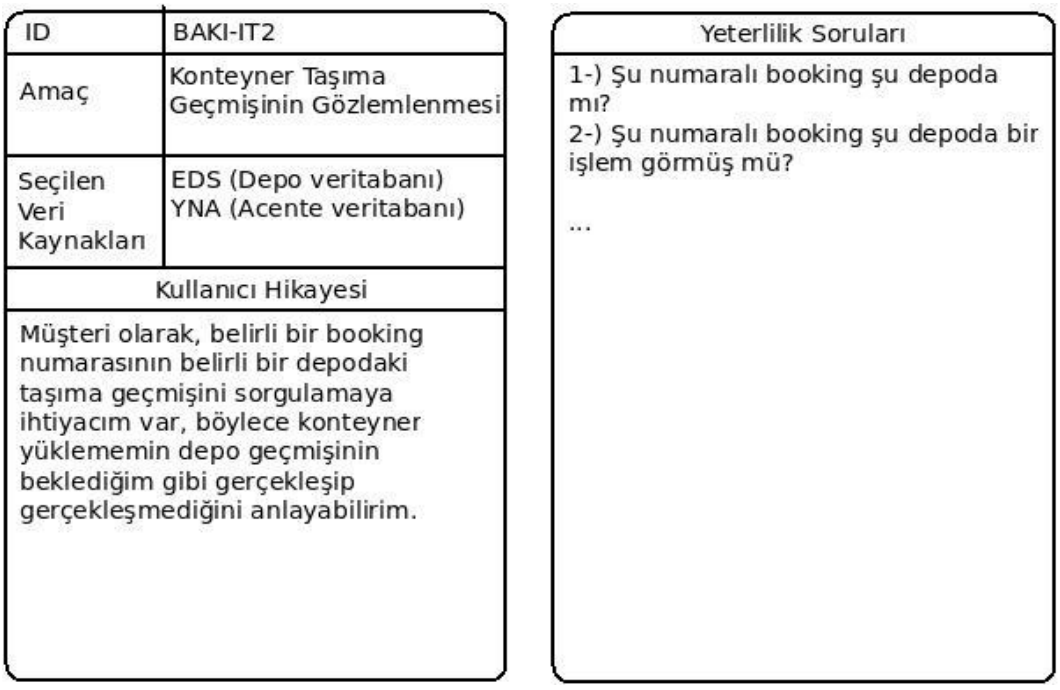
[6] http://en.wikipedia.org/wiki/Change_data_capture (Erişim tarihi: 26 Mart 2015)

örneklerine dönüştürmeye karar vermiştir. Geliştirim takımı yüksek performanslı ticari bir CDC aracının kullanılması üzerinde anlaşmış ve CDC aracından gelen mesajların dönüştürülmesi için bir Bütünleştirme Birimi geliştirilmesini planlamıştır. Ayrıca, CDC aracını ve bütünleştirme birimini senkronize edebilmek için açık kaynaklı bir mesaj kuyruğu kullanılmasına karar verilmiştir. Dönüştürülen RDF verilerinin saklanması için ise merkezi bir RDF saklayıcı kullanılmasına karar verilmiştir. Şekil 3.8' de bu iterasyona ait genel mimari görülmektedir. Bu iterasyon için değer- lendirme kriterleri ile ilgili bir değişiklik yapılmamıştır. Bu yüzden, mimari değerlendirme test planları aynı kalmıştır.



Şekil 3.6 İterasyon 2' de uygulanan yöntem

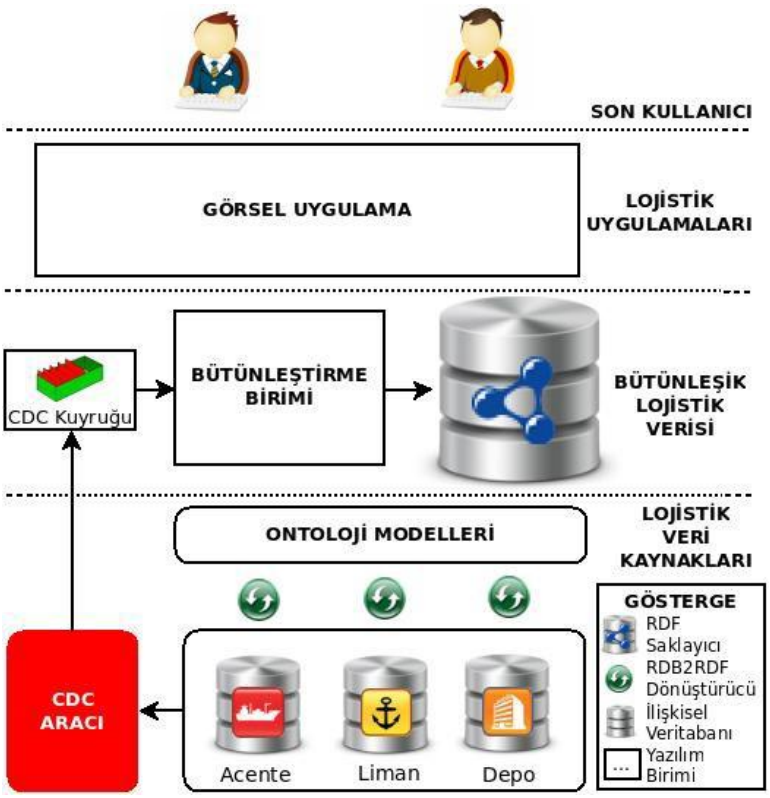
Ontoloji Modelleme aktivitesinde geliştirim takımı bir önceki iterasyonda elde ettiği deneyimleri göz önünde bulundurarak depo ontolojisini tanımlamaya başlamıştır. Bu yüzden bu aktivitenin sonunda test durumları üretildi. Geliştirim takımı geliştirilen ontolojinin yeterliliğini ve doğruluğunu test edebilmek için seçilen UGK' nın yeterlilik sorularını revize etti, genişletti ve bu sorguların SPARQL versiyonlarını yazmaya çalıştı. Bu çalışmalar sonucunda, bu iterasyon için seçilen UGK kartı Şekil 3.9' daki hale gelmiştir. Yeterlilik sorularının başarılı bir şekilde SPARQL sorgusuna dönüştürülmesinden sonra geliştirim takımı *Bağlı Veri Üretme* aktivitesine geçmiştir. Bu aktivitede depo veri kaynağının R2RML eşlemeleri yapılmış ve oluşturulan tüm RDF verisi merkezi bir RDF saklayıcıda toplanmıştır.Daha sonra üretilen RDF verisi Ontoloji Modelleme aktivitesinde üretilen SPARQL sorguları ile oluşturulan testler kullanılarak test



Şekil 3.7. İterasyon 2' ye ait örnek UGK

edilmiştir. Bu noktada oluşturulan verinin daha detaylı test edilebilmesi için alan uzmanlarının da yardımıyla yeterlilik soruları ve SPARQL sorguları detaylandırılmıştır. Örneğin; Ontoloji Modelleme aktivitesinde "Şu numaralı rezervasyon' a atanan konteyner şu depoda hasar gördü mü?" yeterlilik sorusu SPARQL sorgusuna çevirilmişti. Bağlı Veri Üretme aktivitesinde ise bu sorgu belirli bir rezervasyon numarası, depo ve bu rezervasyon içerisindeki bir konteyner için olacak şekilde düzenlenmiş ve hasar yeri, hasar tipi gibi bilgiler de eklenerek daha detaylı hale getirilmiştir. Aynı zamanda ilgili yeterlilik sorusu

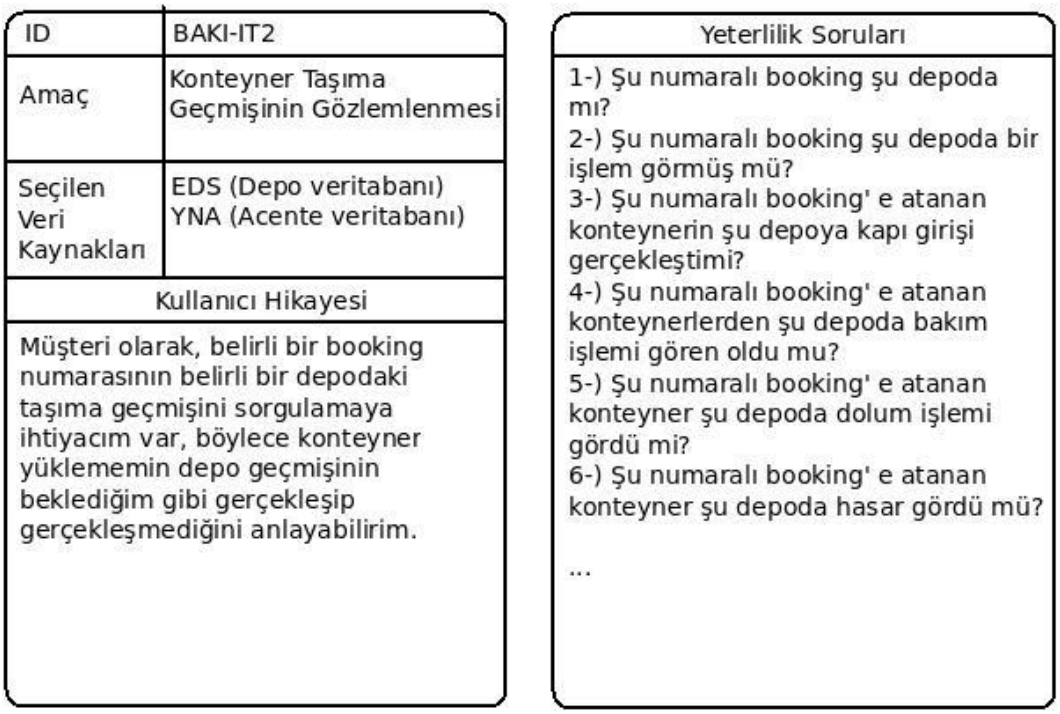
da bu detaylar eklenerek genişletilmiştir. Bu yeterlilik sorusu ve SPARQL sorgusunun ontoloji modelleme ve bağlı veri üretme aktiviteleri arasındaki evrimi şekil 3.10' da gösterilmektedir. Alan uzmanlarından belirlenen rezervasyon, depo ve bu rezervasyona ait konteyner için test verisi istendi ve bu veriler beklenen test sonuçları olarak tanımlanmıştır. Daha sonra detay landırılmış SPARQL sorgusunu



Şekil 3.8. İterasyon 2'ye ait mimari

çalıştıran test yazılmış ve sorgu sonucunda elde edilen verinin beklenen veri ile aynı olup olmadığı doğrulanmıştır. Bu test işlemi diğer yeterlilik soruları için de gerçekleştirilmiş ve oluşturulan testler sürekli tümleşim ortamına eklenmiştir. Üretilen depo verisi test edildikten sonra, acente ve liman veri kaynaklarıda daha önce tanımlanmış olan R2RML eşlemelerine göre RDF'e dönüştürülmüş ve merkezi RDF saklayıcıda saklanmıştır. Eskiyen veri problemini çözmek için geliştirim takımı bütünleştirme birimini geliştirmeye başlamıştır. Bu birimde, CDC aracı kurumun iç veri kaynakları olan ilişkisel veritabanlarında gerçekleşen değişiklikleri eş zamanlı olarak sisteme yük bindirmeden karşılar ve her değişikliği önceden tanımlı bir XML formatı kullanarak bir java mesaj kuyruğuna (CDC Kuyruğu) gönderir. Bütünleştirme birimi, CDC kuyruğunda bulunan XML formatındaki değişiklik mesajlarını tüketmekten sorumludur. Değişiklik mesajları RDF üçlülerine dönüştürülür ve RDF saklayıcı bu üçlülere göre güncellenir.

Fakat, veri kaynakları günde yaklaşık üç milyon değişiklik mesajı ürettiğinden dolayı bu birimin ölçeklenebilir bir ortamda ve eş zamanlı olarak çalışması



Şekil 3.9. İterasyon 2 ontoloji modelleme aktivitesi sonunda örnek UGK'nın evrimi

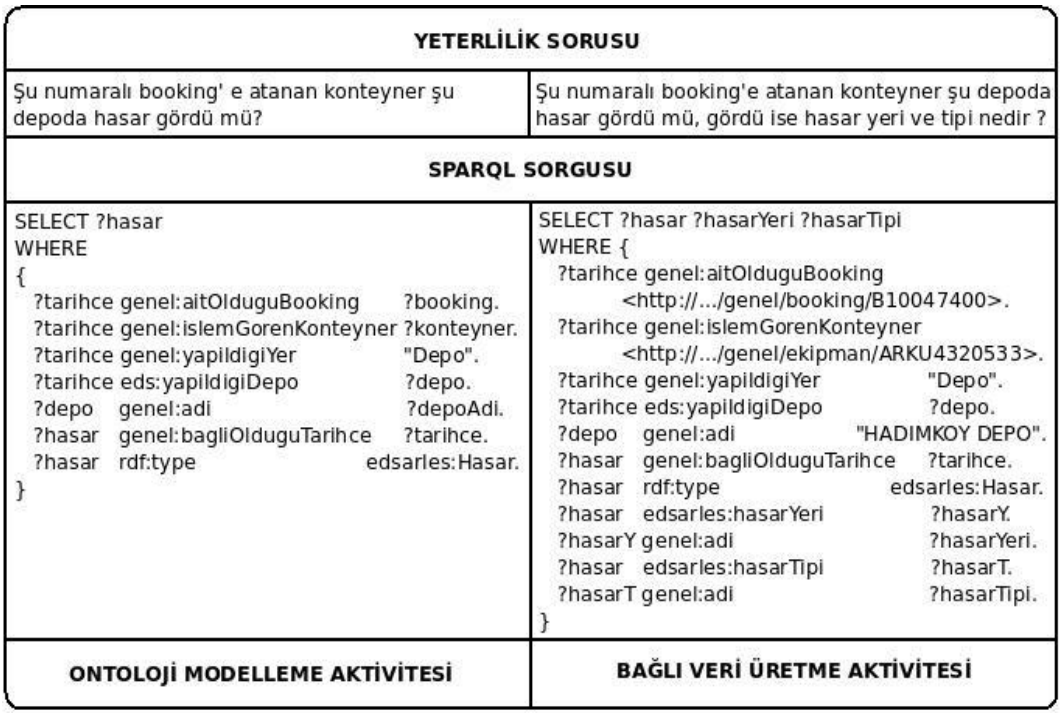
gerektiği ortaya çıkmıştır. Bu mesaj trafiğini karşılayabilmek için geliştirim takımı AKKA[7] altyapısını kullanmaya karar verdi. AKKA eşzamanlılık, paralellik ve hata toleransını yönetemek için son derece ölçeklenebilir olay güdümlü bir etmen sistemidir. Bütünleştirme birimindeki her AKKA aktörü (Güncelleme Aktörü) mesaj kuyruğundaki XML mesajlarını okur, RDF üçlülerine dönüştürür ve RDF saklayıcıyı bu değişikliklere göre günceller. Şekil 3.11' da bütünleştirme biriminin AKKA altyapısı gösterilmektedir.

Bağ Oluşturma aktivitesinde, bağlar ontolojik seviyede oluşturulduğu için bir işlem yapmaya gerek yoktur. Bu yüzden, sadece üretilen depo RDF verisi ve daha önce üretilmiş olan RDF verileri arasındaki bağlar test edilmiştir.

Üretilen Bağlı verinin başarıyla test edilmesinden sonra, geliştirim takımı konteyner taşımasını gözlemleyebilmek için görsel bir arayüz geliştirmeye başlamıştır. Müşteri ile yapılan tartımalar sonucunda, geliştirim takımı bir görsel

[7] http://akka.io/ (Erişim tarihi: 27 Mart 2015)

tasarım belirlemiş, uygun görselleştirme tekniğini ve aracını seçmiştir. Fakat, bu arayüzün geliştirilmesi çok uzun sürmüş ve planlanan iterasyon süresi içerisinde tamamlanamamıştır.

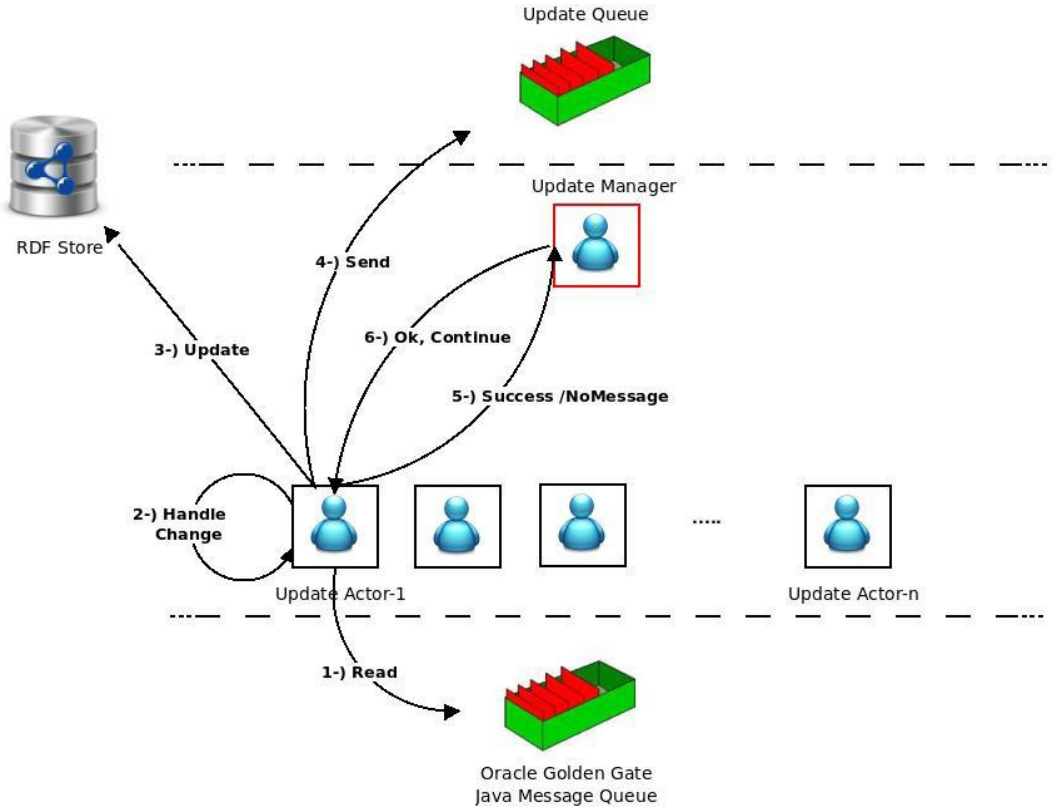


Şekil 3.10. İterasyon 2 örnek yeterlilik sorusu ve SPARQL sorgusunun aktiviteler arası evrimi

3.2.1 İterasyon 2' nin değerlendirilmesi

- Ontoloji Modelleme, Bağlı Veri Üretme ve Bağ Oluşturma aktivitelerinde test durumlarının üretilmesinin ve testlerin yapılmasının iyi bir fikir olduğu görüldü.

- Görsel bir arayüzün bağlı veri üretildikten sonra gerçekleştirilmesi ürün teslimini geciktirmekte ve proje taraflarının motivasyonunu düşürmektedir. Bu yüzden, uygulama geliştirme süreci bağlı veri oluşturma süreci ile paralel olarak işletilmelidir. Bu sebeple, bu iterasyon sonucunda önerilen yönteme Uygulama Geliştirme Yaşam Döngüsü eklenmiştir.

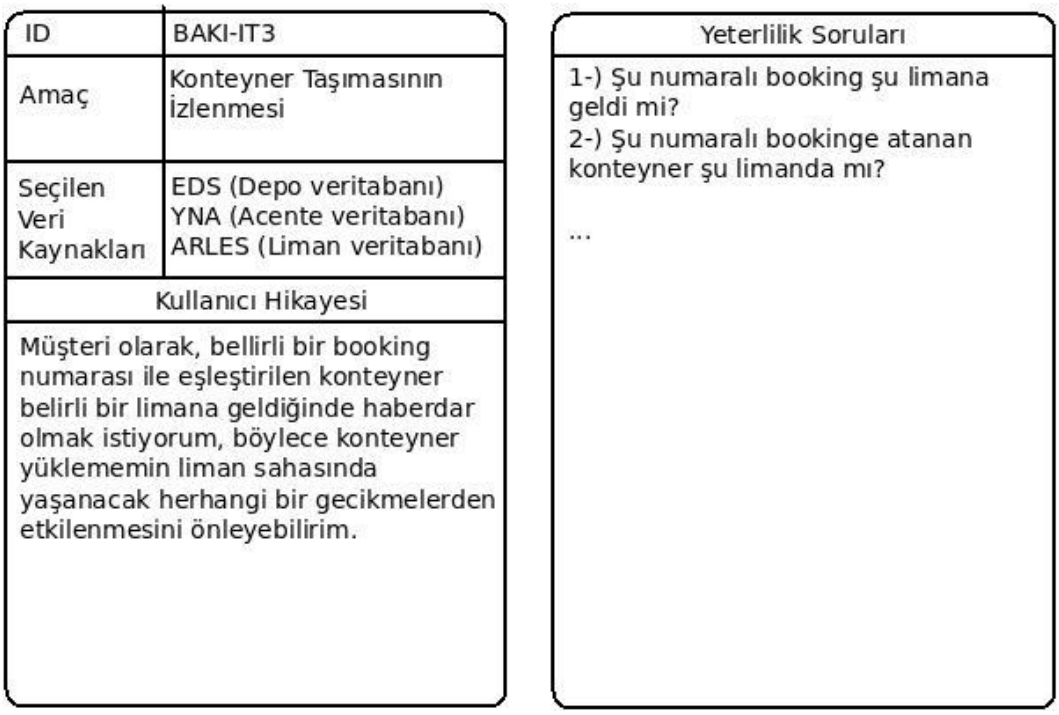


Şekil 3.11. Bütünleştirme Birimi AKKA Altyapısı

3.3 İterasyon 3' e Genel Bakış

Önceki iterasyonda alınan dersler sonunda yöntem Şekil 4.1' deki nihai haline gelmiştir. Geliştirim takımı üçünü iterasyonu bu yöntemi uygulayarak gerçekleştirmiştir. Bu iterasyona geliştirim takımı izleme amacı için bir UGK eklemiştir. Ayrıca, müşteri konteyner taşımasının kara taşıma gibi genellikle 3. parti dış şirketler tarafından gerçekleştirilen bölümlerini de izlemek istediği için bu iterasyona kara taşıma dış kaynağına ait bir UGK da eklenmiştir. Şekil 3.12' de izleme amacına ait UGK görülmektedir. *Mimari Belirleme* aktivitesinde geliştirim takımı ARKAS holdingin iç veri kaynakları için Tarama desenini, dış kara taşıma şirketi için ise Sorgu Birleştirme desenini kullanan hibrit bir mimari oluşturmaya karar vermiştir. Dış veri kaynağı için sorgu birleştirme deseninin kullanılma sebebi müşterinin bu kaynağa ait olayların izlenmesinin iş açısından öncelikli olmadığını belirtmesidir. izleme olaylarını mimari seviyesinde yakalayabilmek için geliştirim takımı ayrı bir mesaj kuyruğu mekanizması ile yeni bir AKKA organizasyonu kullanmayı düşündü. RDF formatındaki değişiklikler yeni AKKA organizasyonuna bir önceki iterasyonda gerçekleştirilen AKKA organizasyonu tarafından gönderilir ve bu yeni organizasyondaki aktörler (İzleme Aktörü) önceden tanımlanmış izleme kuralları ile gelen RDF verisini karşılaştırarak

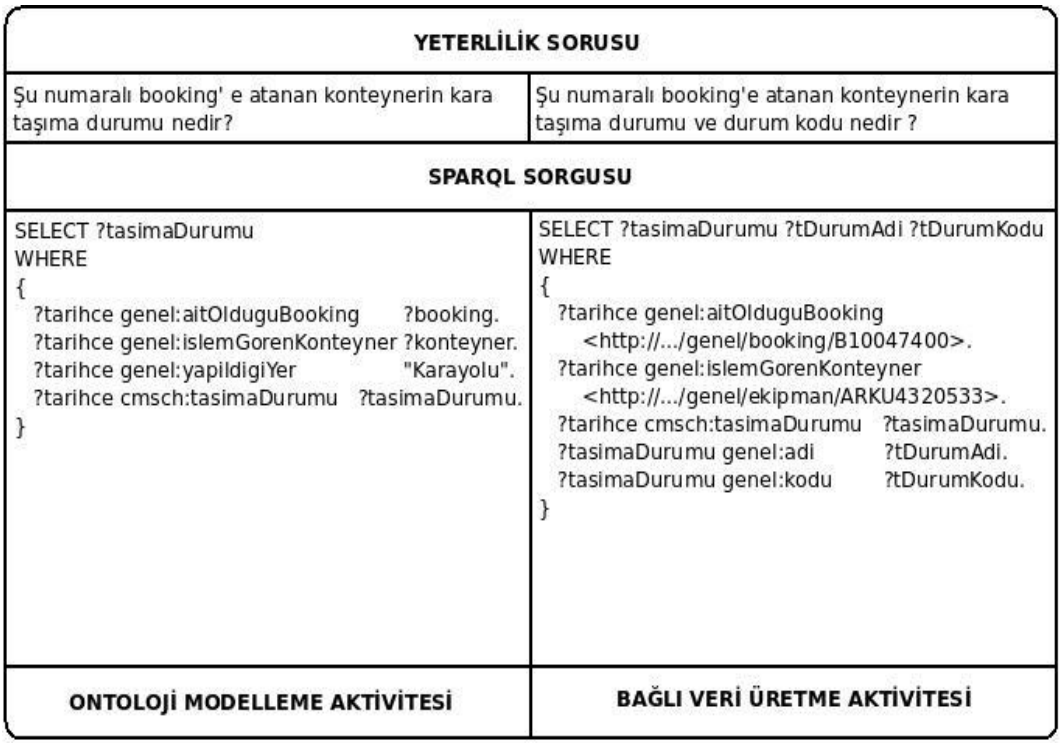
izleme olaylarını yakalar. Geliştirim takımı izleme özelliğini test edebilmek için günlük maksimum mesaj sayısı ile yük testi yapmayı planlamıştır ve bu yük testinin boyutuna karar verebilmek için gerçek sistemdeki CDC olaylarını gözlemlemeye karar vermiştir.



Şekil 3.12. İterasyon 3' e ait örnek UGK

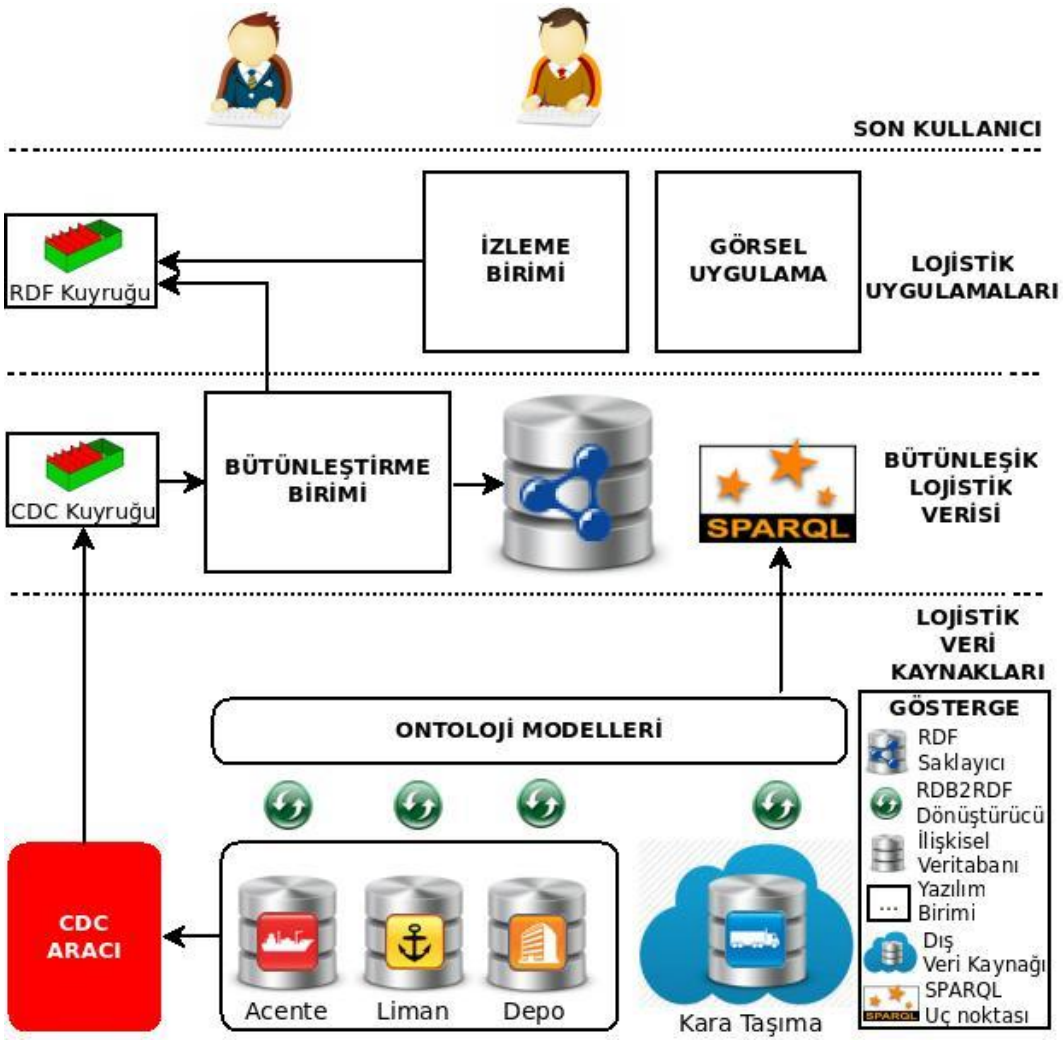
İGB aktivitesinden sonra, geliştirim takımı *Mimari Belirleme* aktivitesi ile paralel olarak *İlk Seviye Görsel Uygulama Geliştirme* aktivitesinde izleme amacı için görsel bir arayüz geliştirmeye başlamış ve müşteri ile bu arayüzler üzerinde çalışmıştır. *Ontoloji Modelleme* aktivitesinde, temel, basit ve dış şirketin veritabanı şemasına benzer bir kara taşıma ontolojisi geliştirilmiş ve test durumları üretilmiştir. Ayrıca ontoloji modelleme takımı, *Yapay Veri Üretme* aktivitesinde uygulama geliştiricilerle birlikte çalışarak paralelde yapay verinin üretilmesine yardımcı olmuştur. *Bağlı Veri Geliştirme* yaşam döngüsünde, *Bağlı Veri Üretme* aktivitesi devam ederken *Uygulama Geliştirme* yaşam döngüsünde uygulama geliştiriciler görsel arayüzü bu yapay veri ile bütünleştirerek çalışmalarına devam etmiştir. Ayrıca, arayüz uygulamasına görsel kabul testleri yapılarak bu testler sürekli tümleşim ortamına eklenmiştir. *Bağlı Veri Üretme* aktivitesinde kara taşıma kaynağının R2RML eşlemeleri yapılmış ve şirketin RDF verisi bir uç nokta aracılığıyla yayınlanmıştır. Ayrıca, üretilen bağlı veri daha önce üretilmiş olan test durumları ile test edilmiştir ve bulunan hatalar düzeltilmiştir. Bu amaçla ontoloji modelleme aşamasında üretilen test durumları ve yeterlilik soruları genişletilmiş ve

detaylandırılmıştır. Şekil 3.13' de yeterlilik sorularının ve SPARQL sorgularının bu evrimi gösterilmektedir. Bağlı veri üretiminden sonra, uygulama geliştirim takımı görsel uygulamayı üretilen gerçek bağlı veri ile bütünleştirmiştir ve görsel kabul testlerini güncellemiştir.



Şekil 3.13. İterasyon 3 örnek yeterlilik sorusu ve SPARQL sorgusunun aktiviteler arası evrimi

Bu iterasyonda uygulama ilgili taşıma olayları oluştuğunda kullanıcıyı bilgilendirme ve dış kara taşıma kaynağına ait taşıma geçmişini daha önce gerçekleştirilmiş olan arayüze ekleme gereksinimleri etrafında şekillenmiştir. Geliştirim takımı taşıma olaylarını yakalayabilmek için yeni bir AKKA organizasyonu içeren *İzleme Birimini* geliştirmeye başladı. İzleme birimi, Bütünleştirme birimi tarafında RDF' e dönüştürülen tüm taşıma yaşam döngüsü olaylarını analiz ederek ilgili olayları bulmaktan sorumludur. Bu birimde, izleme aktörleri oluşan değişiklikten etkilenen taşıma kurallarını bulur ve bunları kullanıcı arayüzüne gönderir. Kullanıcı arayüzü ilgili olayları anlık olarak kullanıcıya bildirir. Her değişiklik birden fazla taşıma kuralı ile ilgili olabilir. Uygulamaya ait son mimari Şekil 3.14' de gösterilmektedir. İterasyon sonunda geliştirilen uygulama *Doğrulama ve Onaylama* aktivitesinde doğrulandı. Kabul ve yük testleri müşteri ve ürün sahibi ile birlikte değerlendirildi ve iterasyon başarılı bir şekilde sonlandırıldı.



Şekil 3.14. Son uygulama mimarisi

3.3.1 İterasyon 3' ün değerlendirilmesi

- Dış veri kaynaklarının entegrasyonu mimari kararları etkilediğinden ve bağlı veri mimarisini yeni bir şirkete tanıtmak için çok fazla efor gerektiğinden dolayı erken iterasyonlarda yapılmalıdır.

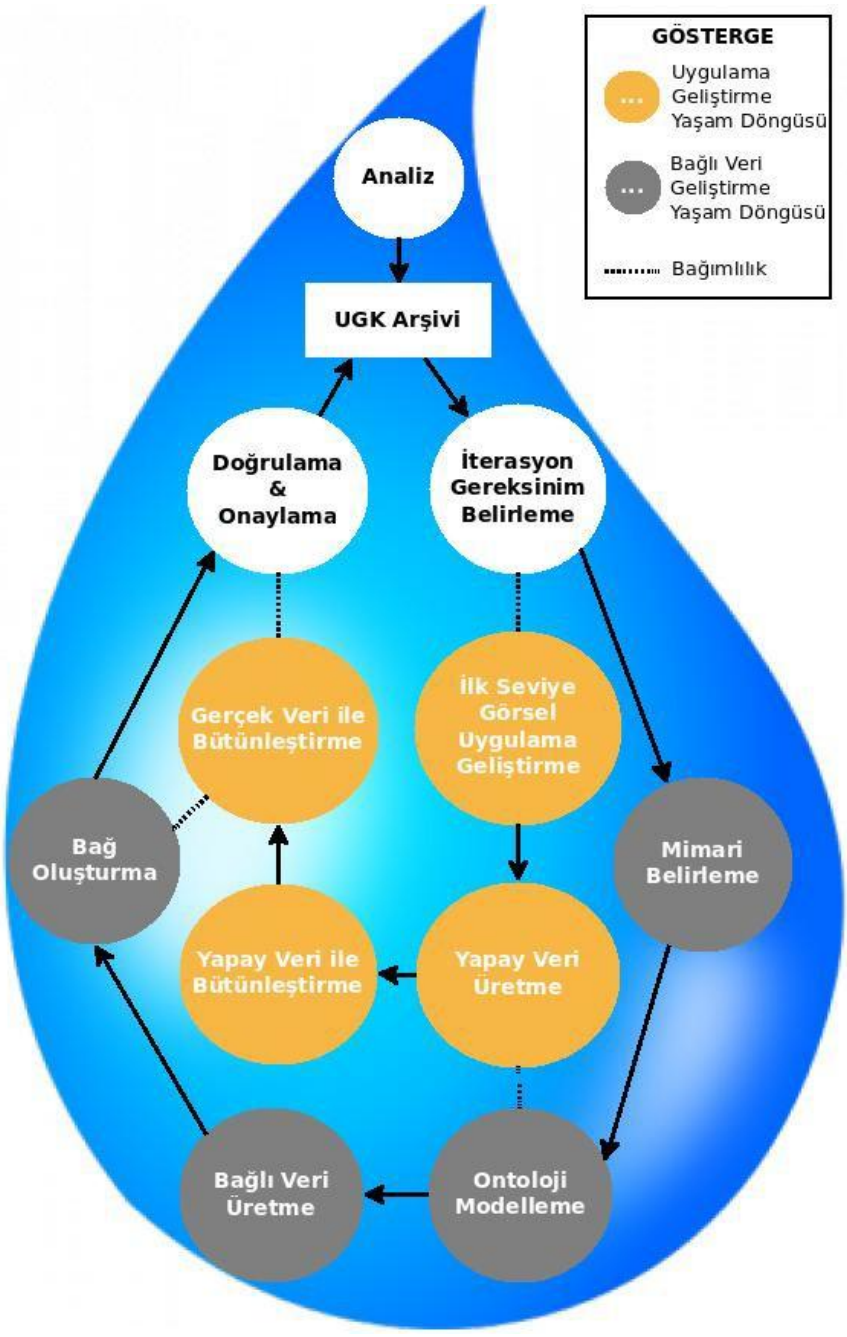
4. BLOB (A Methodology to Bring Agility into Linked Open Data Development for Businesses) YÖNTEMİ

Çevik yöntemlerin yazılım geliştirmede değişen iş ve mimari gereksinimler ile başa çıkmak için iyi bir yaklaşım olduğu anlaşılmıştır. Sadece yazılım geliştirme için değil, aynı zamanda Collier' in (2011) de bahsettiği gibi iş zekası ve analizi uygulamaları içinde faydalı bir yaklaşımdır. "A Methodology to **B**ring Agility into **L**inked **O**pen Data Development for **B**usinesses" (BLOB) metodolojisi, çevik yazılım geliştirme ve çevik analiz yaklaşımlarından SCRUM (Schwaber, 2004), XP (Beck and Andres, 2004), özellik/hikaye odaklı geliştirim, önce test yaklaşımı (Fraser et al., 2003), müşteri katılımı ve sürekli tümleşim (Campos et al., 2014) gibi pratikleri almıştır. SCRUM altyapısı, projeyi kendinden organize takım dinamikleri ve yinelemeli bir yaşam döngüsü ile yönetmek için kullanılmaktadır. Bağlı veri ortamı yeni verilerin, bağların oluşması ve ontolojilerdeki değişiklikler ile sürekli değişmektedir. Bu oldukça dinamik ortam iş ve mimari gereksinimlerinin de değişmesine sebep olabilmektedir. Bu yüzden, çevik pratikler yöntemi bağlı veri uygulama geliştirme ortamının dinamik yapısına uyumlu hale getirmektedir.

Bağlı veri geliştirme, kritik araç ve mimari desenlerinin sürekli olarak evrildiği genç bir alandır. Ayrıca, iş gereksinimleri ve/veya bağlı veri ortamındaki değişiklikler ilk seviye mimari varsayımları etkileyebilmektedir. Bu sebeple, geliştirim takımı, geliştirim süresince mimarinin evrimini ve performansını gözlemlemelidir. Çevik mimari (Brown et al., 2010) yaklaşımında mimari değerlendirmenin süreç boyunca gözlemlenmesi gerektiği açık bir biçimde anlatılmaktadır. Çevik mimari yaklaşımında da anlatıldığı gibi, önerilen yöntem mimari modelini ve değerlendirme kriterlerini her iterasyonun başında belirler ve her iterasyon sonunda bu mimari varsayımları doğrular.

Bağlı veri geliştirme, çoğu yöntemde (Villazon- Terrazas et al., 2011; Hyland and Wood, 2011; Hausenblas, 2011; Auer et al., 2012) bahsedildiği gibi bazı belirli adımları içermektedir. Ayrıca, ontoloji modelleme literatürü önerilen ve kullanılan bir çok yöntem ile uzun bir geçmişe sahiptir (Fernandez- Lopez et al., 1997; Noy and McGuinness, 2001; Pinto et al., 2004; Nicola et al., 2009; Suarez-Figueroa et al., 2012). Önerilen yöntem, bağlı veri geliştirme yöntemlerinden ve ontoloji modelleme yaklaşımlarından gerekli adımları alarak bunları yinelemeli bir yaşam döngüsü içerisinde çevik yöntem pratikleri ile birleştirmiştir. Yöntem bir yıldan fazla süren uygulama geliştiriminin dört iterasyonu boyunca evrilmiştir. Iterasyonlar boyunca yaşam döngüsündeki test

durumlarını ve test noktalarını belirleyerek önce test yaklaşımını gerçekleştirmeye ihtiyaç olduğu açıkça görülmüştür. Diğer bir kritik gözlem ise, *Bağlı Veri Geliştirme* ve *Uygulama Geliştirme* yaşam döngülerinin paralel olarak işletilmesi gerektiğidir. Bağlı Veri Geliştirme yaşam döngüsü veri ile ilişkili tüm adımları içermektedir. Diğer yandan, Uygulama Geliştirme yaşam döngüsü, üretilen bağlı veri üzerinde yazılım geliştirme aktivitelerini içermektedir. Şekil 4.1' de yöntemin iç döngüsü Uygulama Geliştirme yaşam döngüsünü, dıştaki ise Bağlı Veri Geliştirme yaşam döngüsünü göstermektedir.



Şekil 4.1. BLOB yöntemi

4.1 Analiz

Analiz aktivitesinde ürün gereksinimleri başlıca ürün sahibi tarafından müşteri rolü(leri), geliştirim takımı üyesi(leri) gibi gerekli proje taraflarının da katılımı ile belirlenir. Analiz aktivitesinin ilk adımı olarak, uygulamanın temel amaçları ve her amacı gerçekleştirmek için gerekli olan kullanıcı hikayeleri tanımlanır. Bağlı veri perspektifinden bakıldığında analiz aktivitesinin kritik noktası, kullanıcı hikayeleri için gerekli olan veri kaynaklarının belirlenmesidir. Bu veri kaynakları geliştirim takımı tarafından belirlenir ve "Uygulama Gereksinin Kartı"(UGK) içerisine eklenir. Bu aşamanın klasik kullanıcı hikayesi tanımlamadan ikinci kritik farkı ise UGK kartının yeterlilik soruları bölümüdür. Yeterlilik soruları, ontoloji geliştirme yöntemlerinde geliştirilen ontolojiyi doğrulamak için kullanılan ve iyi bilinen bir yöntemdir. Burada ise yeterlilik soruları veri kaynaklarının bağlı görünümünü düşünmeyi ve bu bağlı veri görünümü üzerinden kullanıcı hikayelerini doğrulamayı sağlar. Kullanıcı hikayeleri, kullanıcı hikayesi test durumlarının başlıca kaynağıdır. Analiz aktivitesi iterasyonların bir parçası değildir, gerekli görüldüğünde işletilebilir. Ürün sahibi iterasyonları gözlemler, müşteri ile sürekli iletişim içindedir ve duruma göre yeni amaçlar tanımlayabilir, var olan amaçları güncelleyebilir ve/veya yeni veya eski bir amaç için yeni UGK(lar) tanımlayablir. Bu UGK(lar), UGK arşivinde toplanır.

UGK' lar her kullanıcı hikayesi için aşağıdaki bölümleri içerecek şekilde hazırlanır;

- *ID*: UGK'nın belirleyicisi.
- *Uygulama Amacı*: Uygulamanın kullanım amacını içerir. Bu amaç, uygulamanın sınırlarını belirlemek için kullanılır.
- *Seçilen Veri Kaynakları*: Bağlı veriye dönüştürülecek ve uygulama tarafından tüketilecek olan veri kaynaklarıdır. Bunlar; ilişkisel veritabanları, dökümanlar, veb sayfaları vb. olabilir.
- *Kullanıcı Hikayesi*: Uygulama amacının belirlenen özelliklerini uygulamak için gerekli olan son kullanıcı bakış açısıyla yazılmış kullanıcı hikayesidir.
- *Yeterlilik Soruları*: Kullanıcı hikayesini doğrulamak için kullanılacak olan sorulardır.

4.2 İterasyon Gereksinim Belirleme

Iterasyon Gereksinim Belirleme (İGB) bir planlama aktivitesidir. UGK arşivinde bulunan kartlar iş değerleri ve içerdikleri veri kaynaklarına göre önceliklendirilir. Buradaki veri kaynağına göre önceliklendirme işlemi bağlı veri ile geliştirme açısından kritik bir adımdır. Eğer birden fazla veri kaynağı ile ilişkili kullanıcı hikayeleri var ise bu durum Bağlı Veri Üretme, Bağ Oluşturma ve ayrıca Mimari Belirleme adımlarını da etkiler. Bağlı veri mimarilerinin temel özellikleri farklı veri kaynaklarını yayınlama ve bütünleştirme kararlarına dayanmaktadır ve birden fazla veri kaynağının sisteme dahil edilmesi temel mimarinin erken iterasyonlarda oluşturulması ve değerlendirilmesi açısından kritiktir. Bu yüzden, İGB aktivitesi sonunda geliştirim takımı iterasyona dahil edilecek olan UGK kartlarını, bu kartların içerdikleri kullanıcı hikayelerini iş ve mimari açısından değerlendirerek kararlaştırır.

4.3 Bağlı Veri Geliştirme Yaşam Döngüsü

4.3.1 Mimari belirleme

Mimari Belirleme aktivitesi çevik mimari için ilk adımdır. Öncelikle, bir önceki aktivitede seçilen UGK kartlarının kullanıcı hikayeleri analiz edilerek uygulamanın gereksinimlerine göre mimari desen(ler) belirlenir. Önce test yaklaşımı açısından, mimari değerlendirme için test planlaması bu aktivitede yapılır. Mimari iki aşama test uygulanarak değerlendirilir. Birinci aşama üretilen bağlı verinin getirilme performansı ile ilgilenirken, diğer aşama ise son uygulama için seçilen performans, ölçeklenebilirlik gibi kalite parametrelerinin (Meier et al., 2008) değerlendirilmesi ile ilgilenir. İlk seviye test Bağlı Veri Üretme ve/veya Bağ Oluşturma aşamasında yapılırken, ikinci seviye test Doğrulama ve Onaylama aktivitesinde yapılır. Bu noktada veri getirme kriterleri, kalite parametreleri ve bunların beklenen sınırları belirlenir ve ilk seviye mimari değerlendirme test planı olarak dökümante edilir.

Ugulama ihtiyaçlarına göre bağlı veri tüketimi için literatürde iyi bilinen üç yaklaşım bulunmaktadır: Anında Çözümleme Deseni, Tarama Deseni ve Sorgu Birleştirme Deseni (Heath and Bizer, 2011). Anında Çözümleme deseni vebi çözümlenebilir URI(ler) içeren dökümanlardan oluşan bir çizge olarak kavramsallaştırır. Böylece, bir uygulama çözümlenebilir URL adresi ile ulaştığı RDF dökümanı üzerinde sorgu çalıştırır ve sorgu çalıştırılması sırasında bu dosya

ayrıştırılarak içerisindeki çözümlenebilir URI bağları takip edilir (Hartig et al., 2009). Tarama Deseninde, veb üzerindeki veri çözümlenebilir URL ler aracılığıyla sürekli taranır, bağlar takip edilir ve bulunan veri yerelde tümleştirilir. Sorgu Birleştirme Deseni ise karmaşık sorguların alt sorgulara bölünmesi ve herbir alt sorgunun ilgili veri kümeleri üzerinde çalıştırılmasına dayanır. Bu desen alt sorguları çalıştırabilmek için veri kümelerine SPARQL uç noktaları üzerinden erişmeye gereksinim duyar.

Tüm bu desenlerin mimari kararları verirken göz önünde bulundurulması gereken avantajları ve dezavantajları bulunmaktadır. Anında Çözümleme deseninde karmaşık işlemler çok yavaştır, çünkü binlerce URI arka planda çözümlenir, fakat eskiyen veri ile asla çalışılmaz. Tarama deseninin temel avantajı performanstır. Uygulamalar yüksek hacimli tümleştirilmiş veriyi diğer desenlerden daha yüksek performans ile kullanabilir. Diğer yandan, bu desenin temel dezavantajı eskiyen veri problemi ve verinin çalışma anında otomatik olarak bağlanmasının karmaşık olmasıdır. Sorgu Birleştirme deseni uygulamaların verinin yerele kopyalanmasına gerek duymadan güncel veri ile çalışmasına olanak sağlar, fakat bu yaklaşımın da temel dezavantajı birden çok veri kümesi üzerinde çalıştırılması gereken parçalar içeren karmaşık sorgulardır.

Ayrıca, literatürde bağlı veri modelleme, yayınlama ve tüketme için çok sayıda bağlı veri tasarım deseni bulunmaktadır (Dodds and Davis, 2012). Geliştirim takımı ya da veri yayıncıları ihtiyaçlarına göre uygun deseni veya bu desenlerin bir karışımını kullanabilirler. Buna rağmen, bu desenlerin seçimi mimari kararları etkileyecek veri tazelik seviyesi, uygulama cevap süresi ve çalışma anında yeni kaynakları keşfetme gibi bir çok farklı faktöre bağlıdır. Geliştirim takımı mimari kararı seçilen kalite parametreleri ve bu faktörlere göre verir.

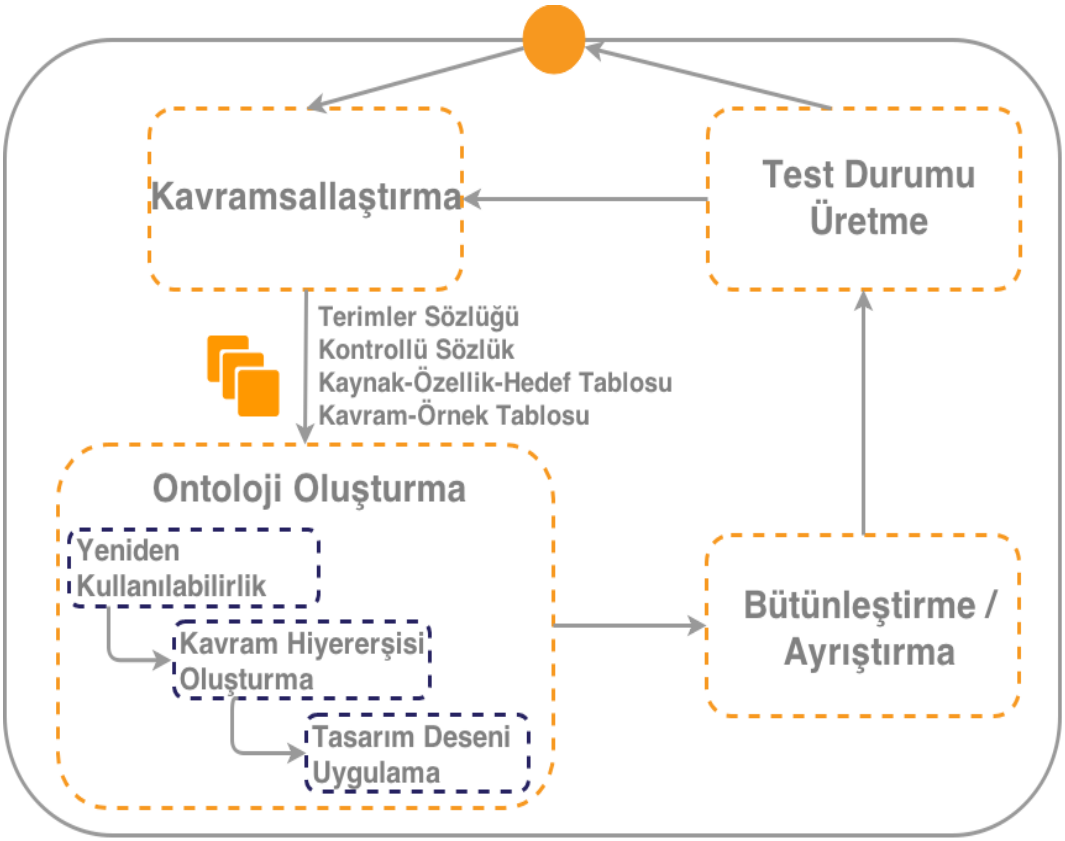
4.3.2 Ontoloji modelleme

Ontoloji modelleme aktivitesinde alandaki kavramlar ve bu kavramlar arasındaki ilişkiler modellenir ve resmi bir ontoloji dili ile kodlanarak ontoloji oluşturulur. Ontoloji modelleme aktivitesi takip eden literatür çalışmalarından esinlenmiştir (Fernandez- Lopez et al., 1997; Noy and McGuinness, 2001; Pinto et al., 2004; Nicola et al., 2009; Guarino and Welty, 2009; Presutti et al., 2009; Ozacar et al., 2011; Suarez- Figueroa et al., 2012). Bu aktivite, *Kavramsallaştırma*, *Ontoloji* Oluşturma, *Bütünleştirme/Ayrıştırma* ve *Test Durumu*

Üretme alt aktivitelerinin birleşiminden oluşmuştur. Şekil 4.2 ontoloji modelleme yaşam döngüsünü göstermektedir.

4.3.2.1 Kavramsallaştırma

Kavramsallaştırma çoğu ontoloji modelleme çalışmasında geçen ortak bir adımdır (Fernandez-Lopez et al., 1997; Pinto et al., 2004; Nicola et al., 2009). Bu aktivitenin temel amacı alan bilgisinin kavramsal bir modelinin oluşturulmasıdır. Alan uzmanları ve ontoloji mühendisleri, İGB aktivitesinde seçilen UGK(lar)' ın veri kaynaklarını analiz eder ve alandaki terimlerin kabaca bir listesini hazırlar, bu terimlerden seçili UGK(lar)' a ait uygulama amacının dışında kalanlar elenir. Alan uzmanları ve ontoloji mühendisleri amaç kapsamında kalan terimler ile bir *Terimler Sözlüğü*, bu terimlerin açıklamaları ile birlikte bir *Kontrollü Sözlük* oluşturur. Bu açıklamalar ek kavramların bulunmasına yardımcı olmaktadır. Ayrıca, takip eden ontoloji modelleme alt aktivitelerini kolaylaştırmak amacıyla *Kaynak-Özellik-Hedef* ve *Kavram-Örnek* tabloları hazırlanır. Birinci tablo kaynak ve hedef terimler arasındaki ilişkileri gösterirken, ikinci tablo kavrama ait örnek verileri göstermektedir.



Şekil 4.2. Ontoloji modelleme yaşam döngüsü

4.3.2.2 Ontoloji oluşturma

Ontoloji oluşturma aktivitesi, bir önceki aktivitede elde edilen kavramsal modelin resmi bir ontoloji dilli ile uygulanmasını içerir. Bu aktivitenin ilk alt aktivitesi *Yeniden Kullanılabilirlik* adımıdır. Yeniden Kullanılabilirlik veb üzerindeki bilinen ve kabul edilmiş ontolojilerin yeniden kullanılmasını amaçlamaktadır. Ontoloji mühendisleri, Swoogle[8], Sindice[9], Watson[10] gibi anlamsal veb arama motorları aracılığıyla uygun ontoloji bulmaya çalışırlar. Eğer kavramsallaştırma aktivitesinin çıktıları ile eşleşen bir ontoloji bulunursa, bu ontoloji devam eden aktivitelerde kullanılmak üzere girdi olarak alınır.

Eğer uygun bir ontoloji bulamaz ise, *Kavram Hiyerarşisi Oluşturma* alt aktivitesi gerçekleştirilir. Bu aktivitede, ontoloji mühendisleri Kaynak-Özellik-Hedef tablosundaki kavram ve ilişkiler ile oluşturulacak olan ontolojinin hiyerarşisi OntoClean metodolojisi (Guarine and Welty, 2009) ile doğrular. *Tasarım Deseni Uygulam*a alt aktivitesi literatürdeki ontoloji tasarım desenlerini (Gangemi, 2005) kullanır ve ontolojinin yapısını seçilen ontoloji tasarım desenleri ile iyileştirmeyi amaçlar. Son olarak ontoloji OWL, RDFs gibi bir ontoloji dili ile TopBraid Composer[11], Protege[12], WebProtege[13] gibi ontoloji geliştirme ortamlarından faydalanılarak gerçekleştirilir.

4.3.2.3 Bütünleştirme / ayrıştırma

Bütünleştirme/Ayrıştırma aktivitesinin temel amacı büyük ontolojilerin yeniden kullanılmasını, bakımını ve evrimini sağlamaktır. ANEMONE (Ozacar, 2011) çalışmasından esinlenerek önceki iterasyonlarda üretilen ontolojilerdeki kavramlar ve bu iterasyonda üretilen ontolojilerdeki kavramlar arasındaki kavramsal bağlar analiz edilir. Daha sonra, ontoloji modüllere ayrılır veya ontolojilerdeki ortak kavramlar yeni bir ontoloji modülünde bütünleştirilir.

---

[8] http://swoogle.umbc.edu/ (Erişim tarihi: 18 Mart 2015)

[9] http://sindice.com/ (Erişim tarihi: 18 Mart 2015)

[10] http://watson.kmi.open.ac.uk/WatsonWUI/ (Erişim tarihi: 18 Mart 2015)

[11] http://www.topquadrant.com/tools/modeling-topbraid-composer-standard-edition/ (Erişim tarihi: 18 Mart 2015)

[12] http://protege.stanford.edu/ (Erişim tarihi: 18 Mart 2015)

[13] http://webprotege.stanford.edu/#List:coll=Home; (Erişim tarihi: 18 Mart 2015)

4.3.2.4 Test durumu üretme

Ontoloji Modelleme aktivitesi UGK(lar) da tanımlanan veri kaynaklarına odaklanır ve bu odaklanılan veri kaynakları için ontolojinin üst-veri bölümünü üretir. Ayrıca, bu aktivitede ontolojiler arasındaki bağ oluşturma gereksinimleri de üst-veri seviyesinde belirlenir. Bu noktada, geliştirilen ontolojinin ve bağ oluşturma gereksinimlerinin doğruluğunu ve yetreliliğini doğrulamak için ontoloji seviyesindeki test durumlarını tanımlamak mümkündür. Bu test durumlarını oluşturmak için kullanılacak temel kaynak *Analiz* aktivitesinde tanımlanan *Yeterlilik Soruları*dır. Ayrıca gerekli ise yeni yeterlilik soruları eklenebilir ya da var olan sorular genişletilebilir. Bu sorular SPARQL sorgusuna dönüştürülür ve geliştirilen ontolojinin yeterlilik sorularının çalıştırılması için gerekli bilgiyi içerip içermediği doğrulanmış olur. Oluşturulan SPARQL sorguları ele alınan UGK kartı için test durumları olarak saklanır.

4.3.3 Bağlı veri üretme

Bağlı Veri Üretme aktivitesi seçilen veri kaynaklarından ontoloji modellerine göre bağlı veri üretme ile ilgilidir. Veri üretme süreci veri kaynaklarının yapısal(örn: ilişkisel veri tabanı), yarı yapısal (örn: XML, CSV, XLS vb.) veya yapısal olmayan (örn: HTML, XHTML vb.) veri kaynağı olmasına göre değişiklik gösterir.

Yapısal veri kaynakları D2RQ veya Ultrawrap gibi RDB2RDF dönüştürücüler kullanılarak direk olarak ontolojiler ve veri kaynağı arasındaki eşlemelerin yapılmasıyla dönüştürüleblir. Ayrıca RDB2RDF dönüştürücüler tarafından kullanılan R2RML[14] eşlemelerinin oluşturulmasında "*RDB2RDF Mapping Patterns*" (Sequeda et al., 2012) çalışmasından faydalanılabilir. Yarı yapısal veri kaynakları için ise tripliser[15], Google Refine RDF Extension[16] gibi bazı yönlendirmeler doğrultusunda dönüştürme yapabilen araçlar kullanılabilir. Dönüştürücüler dışında etiketleme gibi doğal dil işleme metodları kullanılarak ta yapısal olmayan veri kaynaklarınından bağlı veri elde edilebilir.

---

[14] http://www.w3.org/TR/r2rml/ (Erişim tarihi: 18 Mart 2015)

[15] http://daverog.github.io/tripliser/ (Erişim tarihi: 18 Mart 2015)

[16] http://refine.deri.ie/ (Erişim tarihi: 18 Mart 2015)

Bir önceki aktivitede (Ontoloji modelleme aktivitesinin Test Durumu Üretme alt aktivitesi) ontolojileri doğrulamak için test durumları üretilmişti. Bu noktada ise, geliştiriler ontolojiler için veri üretmiş oldu. Bu yüzden, bu test durumları kullanılarak veri kaynaklarının doğru bir şekilde ontoloji örneklerine dönüştürülüp dönüştürülmediğinin onaylanması mümkündür. Bu amaçla, üretilen test durumlarını (SPARQL sorguları) kullanan testler yazılarak çalıştırılan SPARQL sorgu sonuçlarının beklenen sonuçlara eşit olup olmadığı onaylanır ve bu testler sürekli tümleşim ortamına eklenir.

4.3.4 Bağ oluşturma

Bağlı verinin en önemli prensiplerinden birisi bağ oluşturmaktır. Bu aktivitede veri kaynakları arasındaki bağlar otomatik olarak ya da manuel olarak oluştrulur. Bu süreci otomatikleştirmek için SILK[17] ve LIMES[18] gibi bağ keşfetme çerçeveleri kullanılabilir. Eğer, metin gibi yapısal olmayan veri kaynakları bağlanacak ise Dbpedia Spotlight[19] gibi araçlar kullanılabilir. Ayrıca diğer bir metod ise kaynaklar arasında bağ oluşturmak için farklı kaynaklardaki aynı kavramlar için ontolojik seviyede aynı URIleri kullanmaktır.

Bağlı Veri Üretme aktivitesine benzer olarak bu aktivitede bağ oluşturma gereksinimi içeren SPARQL sorguları için testler yazılır ve sürekli tümleşim ortamına eklenir.

4.4 Uygulama Geliştirme Yaşam Döngüsü

4.4.1 İlk seviye görsel uygulama geliştirme

Bağlı veri görselleştirme görselleştirilecek içeriğin büyüklüğüne bağlı olarak karmaşık ve zaman alan bir görev olabilir. Bu yüzden, Uygulama Geliştirme yaşam döngüsünün iterasyonun başında başlaması kritiktir. İlk olarak UGK(lar)'ın gereksinimleri göz önünde bulundurularak görsel bir arayüz taslağı oluşturulur. Daha sonra geliştirim takımı seçilen görselleştirme altyapısına göre farklı tasarımları değerlendirir ve üzerinde çalışısan UGK için yeni arayüz

[17] http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/ (Erişim tarihi: 18 Mart 2015)

[18] http://aksw.org/Projects/LIMES.html (Erişim tarihi: 18 Mart 2015)

[19] https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki (Erişim tarihi: 18 Mart 2015)

örnekleri üretir. Bu arayüz örnekleri müşteri ile tartışılır ve iterasyon için ilk seviye arayüz tasarımına karar verilir.

4.4.2 Yapay veri üretme

Uygulama sadece görsel arayüz birimlerini içermez, ayrıca bağlı veri tarafından gelen veriyi kullanmak için bir veri bütünleştirme katmanı da içerir. Gerçek bağlı veri Bağlı Veri Geliştirme yaşam döngüsünün Bağ Oluşturma alt aktivitesi sonunda elde edilir. Fakat, uygulama geliştirme bağlı veri üretilene kadar beklememeli ve sorunsuz bir şekilde devam etmelidir. Bu yüzden, geliştirim takımı Ontoloji Modelleme aktivitesine katılır ve/veya ontoloji modelleme takımı ile fikir alış verişi yaparak yapay bağlı veri havuzları oluşturur. Bu veri havuzları görsel tasarımın veri bütünleştirme katmanında çalışmasına imkan tanır ve görsel birimlerin iyileştirilmesine yardımcı olur.

4.4.3 Yapay veri ile tümleştirme

Bu aktivitede görsel tasarım ve tasarımın veri bütünleştirme katmanı üretilen yapay veri havuzları kullanılarak tamamlanır. Böylece tüm görsel tasarım işlevsel hale gelir ve geliştirim takımı kabul test senaryolarını tanımlayabilir. Analiz aktivitesinde tanımlanan ve Ontoloji Modelleme aktivitesinde işlenen yeterlilik soruları kabul test senaryolarını şekillendirmek için kullanılır. Bu senaryolar uygulanan görsel tasarımı düşünerek tanımlandığından, geliştirim takımı her senaryo için görsel testleri yazar ve bu testleri sürekli tümleşim ortamına ekler.

4.4.4 Gerçek veri ile tümleştirme

Uygulama Geliştirme yaşam döngüsünün son alt aktivitesinde görsel tasarım ve veri bütünleştirme katmanı Bağlı Veri Geliştirme yaşam döngüsü sonunda ortaya çıkan veri ile bütünleştirilir. Son uygulama görsel testler ile test edilir ve tüm uygulama Doğrulama ve Onaylama aktivitesi için hazır hale gelir.

4.5 Doğrulama ve Onaylama

Görsel testler ve mimari değerlendirme test planları Doğrulama ve Onaylama aktivitesinin girdileridir. Tüm görsel testler müşteri ve geliştirim takımı ile birlikte bu iterasyonda ele alınan tüm UGK ları kapsayacak şekilde değerlendirilir. Daha

sonra, mimari değerlendirme testleri mimari değerlendirme test planlarına dayanarak oluşturulur. Tanımlanan tüm testler başarılı bir şekilde sonlandığında iterasyon tamamlanır.

5. SONUÇ

Önerilen yöntemin lojistik konteyner taşımacılığı kapsamında uygulanması ile yöntemin avantajları ve dezavantajları görülmüş ve yönteme son hali verilmiştir. Buna göre, her geliştirim döngüsüne başlarken mimari değerlendirme yapılması ve her döngü sonunda oluşturulan mimarinin değerlendirilmesinin iyi bir fikir olduğu görülmüş ve ortaya kaliteli ürün çıkarılmasına büyük katkısı olduğu gözlemlenmiştir.

Buna ek olarak geliştirme döngüsü boyunca oluşturulan bağlı veri, bağ, uygulama ve mimari testleri son aşamaya bırakıldığında oluşan problemler gözlemlenmiş ve bunun sonucunda test süreçlerinin ilgili bilgi hazır olduğunda gerçekleşmesi gerektiği kararı verilmiştir. Bu karar yöntem açısından oldukça kritiktir ve yönteme çeviklik kazandırmıştır.

Yöntemin uygulanmasının diğer bir avantajı da, uygulama geliştirme ve bağlı veri oluşturma süreçlerinin paralel olarak işletilmesi gerektiğinin görülmesidir. Çünkü bu iki süreç sırayla gerçekleştirildiğinde geliştirim süresinin uzadığı, çevik ilke ve pratiklerin dışına çıkıldığı görülmüştür.

Önerilen yöntem literatürdeki bağlı veri yaşam döngüleri ile karşılaştırıldığında, sadece bağlı veri üretme ile değil aynı zamanda uygulama geliştirme ile de ilgilendiğinden dolayı diğerlerinden farklı bir perspektife sahiptir. Bu açıdan, önerilen yöntem bağlı veri uygulaması geliştirmek isteyen kurum ve kişiler tarafından kolaylıkla kullanılabilir. Ayrıca, yöntemin lojistik konteyner taşımacılığı gibi farklı roller arasında veri alışverişi gerektiren karmaşık bir alanda uygulanması, yöntemi kullanmak isteyen kurum ve kişilere örnek uygulama olması açısından önemlidir. Fakat, bu örnek uygulama kapsamında sadece yapısal veri ile çalışılmıştır. Bu nedenle gelecek dönemlerde önerilen yöntem ilişkisel olmayan farklı veri modellerini bütünleştirme çalışmalarında denenerek bu durumlara da uyum sağladığı konusunda gerekli çalışmanın yapılması planlanmaktadır.

Ayrıca, tez çalışması kapsamında elde edilen ve taşıma sürecine katılan veya katılacak olan acente (Bkz. Ek 4), liman (Bkz. Ek 5), depo (Bkz. Ek 6) ve karayolu (Bkz. Ek 7) gibi şirketlerin verilerini nasıl bir formata göre bağlı veriye dönüştüreceklerini gösteren ontolojiler oluşturulmuş ve yayınlanmıştır. Ortaya çıkan bu ontolojiler için yeni ve özgündür. Ayrıca, yöntemin uygulanması

kapsamında ortaya çıkan veri bütünleştirme çözümü lojistik sektörü için kritiktir. Bu mimari çözüm ile konteyner taşıması boyunca oluşan lojistik verisi anlık olarak bağlı veriye dönüştürülerek bütünleştirilmektedir.

KAYNAKLAR DİZİNİ

Ahn, S. B., 2005, Container tracking and tracing system to enhance global visibility, In Proceedings of the Eastern Asia Society for Transportation Studies, vol. 5, 1719 - 1727 pp.

Auer, S., Bühmann, L., Dirschl, C., Erling, O., Hausenblas, M., Isele, R., Lehmann, J., Martin, M., Mendes, P. N., Van Nuffelen, B., Stadler, C., Tramp, S. and Williams, H., 2012, Managing the life-cycle of linked data with the LOD2 stack, In Proceedings of the 11th international conference on The Semantic Web - Volume Part II (ISWC'12), Berlin, Heidelberg, 1-16 pp.

Beck, K. and Andres, C., 2004, Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley Professional, 189p.

Becker, M. and Smith, S. F., 1997, An ontology for Multi-Modal Transportation Planning and Scheduling. Carnegie Mellon University Technical Report.

Bizer, C., Heath, T. and Berners-Lee., T., 2009, Linked data - the story so far, Int. J. Semantic Web Inf. Syst., 5(3):1_22. DOI:10.4018/jswis.2009081901 5, 29

Brown, N., Nord, R. and Ozkaya, I., "Enabling Agility Through Architecture", Software Engineering Institute, https://resources.sei.cmu.edu/asset_files/WhitePaper/2010_019_001_28859.pdf (2010) (Erişim tarihi: 16 Mart 2015).

Campos, J., Arcuri, A., Fraser, G. and Abreu, R., 2014, Continuous test generation: enhancing continuous integration with automated test generation, In Proceedings of the 29th ACM/IEEE international conference on Automated software engineering (ASE '14). ACM, New York, NY, USA, 55-66 pp.

Collier, K. W., 2011, Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing, Addison Wesley Professional, 329 p.

Dalmolen, S., Cornelisse, E., Moonen, H. and Stoter, A., 2012, Cargo's Digital Shadow - A Blueprint to Enable a Cargo Centric Information Architecture, In eFreight Conference.

Dodds, L. and Davis, I., "Linked Data Patterns", http://patterns.dataincubator.org/book/index.html (2012) (Erişim tarihi: 18 Mart 2015)

KAYNAKLAR DİZİNİ (devam)

Fernandez-Lopez, M., Gomez-Perez, A. and Juristo, N., 1997, METHONTOLOGY: From Ontological Art Towards Ontological Engineering, In Proceedings of the AAAI Spring Symposium Series on Ontological Engineering. 33-40 pp.

Fraser, S., Beck, K., Caputo, B., Mackinnon, T., Newkirk, J. and Poole, C., 2003, Test driven development (TDD), In Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering (XP'03), Michele Marchesi and Giancarlo Succi (Eds.), Springer-Verlag, Berlin, Heidelberg, 459-462 pp.

Frischmuth, P., Klímek, J., Auer, S., Tramp, S., Unbehauen, J., HolzweiSSig, K. and Marquardt, C.M., 2012, Linked Data in Enterprise Information Integration, SemanticWeb – Interoperability, Usability, Applicability an IOS Press Journal, 17p.

Gangemi, A., 2005, Ontology Design Patterns for Semantic Web Content, In Proceedings of the 4th International Semantic Web Conference, The Semantic Web – ISWC 2005, Springer, Berlin, Heidelberg, 262-276 pp.

Guarino, N. and Welty, C. A., 2009, An Overview of OntoClean, Handbook on Ontologies International Handbooks on Information Systems, 201- 220 pp.

Harleman, R., 2012, Improving the Logistic Sectors Efficiency using Service Oriented Architectures (SOA), In 17th Twente Student Conference on IT.

Hartig, O., Bizer, C. and Freytag, J. C., 2009, Executing SPARQL queries over the web of linked data, In International Semantic Web Conference, 293-309 pp.

Hausenblas, M., "Linked Data Life Cycles", http://www.slideshare.net/mediasemanticweb/linked-data-life-cycles (2011) (Erişim tarihi: 16 Mart 2015)

Heath, T. and Bizer, C., 2011, Linked Data: Evolving the Web into a Global Data Space (1st edition), Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool, 1:1, 1-136 pp.

Hoxha, J., Scheuermann, A. and Bloehdorn, S., 2010, An Approach to Formal and Semantic Representation of Logistics Services, In Proceedings of the Workshop on Artificial Intelligence and Logistics (AILog) at the 19th European Conference on Artificial Intelligence (ECAI).

KAYNAKLAR DİZİNİ (devam)

Hu, B. and Svensson, G. A., 2010, A Case Study of Linked Enterprise Data, In Proceedings of the 9th International Conference on The Semantic Web, Springer Berlin Heidelberg , 129-144 pp.

Hyland, B. and Wood, D., 2011, The Joy of Data - A Cookbook for Publishing Linked Government Data on the Web. Linking Government Data, 3-26 pp.

Jentzsch, A., Cyganiak, R. and Bizer, C., "State of the LOD Cloud", http://lod-cloud.net/state/ , (2011) (Erişim Tarihi: 16 Mart 2015)

Lian, P., Park, D. and Kwon, H., 2007, Design of Logistics Ontology for Semantic Representing of Situation in Logistics, In the Second Workshop on Digital Media and its Application in Museum & Heritages, 432-437 pp.

Loutas, N., "Case Study: How Linked Data is transforming eGovernment", European Commission ISA Programme, https://joinup.ec.europa.eu/community/semic/document/case-study-how-linked-data-transforming-egovernment (2013) (Erişim tarihi: 2 Nisan 2015)

Meier, J.D., Homer, A., Taylor, J., Bansode, P., Wall, L., Boucher, R. and Bogawat, A., "How To – Design Using Agile Architecture", http://apparch.codeplex.com/wikipage?title=How%20To%20-%20Design%20Using%20Agile%20Architecture&referringTitle=How%20Tos (2008) (Erişim tarihi: 17 Mart 2015)

Mihindukulasooriya, N., Garcia-Castro, R. and Gutiérrez, M.E., 2013, Linked Data Platform as a novel approach for Enterprise Application Integration, In the Prooceedings of the 4th COLD Workshop, 12p.

Nicola, A. D., Missikoff, M. and Navigli, R., 2009, A Software Engineering Approach to Ontology Building.Information Systems 34(2), 258-275 pp.

Noy, N. F. and McGuinness, D. L., "Ontology Development 101: A Guide to Creating Your First Ontology", http://protege.stanford.edu/publications/ontology_development/ontology101.pdf (2001) (Erişim tarihi: 16 Mart 2015)

Nurmilaakso, J.M., 2008, Adoption of e-business functions and migration from EDI-based to XML-based e-business frameworks in supply chain integration, International Journal of Production Economics, 113(2), 721-733 pp.

KAYNAKLAR DİZİNİ (devam)

Ozacar, T., Ozturk, O. and Unalir, M. O., 2011, ANEMONE: An environment for modular ontology development. Data & Knowledge Engineering 70(6), 504–526 pp.

Pinto, H. S., Tempich, C. and Staab, S., 2004, Diligent: Towards a finegrained methodology for distributed, loosely-controlled and evolving engingeering of ontologies, In Proceedings of the 16th European Conference on Artificial Intelligence ECAI, 393-397 pp.

Preist, C., Esplugas-Cuadrado, J., Battle, S.A., Grimm, S. and Williams, S.K., 2005, Automated Business-to-Business Integration of a Logistics Supply Chain Using Semantic Web Services Technology, In Proceedings of the 4th International Conference on the Semantic Web.

Presutti, V., Daga, E., Gangemi, A. and Blomqvist, E., 2009, eXtreme Design with Content Ontology Design Patterns, In *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 516, CEUR.*

Schwaber, K., 2004, Agile Project Management with Scrum, Microsoft Press, Redmond, WA, USA, 133p.

Scheuermann, A. and Hoxha, J., 2012, Ontologies for Intelligent Provision of Logistics Services, In Proceedings of the 7th International Conference on Internet and Web Applications and Services.

Sequeda, J., Priyatna, F. and Villazon-Terrazas, B., 2012, Relational Database to RDF Mapping Patterns, In Proceedings of the Workshop on Ontology Patterns WOP.

Suarez-Figueroa, M., Gomez-Perez, A. and Fernandez-Lopez, M., 2012, The NeOn Methodology for Ontology Engineering, Ontology Engineering in a Networked World, 9-34 pp.

Uschold, M. and Gruninger, M., 1996, Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, 11(2).

Villazon-Terrazas, B., Vilches-Blazquez, L., Corcho, O. and Gomez Perez, A., 2011, Methodological guidelines for publishing government linked data linking government data, Linking Government Data, 27–49 pp.

ÖZGEÇMİŞ

Ad Soyad	Pınar Göçebe
Doğum Tarihi	30.10.1989
Doğum Yeri	Belfort/ Fransa
Uyruğu	Türkiye Cumhuriyeti
Eğitim	
Yüksek Lisans	2012-... Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı
Lisans	2007-2011 Süleyman Demirel Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü
Lise	2003-2007 Turgutlu Anadolu Lisesi
Araştırma Alanları	Anlamsal veb, Bağlı veri, Ontoloji Mühendisliği, Bağlı Veri Geliştirme Yöntemleri

EKLER

Ek 1 Türkçe - İngilizce Terimler Sözlüğü

Bağlı Veri: Linked Data

Çevik Yazılım Geliştirme: Agile Software Development

Çevik Analiz: Agile Analaytics

Çevik Mimari: Agile Architecture

Özellik/Hikaye Odaklı Geliştirme: Feature/Story Oriented Development

Önce Test Yaklaşımı: Test First Approach

Sürekli Tümleşim: Continuous İntegration

Test Durumu: Test Case

Yeterlilik Sorusu: Competency Questions

Anında Çözümleme Deseni: On-the-fly Dereferencing Pattern

Tarama Deseni: Crawling Pattern

Sorgu Birleştirme Deseni: Query Federation Pattern

Uç Nokta: Endpoint

Eskiyen Veri Problemi: Stale Data Problem

Veri Tazelik Seviyesi: Data Freshness Level

Uygulama Cevap Süresi: Application Response Time

Eşleme: Mapping

Yapay Veri: Mock Data

Birleştirilmiş Sorgu: Federated Query

Değişen Verileri Yakalama: Change Data Capture

Servis Odaklı Mimari: Service Oriented Architecture

Dağıtım: Deployment

Rezervasyon Numarası: Booking Number