Mühendislik Bilimleri ve Tasarım Dergisi
8(5), 142-148, 2020
e-ISSN: 1308-6693
Araştırma Makalesi

Journal of Engineering Sciences and Design
DOI: 10.21923/jesd.826355
Research Article

Özel Sayı: Uluslararası Mühendislikte Yapay Zeka ve Uygulamalı Matematik Konferansı (UMYMK 2020)
Special Issue: International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME 2020)

MODEL GÜDÜMLÜ YAZILIM GELİŞTİRME YAKLAŞIMI KULLANILARAK MİKRO SERVİS GELİŞTİRİLMESİ

Büşra İçöz[1*], Oya Kalıpsız[2]
[1] Yıldız Teknik Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye
[2] Yıldız Teknik Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul, Türkiye

Anahtar Kelimeler
Mikro Servis, Model Güdümlü Mühendislik, Model Güdümlü Geliştirme, Model Güdümlü Test, Otomatik Kod Üretimi.

Öz
Son yıllarda, monolitik uygulamaların günümüzün hızlı değişimine beklenilen cevabı verememesinin sonucu olarak uygulama geliştirilmesi sırasında mikro servis mimarisi sıklıkla tercih edilir hale gelmiştir. Birçok monolitik uygulama mikro servis mimarisi kullanılarak daha küçük servislere bölünmeye başlanmıştır. Bu durum mikro servis mimarisi ile uygulama geliştirilmesi sırasında farklı yaklaşımların ortaya çıkmasına sebep olmuştur. Model güdümlü geliştirme yaklaşımı da bunlardan biridir. Bu çalışmada, mikro servis mimari modeli ve model güdümlü mühendislik hakkında genel bilgiler paylaşıldıktan sonra mikro servis mimarisi kullanılarak restful web servislerinin geliştirilmesi sırasında yazılım geliştirme yaşam döngüsünde model güdümlü bir yaklaşım önerilmiştir. Ayrıca, yaklaşımın uygulanabilirliği ve faydaları örnek bir çalışma üzerinden değerlendirilmiştir.

MICROSERVICE DEVELOPMENT USING MODEL-DRIVEN SOFTWARE DEVELOPMENT APPROACH

Keywords
Microservice, Model Driven Engineering, Model Driven Development, Model Driven Test, Automatic Code Generation.

Abstract
In recent years, microservice architecture has become frequently preferred as a result of monolithic applications not responding to the rapid change of today. Many monolithic applications started to be divided into smaller services using the microservice architecture, which caused different approaches to emerge during application development with the microservice architecture. Model-driven development approach is one of them. In this study, after sharing the general information about the microservice architecture model and model-driven engineering, a model-driven approach in the software development life cycle is proposed during the development of restful web services using microservice architecture. In addition, the applicability and benefits of the approach were evaluated through a sample study.

Alıntı / Cite
İçöz, B., Kalıpsız, O., (2020). Model Güdümlü Yazılım Geliştirme Yaklaşımı Kullanılarak Mikro Servis Geliştirilmesi, Mühendislik Bilimleri ve Tasarım Dergisi, 8(5), 142-148.

Yazar Kimliği / Author ID (ORCID Number)
B. İçöz, 0000-0002-6155-3907
O. Kalıpsız, 0000-0001-9553-669X

Makale Süreci / Article Process	
Başvuru Tarihi / Submission Date	15.11.2020
Revizyon Tarihi / Revision Date	26.12.2020
Kabul Tarihi / Accepted Date	28.12.2020
Yayım Tarihi / Published Date	29.12.2020

1. Giriş (Introduction)

Son yıllarda web servislerin hızlı bir şekilde popüler hale gelmesinin sonucu olarak web servislerin geliştirilmesi sırasında farklı yazılım mimari ihtiyaçları ortaya çıkmıştır. Özellikle web servisler geliştirilirken uygulamaların gereksinimleri doğrultusunda geliştiriciler tarafından sıklıkla tercih edilen yazılım geliştirme mimarilerinin başında monolitik ve mikro servis mimari modelleri gelmektedir. Monolitik mimari modeli ile geliştirme, uygulamaların geliştirilmesi sırasında kullanılan geleneksel bir yöntemdir. Monolitik mimariye sahip bir

[*] Büşra İçöz: f0117027@std.yildiz.edu.tr

uygulama, birden fazla yazılım bileşeni içeren tek bir kod tabanına sahiptir. Ayrıca, tüm uygulamayı yapılandıran tek bir derleme sistemi ve tek bir çalıştırılabilir dosyaya sahiptir. Genellikle büyük ölçekli yazılım projelerinde servis odaklı mimariye geçilirken ya da geliştirmeye başlama aşamasında monolitik mimari modeli tercih edilmektedir. Bir uygulama geliştirilmeye başlanırken monolitik mimari modeli ile tasarım yapmak geliştiricilere hız kazandırmaktadır fakat ilerleyen zamanlarda servislerin birbirlerine bağımlılıklarından kaynaklanan yeni teknolojilere adaptasyon ve servislerin bağımsız bir şekilde güncellenememe sorunları ortaya çıkmaktadır. Ayrıca, tüm yazılım bileşenleri birbirine sıkı sıkıya bağlı olduğundan, bireysel yazılım bileşenleri geliştirilirken yazılım geliştirme ekipleri arasında bölünmesi zordur. Bu nedenle, monolitik yazılım çözümlerinin geliştirilmesi, test edilmesi ve bakımı geliştirmenin ileri zamanlarında daha maliyetlidir. Bu sebepten birçok büyük şirket monolitik olan uygulamalarını mikro servis mimarisi kullanarak daha küçük, tek bir işlevi olan servislere bölmüşlerdir. Mikro servis mimarisi ile uygulama geliştirirken, her mikro servisin kendine özel bir işlevi ve bağımsız veri depolama alanı vardır. Ek olarak, mikro servisler geliştiricilerin uygulamanın bireysel hizmetleri üzerinde çalışmasına olanak sağlayarak senkronize edilmiş güncelleme ihtiyacını azaltırlar. Böylece her mikro servis farklı ekipler tarafından geliştirilebilir ve birbirlerinden bağımsız olarak hizmet verebilirler. Mikro servis mimarisinin yazılım performansını iyileştirme, yazılım kaynak kodunun geliştirilme süresini azaltma, kod kalitesini artırma ve otomatik test senaryolarını oluşturma gibi birçok işlevsel ve işlevsel olmayan gereksinimleri vardır. Mikro servislerin geliştirilmesi sırasında farklı yaklaşımlar önerilmiştir, Model Güdümlü Yazılım Geliştirme (MGYG) yaklaşımı da bunlardan biridir. MGYG'nin temel amacı minimum maliyet ve zamanda hızlı ve verimli uygulamalar geliştirebilmektir. MGYG yaklaşımında uygulama geliştirme işlemine model oluşturularak başlanılır ve oluşturulan modeller referans alınarak uygulamanın kaynak kodlarının otomatik olarak üretilmesi amaçlanır. MGYG 'de esas olan model olduğu için ve modeller de programlama dillerinden daha yüksek bir soyutlama düzeyinde yazılım geliştirmeye olanak sağladığından uygulamalar platformdan bağımsız bir şekilde geliştirilebilirler. Bu sayede yazılım bileşenleri MGYG yaklaşımı ile gereksinimlerine uygun programlama dili ve platformda geliştirilebilir. Bu çalışmada, mikro servis mimarisi ile uygulama geliştirirken model güdümlü yazılım geliştirme yaklaşımı önerilmiştir. Mikro servisler tek bir amaca hizmet eden basit sistemler olduğu için tasarımları sırasında oluşturulan modeller de aynı doğrultuda daha az karmaşık ve anlaşılırdır. Mikro servisleri model güdümlü yaklaşım ile geliştirilirken modeller monolitik servislerin modellerine göre daha az karmaşıktır ve kısa sürede oluşturulabilir. Karmaşıklığı az ve belirli bir işlev için oluşturulmuş modellerden, karmaşıklığı az ve anlaşılır otomatik üretilmiş yazılım kaynak kodları elde edilebilir. Mikro servisler için oluşturulan modeller tek bir işlev için oluşturulacağı için güncellenmesi de yine monolitik servislerin modellerine göre daha az maliyetlidir. Bu çalışma şu şekilde yapılandırılmıştır: Bir sonraki bölümde, Model Güdümlü Mühendislik (MGM) alanında literatürde yapılmış çalışmalardan bahsedilmiştir. Sonrasında MGM ve mikro servis mimari hakkında genel bir bilgi paylaşımı yapılmıştır. Daha sonrasında mikro servis mimarisi ile uygulama geliştirilirken model güdümlü yaklaşımın sağladığı faydalardan bahsedilmiştir. Çalışmanın geri kalanında, mikro servis mimarisini kullanılarak web servis geliştirirken MGM yaklaşımından örnek bir çalışma üzerinden hangi aşamalarda nasıl faydalar sağlanabileceği hakkında bilgi paylaşımı yapılmıştır. Son olarak, değerlendirme ve sonuçlar paylaşılmıştır.

2. Referans Çalışması (Literature Survey)

Model güdümlü geliştirme yaklaşımı ile uygulama geliştirilmesi alanında yapılmış çeşitli çalışmalar mevcuttur. Bu alanda yapılan çalışmalar birbirlerinden genel olarak kullanılan kaynak kod üretme yöntemi, otomatik kaynak kod üretme aracı ve uygulama alanlarına göre ayrılmaktadırlar. Vitaliy Schreibmann ve Peter Braun 2015 yılında yaptığı "Restful API'lerin Model Güdümlü Geliştirilmesi"(Schreibmann ve Braun, 2015) çalışmasında, Restful isminden ilk kez bahseden Roy Fielding'in sınırlamalarına ve önerilerine dayanarak Restful servisler için soyut bir model oluşturmuşlardır. UML tabanlı meta-modeli, Restful API'leri modellemedeki diğer meta-modellerin aksine, uygulama durumlarına ve aralarındaki geçişlere odaklanır. Ayrıca çalışmada, meta veri modelleri ile yazılım kaynak kodunu oluşturmak için Xtend otomatik kod geliştirme aracı kullanılmıştır. "REST ile Uyumlu Servisler için Model Güdümlü Bir Yaklaşım" (Haupt vd., 2014) çalışmasında ise birden fazla meta-model yöntemi karşılaştırılmış ve alana özgü model olarak da bilinen DSL (Domain Specific Language) tercih edilmiş. Tercih edilen modelde yazılım kaynak kodu üretilmesi amaçlanmış ve otomatik kod üretme araçları karşılaştırılmıştır. Çalışmada, modellerin uygulanması için Eclipse Epsilon projesinden yararlanılmıştır. Modeller, EMF (Eclipse Modeling Framework) modellerini tanımlamak için kullanılan bir dil olan Emfatic'de tanımlanmıştır. Model dönüşümleri modeli, Epsilon Dönüşüm Dili (EDL) ve grafiksel model editörleri Eclipse Grafik Modelleme Çerçevesi (EGMÇ) kullanılarak oluşturulmuştur. Bu, meta-modellerinden yazılım kaynak kodu Java programlama dilinde otomatik olarak üretilmiştir.

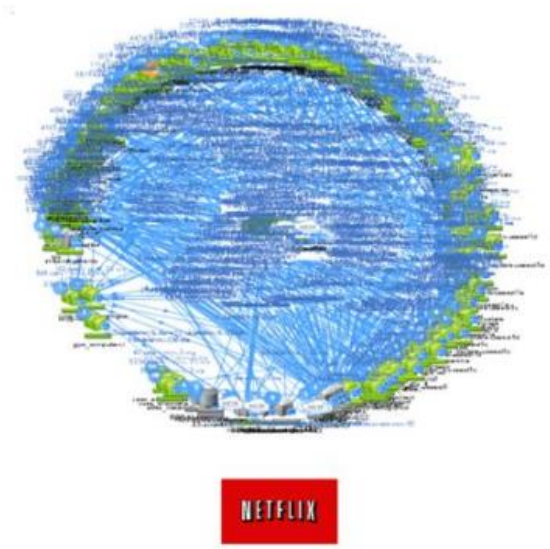
Mikro servisleri geliştirmek için model güdümlü mühendislik yöntemlerinden yararlanılarak yapılmış çalışmalar da mevcuttur. Bu çalışmalardan biri olan "Mikro Servis Yazılım Mimarisi Oluşturulmasında Model Güdümlü Bir Yaklaşım" (Terzić vd., 2018) çalışmasında, mikro servis mimarisi tasarım ilkelerini takip eden yazılım uygulamalarının modellenmesi, geliştirilmesi ve canlı ortamda çalıştırılması ile ilgili zorlukları ele alan model odaklı bir yaklaşım önerilmiştir. Bu çalışmada yazarlar, restful mikro servislerin tasarım ilkelerini takip eden

yazılım uygulamalarını tanımlamak için kullanılan bir model odaklı araç olan MicroDSL (Terzić vd., 2017) dili kullanılarak yazılım geliştirilebilen MicroBuilder (Terzić vd., 2017) adlı modele dayalı geliştirme aracından bahsetmişlerdir. Ayrıca, bu araç ile otomatik Java kod üretiminden de bahsedilmiştir.

Yapılan literatür çalışması sonucunda bu çalışmada da mikro servisler geliştirilirken yazılım geliştirme yaşam döngüsünün her bir aşaması için model güdümlü geliştirme yaklaşımının sağladığı faydalar ve uygulanabilirliği örnek bir çalışma üzerinden değerlendirilmiştir.

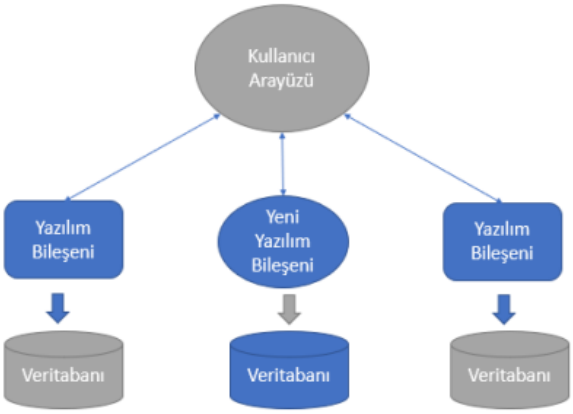
3. Mikro Servis Mimarisi (Microservice Architecture)

Mikro servis mimari modeli, monolitik mimariye sahip uygulamaları daha küçük, bağımsız tek bir amaca hizmet eden parçalara bölünmesi felsefesine dayanan modüler mimari stilinin bir örneğidir. Mikro servis mimari modeli, uygulama ve veri tabanı arasındaki ilişkiyi önemli ölçüde etkiler. Tek bir veri tabanı şemasını diğer servisler ile paylaşmak yerine, her servisin kendi veri tabanı şeması vardır ve her servis ihtiyaçlarına en uygun veri tabanını kullanabilir. Bu yaklaşım çoğu zaman bazı verilerin çoğaltılarak kullanılmasına neden olabilir. Ancak, mikro servis mimarisi kullanarak uygulama geliştirildiğinde, her servisin ayrı bir veri tabanı şemasına sahip olması önemlidir, çünkü bu durum servislerin birbirlerine olan bağımlılığının önemli ölçüde azalmasına fayda sağlar. Netflix'in monolitik mimaride olan servis yapısını mikro servis mimarisi ile değiştirmesindeki en önemli nedenlerden biri de tek bir veri tabanına sahip olması ve bu durumdan kaynaklanan bir hatadan dolayı bir süre hizmet veremediğinde aldığı ciddi zarar olarak biliniyor (Kwiecien, 2019). Bu sebepten Netflix monolitik yapıda olan servislerini Şekil 1'de de gösterildiği gibi mikro servislere bölmüştür.



Şekil 1 : Netflix mikro servis mimari yapısı (Kwiecien, 2019)

Netflix gibi birçok şirket benzer sebeplerden dolayı mikro servis yapısını tercih etmişlerdir. Örneğin; yeni teknolojilere kolay adapte olabilmek, Şekil 2'de gösterildiği gibi yeni bir servisi en az maliyet ile çalışan uygulamaya dahil edebilmek vb.



Şekil 2: Mikro servis mimarisi örneği

Mikro servisler, yazılım geliştirilirken sağladığı faydalardan dolayı birçok büyük teknoloji şirketlerinin de monolitik yapıda olan uygulamalarını mikro servis mimarisi ile daha küçük servislere bölmesine neden olmuştur. Mikro servisler şunları sunar:

-İyileştirilmiş bir hata yalıtımı vardır. Büyük uygulamalar çoğunlukla tek bir yazılım bileşeninin arızalanmasından etkilenmezler.

-Teknoloji bağımlılığı ortadan kalkmıştır. Servisler işlevlerine en uygun olan teknolojiler ve programlama dili ile geliştirilebilir. Bu durum geliştiriciye büyük oranda esneklik sağlamaktadır.

-Mikro servislerin genel işlevinin anlaşılırlığı daha kolaydır. Yazılım geliştirme ekibine yeni bir ekip üyesi dahil olduğu zaman ilk başlarda bütün uygulamaya hâkim olmak yerine geliştirme yapacağı mikro servise odaklanarak hızlı bir şekilde adapte olmasına olanak sağlar.

- Servisler birbirinden bağımsız olduğu için, tüm uygulama yerine, en çok ihtiyaç duyulanlar uygun zamanlarda daha kolay ölçeklendirilebilir. Doğru ölçeklendirme yapıldığında, bu durum maliyet tasarrufunu sağlayabilir.

3. Model Güdümlü Mühendislik (Model-Driven Engineering)

Model Güdümlü Mühendislik (MGM), alana özgü modellerin oluşturulmasına ve kullanılmasına odaklanan bir yazılım geliştirme metodolojisidir. Modeller, belirli bir uygulama alanını yöneten bilgi ve faaliyetlerin soyut temsili olarak algılanabilir (Cuong, 2015). Modeller, ürün yöneticileri, tasarımcılar, geliştiriciler ve uygulama alanına özgü kullanıcıların arasında kapsamlı iletişim ile geliştirme yaşam döngüsünün çeşitli aşamalarında kullanılabilir. MGM, sistemler arasındaki uyumluluğu en üst düzeye çıkararak, tasarım sürecini basitleştirir ve sistem üzerinde çalışan bireyler ve ekipler arasındaki iletişimi teşvik ederek verimliliği artırmayı amaçlamaktadır (Qafmolla ve Nguyen, 2010). Model Tabanlı Yaklaşım (MTY), yazılım yaşam döngüsünü hızlı, verimli, minimum maliyet ve sürede tamamlamak için kullanılan bir yazılım geliştirme yaklaşımıdır. Modele dayalı yazılım geliştirme yaklaşımında, uygulamanın ilk olarak modeli oluşturulur ve bu doğrultuda uygulamanın kaynak kodunun üretilmesi hedeflenir. Model Güdümlü Mimari (MGM) olarak da adlandırılan bu model güdümlü yazılım geliştirme yöntemi, Nesne Yönetim Grubu (NYG) tarafından desteklenmektedir. NYG (Noureen vd., 2016) grubu tarafından belirlenen standartlar, bir dizi modelin, gösterim ve dönüşüm kurallarının nasıl tanımlanacağına ilişkin standartlardan oluşur. Bu şekilde, bir uygulama farklı yazılım platformlarında hızlı ve kolay bir şekilde geliştirilebilir.

4. Mikro Servis Mimarisi ile Uygulama Geliştirilirken Model Güdümlü Yaklaşımın Sağladığı Faydalar (Benefits Provided by Model-Driven Approach While Developing Applications with Microservices Architecture)

Mikro servis mimarisi kullanılarak model güdümlü yazılım geliştirmenin sağladığı faydalar aşağıdaki gibidir:

- Model güdümlü geliştirme kullanarak uygulama geliştirmek daha hızlıdır, Model güdümlü geliştirmede, bir yazılım uygulamasının modeli, geleneksel programlama dillerinden daha yüksek bir soyutlama düzeyinde belirtilir. Bu model ile otomatik olarak üretilen yazılım kaynak kodları sayesinde çalışan bir yazılım uygulaması elde edilir. Kullanılan model daha yüksek bir soyutlama düzeyinde iken, kodda ifade edilen aynı modele kıyasla çok daha küçüktür. Başka bir deyişle; modeldeki her öğe birden çok kod satırını temsil eder. Mikro servis mimarisine ile geliştirilen bir uygulama için oluşturulacak modeller daha basit olacağı için mikro servisler için sağlam ve kaliteli yazılım kodları elde edilebilir. Bir model üzerinden çeşitli programlama dillerini destekleyen otomatik kod üretim araçları ile mikro servislerin amacına ve işlevine uygun kodlar üretilebilir. Bu durum mikro servis mimarisi ile uygulama geliştirmenin temel özelliklerinden olan amacına uygun programlama dili ile servislerin geliştirilmesi gereksinimi bire bir karşılamaktadır.

- Model güdümlü geliştirme yaklaşımı ile geliştirilen uygulamalar daha az maliyetlidir. İlk neden, model güdümlü geliştirme yaklaşımı yazılım geliştirme süresini kısalttığı için uygulamalar daha kısa bir pazara sunum süresine sahip olurlar. İkinci neden, daha düşük maliyetle (daha az insan gücü, kısa geliştirme süresi vb.) ile uygulama geliştirmenin mümkün olmasıdır. Model güdümlü geliştirme yaklaşımı kullanılarak geliştirilen mikro servislerde değişiklik yapılması ve servislerin bakımı da daha az maliyetli olur. Mikro servis mimarisi kullanılarak geliştirilen uygulamalar birbirinden bağımsız servislere sahip olduğu için bakım maliyeti model güdümlü geliştirme ile daha da düşürülebilir. Uygulamanın tamamını güncellemek yerine ilgili servisin modeli güncellenerek aynı doğrultuda yazılım kaynak kodları otomatik olarak üretilebilir.

- Model güdümlü geliştirme yaklaşımı yazılım kaynak kodunun kalitenin artmasına yol açar. Model güdümlü yaklaşım ile geliştirilen bir uygulamanın temelini üst düzey soyut bir model oluşturduğu için; uygulamanın

(teknik) kalitesi model üzerinden yazılım kaynak kodunu otomatik olarak üreten araca bağlıdır. Yazılım kaynak kodlarını otomatik olarak üreten araçlar en iyi deneyimlere göre yazılım kaynak kodlarını üretirler, ayrıca yazılım kodunun dokümantasyonunu da sağlarlar. Bu nedenle, model güdümlü yaklaşım ile geliştirilmiş mikro servislerin kod kalitesinde artış olur çünkü en iyi deneyimler ile üretilmiş ve tek bir elden çıkmış daha az kod tekrarı olan yazılım kaynak kodları elde edilir.

- Model güdümlü geliştirme yaklaşımı ile modeller üzerinden otomatik test durumları üretimi de gerçekleştirilebilir. Bu durum özellikle tek bir işleve sahip mikro servisler için bütün durumların üzerinden geçilmesini ve yazılım test maliyetinin büyük oranda azaltılmasına fayda sağlamaktadır. Mikro servisler tek bir amaca hizmet ettiği için karmaşık modellere sahip monolitik servislere göre oluşturulan test durumları kodun büyük bir kısmını kapsar. Böylece yazılım hataları test sırasında kolay bir şekilde tespit edilebilir ve hızlı bir şekilde düzeltilebilir.

5. Uygulama Çalışması (Case Study)

Mikro servisler model güdümlü bir yaklaşımla geliştirilirken; model oluşturma, model üzerinden otomatik kod üretme, model üzerinden otomatik testlerin oluşturulması ve canlı ortama dağıtılması adımları izlenir. Bu bölümde model güdümlü yaklaşım kullanılarak Restful bir ürün yönetimi sistemi uygulamasının depo yönetimi mikro servisi geliştirilirken izlenilen adımlardan ve mikro servis geliştirilmesi sırasında her bir adımda model güdümlü yaklaşımın uygulanabilirliğinden bahsedilmiştir.

5.1 Gereksinim Analizi (Requirement Analysis)

Ürün yönetimi sistemi; sipariş yönetimi, ürün yönetimi, depo yönetimi ve kullanıcı yönetimini içeren dört adet mikro servisten oluşan bir uygulamadır. Ürün yönetimi sistemi uygulamasının depo yönetimi mikro servisinin gereksinimleri aşağıdaki gibidir;

1. Kullanıcılar depodaki ürünlerin tamamını listeleyebilirler.
2. Kullanıcılar depodaki ürünleri kategorilerine göre listeleyebilirler.
3. Kullanıcılar depoya ürün ekleyebilirler.
4. Kullanıcılar depodaki ürünleri silebilirler.
5. Kullanıcılar depodaki ürünleri güncelleyebilirler.

Yukarıda gereksinimleri tanımlanan depo yönetimi mikro servisi çalışmanın geri kalanında model güdümlü yaklaşım ile geliştirilecektir.

5.2 Model Çıkarma (Model Extraction)

Depo yönetimi mikro servisinin gereksinimleri çıkartıldıktan sonra gereksinimler doğrultusunda UML diyagramı, veri tabanı şeması ve DSL (Domain Specific Language) gibi model yapıları kullanılarak modeller oluşturulabilir. Depo yönetimi mikro servisinin geliştirilmesi sırasında DSL kullanılmıştır.

DSL (Domain Specific Language)

Alana özgü dil olarak da adlandırılan DSL, belirli bir amaç için kullanılan özel bir dildir. Belirli bir sorunu çözmek için kullanılır. Bu çalışmada Şekil 3'de de görüldüğü gibi Eclipse ile birlikte Teloys aracı üzerinde DSL oluşturulmuştur.

```
1 // Entity Depo
2
3 Depo {
4   urunFiyati : long ;
5   urunTipi:    string;
6   urunNumarası: int;
7   urunEkle(Depo urun)   : void
8   urunSil(int urunNumarasi)   : void
9   urunGuncelle(Depo urun)   : void
10  urunFlitrele(String urunTipi):string
11 }
12
```

Şekil 3: Eclipse'de Teloys aracı kullanılarak oluşturulmuş DSL örneği

Mikro servisler karmaşık sistemler olmadığı için depo yönetimi mikro servisinin gereksinimleri doğrultusunda oluşturulan model örneği Şekil 3'de de görüldüğü gibi model yazılım geliştirme ekibindeki tüm üyeler tarafından anlaşılır ve güncellenebilir basitliktedir. Mikro servise daha sonradan yeni bir özellik eklemek basit bir modele sahip olduğu için daha az maliyetlidir.

5.3 Otomatik Kod Üretme ve Birim Testi (Automatic Code Generation and Unit Test)

Bu kısımda Eclipse ile birlikte Telosys aracı kullanılmıştır. Telosys UML, sınıf diyagramı, DSL gibi modelleri kullanarak otomatik bir Java kod çerçevesi oluşturur. Ücretsiz ve açık kaynaklı bir kod üretme aracıdır. Şekil 4' de, Teloys kullanarak Eclipse'de otomatik olarak oluşturulan kod parçasının bir örneği gösterilmektedir.

```
@RequestMapping(value = "/sil/") // GET or POST
public String urunSil(RedirectAttributes redirectAttributes) {
    log("Action 'sil'" );
    try {
        boolean deleted = urunService.deleteById( );
        log("Urun deleted. Key : " + toString() + " result = " + deleted );
        //--- Set the result message
        messageHelper.addMessage(redirectAttributes,
        new Message(MessageType.SUCCESS,"delete.ok"));
    } catch(Exception e) {
        messageHelper.addException(redirectAttributes, "stock.error.sil", e);
    }
    return redirectToList();
}
```

Şekil 4: Eclipse de Teloys aracı ile oluşturulmuş kod parçası

Şekil 4'de de göründüğü gibi depo yönetimi mikro servisi'nin ürün silme gereksinimi için oluşturulmuş basit bir kod parçası örneği vardır. Şekil 3'deki model de bir satır olarak belirtilen ürün silme gereksinimi için Teloys aracı ile on dört satır otomatik kod üretilmiştir. Otomatik üretilen kodlar Şekil 4'de de görüldüğü gibi kod içinde açıklayıcı dökümantasyonlar içermektedir. Bu durum yazılım geliştiricilerine, yazılım kodunda ekleme ya da güncelleme yaptıkları sırada büyük ölçüde kolaylık sağlamaktadır.

Yazılımın en küçük fonksiyonunu test ettiğimiz birim testleri, yazılımın kalitesine önemli faydalar sağlar. Birim testleri kusurların düşük bir maliyetle erken zamanda bulmasını sağlar. Model güdümlü geliştirme yönteminde, otomatik kod üretme araçları genellikle model üzerinden otomatik birim test üretimi de gerçekleştirirler. Bu sayede, birim testler tüm kodu kapsayacak şekilde otomatik olarak oluşturulur ve kod kapsamını artırmak için fazladan çabaya ihtiyaç duyulmaz. Şekil 5'te Eclipse üzerinde Teloys aracı kullanılarak otomatik üretilmiş birim test örneği gösterilmiştir.

```
@Test
public void testPutGetRemove() {

    System.out.println("Testing class DepoCache ..." )

    Depo depo = new Depo();
    populate(depo);
    System.out.println("Entity populated : " + depo );

    ProductCache.putProduct(depo) ; // Store in cache

    Depo depo2 = DepoCache.getDepo( );
    assertTrue( depo == depo2 ) ; // Same instance

    DepoCache.removeDepo( ) ; // Remove from cache

    Depo depo3 = DepoCache.getDepo( );
    assertTrue( null == depo3 ) ; // Not in cache
    assertNull(depo3); // Not in cache

}
```

Şekil 5: Eclipse Teloys platformunda otomatik üretilmiş birim testi örneği

Teloys aracı ile Şekil 3'deki modelde bulunan her bir gereksinim için bir adet toplamda dört adet otomatik test senaryosu üretilmiştir . Otomatik oluşturulan birim testlerinin tümü Eclipse platformunda koşulmuştur ve tümü başarılı olmuştur.

5.4 Dağıtım ve Hizmet Yönetimi (Deployment and Service Management)

Model güdümlü yaklaşım ile bulut ortamına ve fiziksel ortama dağıtım için literatürde yapılan çalışmalar mevcuttur, fakat bu çalışmada oluşturduğumuz model üzerinden doğrudan otomatik dağıtım dosyasını oluşturabilecek bir araç kullanılmamıştır. Dağıtım el ile oluşturulmuş bir docker yaml dosyası ile bulut ortamına manuel olarak yapılmıştır. Bu aşamada her bir mikro servis için ortam gereksinimleri farklılık gösterebileceği için model güdümlü geliştirme yaklaşımı ile oluşturulmuş modellerin sıklıkla güncellenmesi ihtiyacı ortaya çıkabilir ve bu yaklaşım geliştiriciler açısından kullanışlı olamayabilir. Yapılan uygulamanın sonucu olarak model güdümlü geliştirmenin mikro servislerin dağıtımı sırasında yetersiz kaldığı gözlemlenmiştir.

6. Değerlendirme ve Sonuçlar (Result and Discussion)

Bu çalışmada, mikro servis mimarisi ve model güdümlü yaklaşım hakkında genel bilgi paylaşımı yapılmıştır. Ayrıca, bir mikro servisin geliştirilmesi sırasında yazılım yaşam döngüsünün aşamalarında model güdümlü yaklaşımdan nasıl faydalanılabileceğini gösteren örnek bir çalışma yapılmıştır. Yapılan çalışmada gözlemlenen, model tabanlı yaklaşım, özellikle yazılım kodu ve otomatik test senaryolarının üretiminde yazılımın verimliliğini ve hızını arttırmaktadır ve bu durum mikro servislerin geliştirilmesinde önemli bir fayda sağlamaktır. Ayrıca, kod şablonunun otomatik oluşturulması, yinelenen kodların azaltılması, geliştirme süresinin kısalması gibi birçok fayda sağladığı gözlemlenmiştir. Mikro servisler platformdan bağımsız uygulamalardır. Model güdümlü bir yaklaşım ile geliştirme yapılarak platform bağımlılığı ortadan kaldırılmıştır. Model güdümlü yaklaşım ile benzer mikro servisler kısa bir süre içinde modeller güncellenerek oluşturulabilir. Mikro servisler tek bir işleve sahip olan basit servisler olduğu için model güdümlü yazılım geliştirme yaklaşımı ile uygulama geliştirilmesinin uygun olduğu görülmüştür. Büyük ve karmaşık uygulamalarda aynı doğrultuda model karmaşıklığı da artacağından otomatik oluşturulan yazılım kaynak kodunun doğruluğunun ölçülmesi ve kaynak kodunun güncellenmesi zorlaşmaktadır, bu nedenle model güdümlü yaklaşımın daha basit uygulamalar için daha verimli sonuçlar verdiği gözlemlenmiştir. Yapılan çalışmalar ve gözlemler sonucunda çalışmanın bir sonraki aşamasında model güdümlü yazılım geliştirme yaklaşımının yetersiz kaldığı düşünülen alan olan mikro servis mimarisi ile geliştirilen uygulamaların model güdümlü olarak dağıtımı üzerinde çalışılmasına karar verilmiştir.

Çıkar Çatışması (Conflict of Interest)

Yazarlar çıkar çatışması bildirmemişlerdir.

Kaynaklar (References)

Aleksandra Kwiecien, 2019. “10 companies that implemented the microservice architecture and paved the way for others” [Online], Available https://divante.com/blog/10-companies-that-implemented-the-microservice-architecture-and-paved-the-way-for-others/ [Accessed:24-12-2020].

Branko Terzić, Vladimir Dimitrieski, Slavica Kordić, Gordana Milosavljević, Ivan Luković, 2017. “MicroBuilder: A Model-Driven Tool for the Specification of REST Microservice Architectures”.

Branko Terzić, Vladimir Dimitrieski, Slavica Kordić, Ivan Luković , 2018. “A Model-Driven Approach to Microservice Software Architecture Establishment”.

Compliant Services”, IEEE International Conference on Web Services

Florian Haupt; Dimka Karastoyanova; Frank Leymann; Benjamin Schroth, 2014. “A Model-Driven Approach for REST

Nicolas Ferry, Marcos Almeida and Arnor Solberg, 2017. “The MODAClouds Model-Driven Development”.

Noureen, A.; Amjad, A.; Azam, 2016. “ F. Model Driven Architecture - Issues, Challenges and Future Directions”. J. Softw. ,11, 924–933, doi:10.17706/jsw.11.9.924-933.

Viet-Cuong Nguyen, 2015.” Model Driven Testing of Web Applications Using Domain Specific Language”,

Vitaliy Schreibmann and Peter Braun ,2015. “Model-driven Development of RESTful APIs”. WEBIST 2015-11th International Conference on Web Information Systems and Technologies.

X. Qafmolla, V. Nguyen, 2010. “Automation of Web Services Development Using Model-driven Techniques. In Institute of Electronics Engineers”, The 2nd International Conference on Computer and Automation Engineering (ICCAE 2010), pp. 190-194.