Yazılım Dağıtım Sürecinin Otomatikleştirilmesine İlişkin Uygulamalı Bir Çalışma

***

A Practical Study on Automatizing the Software Deployment Process

Ender ŞAHİNASLAN[1]
Nusret ARPACIOĞLU[2]
Önder ŞAHİNASLAN[3]

DOI:10.33461/uybisbbd.1206484

Makale Bilgileri

Makale Türü: Araştırma Makalesi

Geliş Tarihi: 24.11.2022

Kabul Tarihi: 30.04.2023

Öz

Yazılım dağıtımı, geliştirilen bir uygulamanın çalıştırılacak dijital ortamlara yüklenmesidir ve yazılım geliştirme yaşam döngüsünde önemli bir aşamadır. Herhangi bir hataya veya kesintiye yol açmayacak şekilde titizlikle uygulanmalıdır. Küçük organizasyon ve sistemlerde bu sürecin işleyişinin manuel olarak işletilebilmesi mümkündür. Ancak orta ve büyük organizasyonlarda dağıtım yapılacak ortamların çeşitliliği, büyüklüğü, karmaşıklığı buna engeldir. Ayrıca iş sürekliliği, güvenlik ve uyum gibi gereksinimler dikkate alındığında manuel dağıtımdan kaynaklı birçok risk söz konusudur. Bu tür risk ve sorunların aşımında otomatik yazılım dağıtım araçlarına ihtiyaç vardır. Bu çalışmada yazılım yaşam ve dağıtım süreci, manuel dağıtımdan kaynaklı sorunlar, sorunların çözümünde kullanılan otomatik dağıtım araçları incelenmiştir. Yazılım dağıtım sürecinin otomatikleştirilmesi uygulamalı olarak çalışılmıştır. Bu çalışma daha önce yapılan çalışmalardan farklı olarak, konusu ve uygulamalı bir sunum olması sebebiyle ilgili kurum çalışanları, öğrenci ve araştırmacılar için teknik bir rehber niteliğindedir. Yazılım dağıtım süreci ve bu sürecin otomatikleştirilmesine yönelik uygulamalı çalışma literatüre kazandırılmıştır.

Anahtar Kelimeler: *Teknoloji ve Yenilik, Bilişim, Otomatik Yazılım Dağıtımı, Süreç Yönetimi.*

Abstract

Article Info

Paper Type: Research Paper

Received: 24.11.2022

Accepted: 30.04.2023

Software deployment is the installation of a developed application on digital media to be run and it is an important stage in the software development life cycle. It should be applied meticulously so as not to cause any errors or interruptions. In small organizations and systems, it is possible to operate this process manually. However, the diversity, size, and complexity of the environments to be distributed in medium and large organizations prevents this. In addition, there are many risks arising from manual deployment when considering requirements such as business continuity, security and compliance. Automatic software deployment tools are needed to overcome such risks and problems. In this study, software lifetime and deployment process, problems arising from manual deployment and automatic deployment tools used to solve problems are examined. Automation of the software deployment process has been practiced. This study, unlike previous studies, is a technical guide for the relevant institution staff, students and researchers due to its subject and practical presentation. The software deployment process and practical study on automating this process have been brought to the literature.

Keywords: *Technology and Innovation, Informatics, Automatic Software Deployment, Process Management.*

Atıf/ to Cite (APA): Şahinaslan, E., Arpacioğlu, N., & Şahinaslan Ö. (2023). Yazılım Dağıtım Sürecinin Otomatikleştirilmesine İlişkin Uygulamalı Bir Çalışma. Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi, 7(1), 41-67.

[1] Dr. Öğr. Üyesi, Mudanya Üniversitesi, Bilgisayar Mühendisliği Bölümü, dr.endsa@gmail.com
[2] Yüksek Lisans, Maltepe Üniversitesi, Bilgisayar Mühendisliği Bölümü, nusretar@gmail.com
[3] Dr. Öğr. Üyesi Maltepe Üniversitesi, Bilişim Bölümü, ondersahinaslan@maltepe.edu.tr

1. GİRİŞ

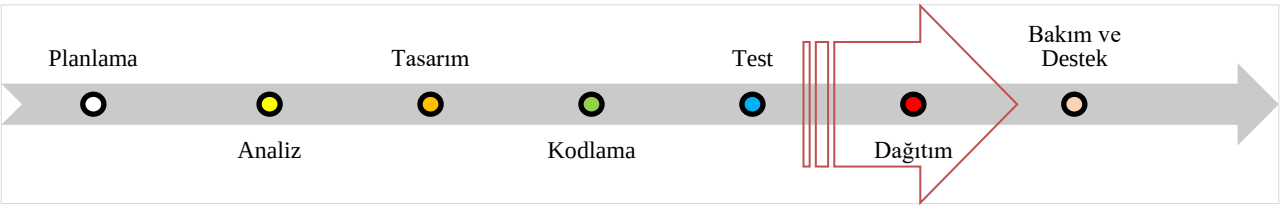
Yazılım ürünleri bir ihtiyacın karşılanması için tasarlanır, kodlanır, uygulama/ürün haline getirilerek servis edilir. Dijitalleşen bir dünyada yazılımlara duyulan ihtiyaç sürekli artmaktadır. Bu ihtiyaç bazen yeni bir uygulama bazen de var olan bir uygulamada yapılan değişiklikler yoluyla giderilmektedir. Yeni geliştirilen yazılım ürününün ya da mevcut bir yazılımın küçük bir parçasının değişimi sonucunda oluşan uygulamanın dağıtımı sürekliliği olan bir süreçtir (Highsmith & Cockburn, 2001). Bir uygulamanın dağıtımında konuşlandırılacak altyapı, sistem ve uygulama ortamlarının özel yapılandırma ihtiyaçları dikkate alınır. Dağıtımın eksiksiz, tam, güvenli ve çalışabilir şekilde uygulanması gerekir. Yazılım geliştirme yaşam döngüsünün son aşamalarından biri olan dağıtım ('*deployment*') kısmı yakın bir geçmişe kadar genellikle işinde uzman kişiler tarafından manuel olarak gerçekleştirilmekteydi. Ancak insana dayalı manuel olarak gerçekleştirilen bir dağıtım işlemi insanın dikkat ve becerisine de bağlı olarak birçok hata ve riske açıktır. Bu tür sorunlar dikkate alındığında manuel yerine otomatik dağıtım süreçlerine geçiş kaçınılmaz olmuştur. Özellikle bankacılık, sigortacılık ve savunma gibi çok farklı sistem ve mimarilerdeki uygulamaların çalıştığı, hata ve gecikmelere tahammül edilemeyen sektörlerde yazılım dağıtım işlerinin manuel olarak yürütülebilmesi imkânsız hale gelmiştir. Hızlı ve hatasız gerçekleştirme arzusu, teknolojik ilerlemeler, BT standart gereksinimleri, güvenlik ve yasal uyum gibi ihtiyaçlar neticesinde yazılım dağıtımının otomatik kullanımı yaygınlaşmaya başlamıştır. Dağıtım süreci ve sürecin otomatikleştirilmesinde kullanılan araçlarla ilgili bilgi ve tecrübe yetersizlikleri hatalı veya eksik yapılandırmalara sebep olabilmektedir. Bu tür sorunların varlığı dağıtım sürecinin başarılı bir şekilde yürütülebilmesi önünde birer engeldir.

Yazılım dağıtım süreci, geliştirilen bir uygulamanın ilgili sistemlere dağıtımını, kullanıma hazır hale getirilmesini diğer bir deyişle uygulamanın çalışır durumda kalmasını içeren üretim sonrası karmaşık bir süreçtir (Arcangeli vd., 2015). Her bir yazılım ürününün farklı bir yapılandırma ve dağıtım yöntemine ihtiyaç duyması, bir dağıtımın farklı zamanlarda benzer şekilde gerçekleştirilme talebi ek insan kaynağı ihtiyacına dolayısıyla iş yükünün artmasına sebep olmaktadır. Aynı zamanda operasyonel işlemlerden kaynaklı hata ve risklere açık ve maliyetli bir yazılım dağıtım sürecinin manuel bir şekilde yönetilebilmesi oldukça güçtür. Oysa otomatik dağıtım araçlarının kullanılması durumunda; başlangıçta dikkatlice tasarlanan bir otomatik dağıtım süreci ile insan müdahalesi olmadan, neredeyse sıfır hata ile istenildiği kadar dağıtım gerçekleştirilebilir. Otomatik dağıtım süreci, dağıtım yapılacak uygulama ve ortamları bilen teknik uzmanlar tarafından bir defaya mahsus modellendikten sonra manuel herhangi bir müdahale olmadan istenildiği kadar kullanılabilir.

Yazılım yaşam döngüsü, bir değişiklik yapma kararından üretim safhasına kadar geçen süre olarak bilinir (Farley & Humble, 2010). Bu süre birçok kurumda yazılım geliştirme süreci haftalar hatta aylarca sürerken, kritik bir hataya yönelik yeni bir özelliğin eklenmesi veya küçük bir düzeltme de çoğu zaman günlerce veya haftalarca sürebilmektedir (Poppendieck ve Poppendieck, 2007). Yazılım geliştirme sürecindeki küçük iyileştirme, yazılım kalitesinde önemli bir iyileşmeyle sonuçlanabilir (Pai, 2002). Ancak küçük bir değişikliğin bile uzun sürmesi bir şeylerin yanlış olduğunun açık bir göstergesidir (Morris, 2016). Özellikle orta ve büyük ölçekli kurumlarda yazılım dağıtım sorunlarının aşılmasına yönelik çalışmalar yürütülmektedir. Sunulan çözümlerden birisi yazılım dağıtımında otomatik araç kullanımıdır. Yazılım dağıtımında otomatik araç kullanımı sürecin karmaşıklığını azaltırken yazılım uyarlamasında geçen süreyi de en aza indirir (Ruiz-Rube vd., 2015). Karmaşık yazılımlarla bile, kontrollü bir otomatik süreç olarak oluşturma, devreye alma, test etme ve dağıtım işlemleri otomatikleştirilebilir, döngü süresi saatlere hatta dakikalara düşürülebilir (Eloranta, 2018). Bir yazılım uygulamasının yeni sürümünde yapılan tüm değişiklikler, eklenen özellikler, iyileştirmeler, hata düzeltmeleri ve kullanımdan kaldırılan özelliklere dair tüm bilgiler sürüm notları içerisine işlenir. Sürüm notları ise yazılım geliştirmede hayati bir rol oynamasına rağmen genellikle el yordamıyla oluşturur (Ali vd., 2020). Sürekli tekrarlanan bir yazılım dağıtım sürecinin otomatikleştirilmesi yazılım dağıtım sürecinin daha hızlı, doğru ve çevik bir biçimde yürütülmesine katkı sunar.

1.1. Yazılım Geliştirme Süreci

Yazılım geliştirme süreci, kullanıcı ihtiyaçlarını karşılamak için kabul edilebilir bir sürede kullanışlı çıktılar üretecek şekilde belli yöntem, adım ve işlemler kullanarak bir standart oluşturma işidir (Şahinaslan, 1998). Bu süreç diğer ürün geliştirme süreçlerine göre daha uzun zaman alan, karmaşık ve zor bir süreçtir (Borandag & Yücalar, 2020). Genel kabul gören bir yazılım yaşam döngüsü, talep ve ihtiyaçlara göre kalite güvence gereksinimleri, insan, ortam, süre, maliyet, risk gibi unsurların göz önüne alındığı fizibilite ve planlama çalışmaları ile başlar. Analiz, tasarım, kodlama, test, dağıtım, bakım ve destek süreçleriyle devam eder. Şekil-1'de yazılım geliştirme süreci temel aşamaları gösterilmektedir.



Şekil 1. Yazılım geliştirme süreç aşamaları

Bu temel aşamalar tek yönlü bir akış gibi gösterilmekle birlikte uygulamada her bir adımdan önceki bir adıma geri dönüş de mümkündür. Ancak asıl hedeflenen her bir adımı mümkün olduğunca eksiksiz tamamlayarak sonraki aşamaya geçmek, süreci başarıyla tamamlamaktır.

1.2. Dağıtım Süreci

Çalışmanın ana konusu olan dağıtım süreci, yeni geliştirilen veya güncellenen bir yazılımı kullanıma hazır hale getirme, kullanıcıya sunma evresidir. Yazılım dağıtımı genelde kolay bir süreç gibi görünmekle birlikte titiz ve detaylı bir çalışma gerektiren bir iştir. Test sürecinin başarıyla geçilmesine rağmen dağıtım sonrası istenmeyen sorunlarla karşılaşılması da muhtemeldir. Özellikle test ortamı ile çalışma ortamındaki kullanıcı ve rol yapısı farklılığı, yetkilendirme, güvenlik, iş sürekliliği, sistem kapasite ve performans farklılıklarından kaynaklı birçok sorunla karşılaşılabilmektedir. Bu süreç kodlama ve test aşamasını geçen bir uygulamanın kullanılacağı ortamlara dağıtımı ve başarıyla yürütülmesinde geçen aşamadır. Bu süreç kapsamın belirlenmesi, planlama, tasarım, dağıtım/yüklenme/yaygınlaştırma, çalıştırma, izleme ve ihtiyaç kalmayan kod parçalarının ortamdan kaldırılması, kullanıcı hak ve yetkilerinin düzenlenmesi gibi süreç sonunda yürütülen faaliyetleri içerir. Dağıtım süreç aşamaları Şekil 2'de gösterilmektedir.

Şekil 2. Uygulama dağıtım süreçleri

Uygulama, veri tabanı, işletim sistemi, kullanıcı sayısı, etkilenecek sistemler dikkate alınarak *kapsam* belirlenir. Dağıtımda yapılacak işlerin listesi, uygulanmasında karşılaşılabilecek bir başarısızlık veya acil bir durumda yapılacaklar ve alınacak önlemler, yedekleme ve *dağıtım planı* oluşturulur. Dağıtım süreci ve öncesinde yapılacak iş, işlem, ihtiyaç ve gereksinimleri dikkate alınarak *tasarım* gerçekleştirilir. Uygulama yardımcı araçlar kullanarak belirlenen sunucu ve sistemlere *yüklenir*. Yükleme öncesinde ilgili sunucular üzerinde ihtiyaç duyulan uygulama ve sistemler erişim yetkileri de verilerek çalışmaya hazır hale getirilir. Yüklenen uygulamalar kontrollü bir şekilde *çalıştırılır*, test ve ön kontrol sonuçlarına göre ilgili sunucularda ihtiyaç duyulmayan kod parçaları silinir, erişim yetkileri gözden geçirilerek uygulama yaygın kullanıma açılır. Canlı (PROD) ortam üzerinde çalışmaya alınan uygulamalar izleme araçları yardımıyla *izleme*ye alınır. Uygulamalar, yeni bir talep ya da güncelleme gelmediği sürece çalışır.

1.3. Dağıtım Araçları

Geliştirilmesi tamamlanmış olan bir uygulamanın çalışacağı ortama manuel yöntemlerle yapılan dağıtım ve yaygınlaştırmada yaşanan problemlere karşı dağıtım sürecinin otomatik dağıtım araçları yardımıyla gerçekleştirilmesi bu sürece pozitif katkı sunar. Dağıtım sürecinin otomatikleştirilmesine yardımcı birçok uygulama yazılım aracı vardır. Yaygın kullanılan uygulamalara ait liste Tablo-1'de sunulmaktadır.

Tablo 1: Yazılım Dağıtımının Otomatikleştirmesinde Kullanılan Uygulama Araçları

Uygulama	**Açıklama**	**Kaynak**
AWS CodeDeploy	Amazon EC2 bulut sunucuları ile kurumsal ağ ya da herhangi bir bulut sunucusuna yazılım dağıtımını otomatikleştiren bir hizmet aracıdır.	(AWS Code Deploy, 2022)
Azure DevOps	Kullanıcıların sürüm tanımlama, otomasyon çalıştırma, sürüm izleme imkânı sunan, farklı platformlara aynı anda dağıtım yapabilen, Windows, Linux veya Mac işletim sistemlerinde derlenebilen açık kaynak kodlu bir uygulama aracıdır.	(DevOps, 2022)
Bamboo	Farklı ölçekteki yapılarda koddan devreye almaya kadar otomatikleştirilmiş iş akışlarıyla güvenli ve ölçeklenebilir bir çözüm sunan dağıtım uygulamasıdır.	(Bamboo, 2022)
Capistrano	Ruby'de yazılmış, farklı dilleri destekleyen açık kaynak kodlu bir uzak sunucu otomasyon dağıtım aracıdır.	(Capistrano, 2022)
CircleCI	Esnekliğine, güvenilirliğine ve hızına önem veren bir CI çözümüdür. GitHub, GitLab SaaS veya Bitbucket kullanarak derlemeleri birçok ortamda otomatikleştirme yeteneklerine sahiptir. Farklı dil ve uygulamaları destekler.	(CircleCI, 2022)
Codar	HP'nin sürekli dağıtım çözümüdür. Bir paketi dağıtıldıktan sonra aynı şablonu kullanarak paketin daha yeni bir sürümlerinde kullanmayı hedeflemektedir.	(Codar, 2022)
CloudBees	Kurumsal boyuttaki gereksinimler için tasarlanmış esnek, ölçeklenebilir ve yönetilen sürekli dağıtım çözümüdür.	(CloudBees, 2022).
DeployBot	Çoklu ortamlar için manuel veya otomatik dağıtımlara izin verir, aynı anda birden fazla sunucuya dağıtma yeteneğine sahiptir.	(DeployBot, 2022)
Digital.ai	Uygulama dağıtımını herhangi bir teknoloji ortamına göre otomatikleştiren, dağıtım hızı ve ölçeklenebilirliğini artıran, model tabanlı, aracısız bir dağıtım otomasyon çözümüdür.	(Digital, 2022
GoCD	Uygulamanın sürümünü istenilen yere istenilen zamanda güvenli ve denetlenebilir bir yapıda dağıtıma olanak sunan açık kaynaklı bir dağıtım çözümüdür.	(GoCD, 2022)
Gradle	Yazılımların daha hızlı oluşturmasına, otomatikleştirmesine ve sunmasına yardımcı bir araçtır. LinkedIn, Netflix, Adobe, android tarafından kullanılan bir yaygınlaştırma aracıdır.	(Gradle, 2022)
IBM UrbanCode	Sürekli dağıtım ve devreye alma sürecini otomasyonu sağlayan, izlenebilirlik ve denetleme yetenekleri olan bir uygulama dağıtım çözümüdür.	(IBM, 2022)
Jenkins	Yazılım oluşturma, test ve teslim etme veya dağıtma ile ilgili her türlü görevi otomatikleştirmede bağımsız kullanılabilen açık kaynaklı bir otomasyon aracıdır.	(Jenkins, 2022).
Octopus Deploy	Modern DevOps sürekli yazılım dağıtan, geri bildirim alan, değişiklik yapan ve yeniden dağıtan ekiplerle çalışmak üzere tasarlanmıştır. .NET uygulamaları dağıtımını otomatikleştirmek amacıyla oluşturulmuştur.	(Octopus, 2022).
TeamCity	Esnek iş akışlarına, iş birliğine ve geliştirme uygulamalarına izin veren genel amaçlı bir CI/CD yazılım platformudur. Sürekli dağıtıma olanak tanır.	(TeamCity, 2022)
Travis CI	Projenin bulut ve şirket içi test ve dağıtıma izin veren açık kaynak kodlu bir uygulamadır. PHP, Java, Ruby, Python gibi 30'dan farkı program dilini desteklemektedir.	(Travis, 2022)

2. LİTERATÜR TARAMASI

Yazılım dağıtımı ve otomatikleştirilmesine yönelik çalışmalar ele alındığında oldukça kısıtlı sayıda yayına ulaşılabilmiştir. Bu çalışmada olduğu gibi uygulamalı bir örneğe rastlanamamakla birlikte elde edilen konuyla ilgili yakın çalışmalar aşağıda incelenmiştir.

(Poppendieck ve Poppendieck, 2007) tarafından yapılan çalışmada, bir kurumda yalnızca tek bir kod satırında yapılan bir değişikliğin dağıtım süreleri incelenerek yapılan bu işlemin sonraki zamanlarda güvenilir bir şekilde tekrarlanıp tekrarlanamayacağı sorgulanmıştır.

(Arcangeli vd., 2015) dağıtık yazılım sistemlerinde otomatik dağıtıma ilişkin son teknolojilerin ele alındığı, 2000'lerden itibaren ise Linux Redhat, Microsoft Windows, Microsoft .Net, Oracle JavaBeans (2013), VMware (2008) sanallaştırma gibi farklı teknolojiler üzerinde yazılım dağıtım araçlarının kullanılmaya başladığı vurgulanmıştır. O günün koşullarında otomatik sürüm güncellemeleri, dağıtıma ait süreç yönetimleri ve teknolojik çözümleri ele almışlardır. Dağıtım otomatikleştirmede mevcut teknik çözümlerin istemci-sunucu mimarideki kurumsal ağlarda uygulama kurulumu veya kaldırma ile sınırlı olduğunu tespit etmişlerdir. 2014 yılına ait yazılım dağıtımının otomatikleştirilmesine yönelik bu sektörde Ansible, Chef, Puppet ve Octopus Deploy yapılandırma yönetim araçlarının yaygın kullanıldığını belirtmişlerdir.

(Ruiz-Rube vd., 2015) yazılım süreci dağıtımı ve değerlendirmesine yönelik çalışmada, yazılım sürecinin bir dizi teknik ve yöntemlerle sistematik yazılım üretimini teşvik ettiğini, model tasarımı, doğrulama, geçerli kılma, uygulamaya alma ve değerlendirme gibi faaliyetler içerdiğini konu etmektedir. Her yazılım projesinde uygulamanın işletime alınma süresini azaltma ve dağıtım işlemlerini mümkün olduğunca otomatikleştirmenin amaçlandığı, böylece mevcut karmaşıklığın azaltacağı sonucuna ulaşılmıştır. Bunu gerçekleştirmede iyi bilinen teknik ve uygulamaların kullanımının önemi vurgulanmıştır. Kurumların yazılım geliştirme ve bakım süreçlerinde kalitenin iyileştirilmesinde stratejilerine uygun yöntem, teknik ve uygulama araçlarına sahip olmaları gerekliliği belirtilerek yazılım süreçlerinin dağıtım ve değerlendirilmesi için bir çerçeve sunulmuştur. Otomasyona yönelik dağıtım ve değerlendirme faaliyetleri üzerine daha fazla araştırma yapılamasının gerekliliği vurgulanmıştır.

(Lascu vd., 2015) bileşen tabanlı uygulamaların otomatik devreye alınması adlı çalışmada, yazılım bileşenlerinin sayısı arttıkça özellikle karmaşık dağıtılmış eski sistemlerde bunların yönetilmesi ve otomatik devreye alınmasındaki zorluklara değinilmiştir. Günümüz bulut teknolojilerinde artık kritik bir zorluk haline geldiği, bulut tabanlı dağıtık sistemlerin heterojen yapıda çok sayıda bileşenin birleşiminden oluştuğu, bu tür karmaşık sistemlerde dağıtım sürecinin kolaylaştırılmasında otomasyon araç ve tekniklerin vazgeçilmez bir ihtiyaç olduğu öne sürülmüştür.

(Hofmann vd., 2019) robotik süreç otomasyonu (RPA) adlı çalışmada, dijital dönüşümün artmasıyla birlikte kurumsal yapılarda robotik süreç otomasyon popüler olduğunu ancak akademik araştırmaların yetersiz olduğunu, yazılım robotlarının insan tarafından gerçekleştirilen süreçleri otomatikleştireceğine vurgu yapmışlardır. Kullanımının kolay ve uyarlanabilir olması nedeniyle şirketlerin çevik projelerde yazılım robotlarını tasarlayarak uygulayabileceklerine, böylece şirket organizasyonuna, BT strateji ve yönetişim yapılarına, yönetim sistemlerine doğrudan ve dolaylı ektilerini ele almışlardır. Sonuç olarak kuruluşların, süreç performansı, verimlilik, ölçeklenebilirlik, denetlene bilirlik, güvenlik, uygunluk ve uyumluluk hedeflere ulaşmada RPA uygulayabilecekleri sonucu yanında iş akışı otomasyonu gibi diğer otomasyon yaklaşımlarına karşı bu yaklaşımın avantaj ve dezavantajlarını da göz önünde bulundurmanın gerekliliğine dikkat çekmişlerdir.

(Ali vd., 2020) çalışmalarında sürüm notları ('*release notes*')'ın yazılım geliştirme sürecinde hayati bir role sahip olduğunu, yazılımın yeni bir sürümüne ait önemli bir işlem olmasına rağmen genellikle manuel olarak oluşturulduğunu, yazılımda yapılan lisans değişiklikleri, kullanımdan

kaldırılan kitaplıklar, yeni uygulama ara yüzü ve yazılımda yapılan değişikliklere ait açıklamalarının manuel olarak oluşturulmasının hatalara açık ve zaman alıcı bir işlem olduğunu vurgulamaktadır. Bu sorunun yazılım kaynak kod havuzundan otomatik olarak sürüm notları oluşturan çeşitli program açlarının varlığına dikkat çekmişlerdir. Araştırmacıların sürüm notlarının otomatikleştirilmesine yönelik çalışmalar yürüttükleri belirterek bunlara ait bir inceleme çalışması yürütmüşlerdir.

(Chen vd., 2020) çalışmalarında ağ işlevi sanallaştırma ortamlarında ağ hizmetlerinin otomatik dağıtım ve kontrolü üzerine çalışma yürütmüşlerdir. Bu teknolojinin hıza gelişimi ve uygulamaya alınmasıyla sanallaştırılan ortamlarda ağ hizmet sayı ve türünde ciddi artışlar olduğunu, dağıtımın manuel yapılmasının hataya açık zaman alan bir süreç olduğunu, dağıtım ve kontrol sürecinin otomatikleştirilmesine yönelik yeni mekanizmalara acil ihtiyaç olduğunu vurgulamışlardır. Bu amaçla yazılım tabanlı ağ hizmeti dağıtım ve kontrolü için 'SDNSDC' adında yeni bir mekanizma önermişlerdir. SDNSDC'nin bir örneğini Linux konteyner teknolojisi üzerinde uygulaması sonucunda özelleştirilmiş ağ servislerinin verimli bir şekilde dağıtabildiği ve davranışlarını doğru bir şekilde kontrol edebildiğini ileri sürmüşlerdir.

(Mohd Nordin vd., 2021) yazılım geliştirme verimlilik modeli değerlendirmesinde uzman görüşlerinden yararlanmıştır. Çalışmada günümüz rekabet ortamında firmaların üretim maliyetlerini düşürürken kalitenin artırılması gerekliliği belirtilerek yazılım geliştirme giderlerini azaltmada en etkili stratejinin verimliliği artırmak olduğu vurgulanmıştır. Bulut bilişim ve SaaS'ın yazılım geliştirme üretkenliğini artırmaya yardımcı olabileceği, literatür çalışması sonucunda üretkenliği etkileyen 19 faktör belirlediği, bunların uzman görüşlerine göre değerlendirmesi gerçekleştirilmiştir. Çalışmada, uzmanların 17 alt faktör ile üç ana faktör üzerinde hemfikir olduğu sonucuna ulaşılmıştır.

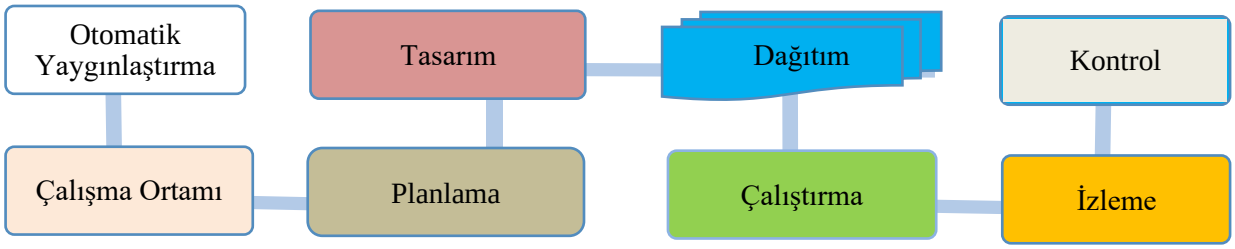
(Zai vd., 2021) çalışmalarında, son yıllarda bulut bilişimde yaşanan hızlı gelişmelere karşın bulut bilişime dayalı bilgisayar uygulamaları için otomatikleştirilmiş bir dağıtım sisteminin tasarımını tartışıldığını, dağıtım stratejisinde tahsis stratejisi, sistem esnekliğini artırmada uç sanal sunucuların kaynak kullanım performansını artıran yapılandırma ve uygulama hizmet kalitesini artıracağını vurgulamışlardır. Bileşen tabanlı programların dağıtım problemini biçimlendirmede dinamik ölçeklendirme algoritmalarına dayalı verimliliği optimize eden çalışma deney sonuçlarına göre önceki veri merkezlerine göre bulut veri merkezlerinin başarım oranı %87'nin üzerinde bulunmuştur. Araştırma sonucuna göre dağıtım sistem kullanıcılarının belirli uygulama gereksinimlerini karşıladığı, bazı mevcut sistemlere kurulum ve dağıtım yapılabildiği, bulut bilişim ortamının özelliğine göre daha iyi ölçeklenebilir sonuçlar elde edilebileceği sonucuna ulaşılmıştır.

(Zhang vd., 2021) çalışmalarında, endüstriyel internet platform mekanizmasında yaşanan çökelme sorununun hayati öneme sahip olduğu, çok dilli yapısı, çeşitli çalışma modunun varlığı bunların dağıtım ve yönetiminde birçok zorluğa ve tekrarlı çalışmalara neden olduğu vurgulanmaktadır. Bu sorunun çözümünde OpenStack tabanlı, çok dilli bir endüstriyel mekanizma modelinde otomatik dağıtım şeması önerilmektedir. Dağıtım ve yönetimin çeşitliliğine göre geliştirmede talep ve otomatik dağıtım etkileşimine göre özel bir dağıtım ortamı oluşturmanın kolay ve geliştiricilerden tasarruf eden, aynı zamanda iş yükü karşısında çökmelere karşı endüstriyel eko sistem inşası ve endüstriyel gerçekleştirilmesine iyi bir destek olacağı sonucuna ulaşılmıştır.

DergiPark akademik sistemi (DergiPark, 2022) üzerinde yayınlanan akademik çalışmalar ile lisans ve lisansüstü tez çalışmalarını da içeren Piri Keşif aracı (PiriKesif, 2020) üzerinden 'yazılım dağıtımı', 'otomatik yazılım dağıtımı', 'otomatik dağıtım aracı' ve 'sürüm dağıtımı' anahtar alanları ile yapılan Türkçe literatür kaynakların varlığına dair yapılan incelemede herhangi bir çalışmaya ulaşılamamıştır.

3. MATERYAL VE METOT

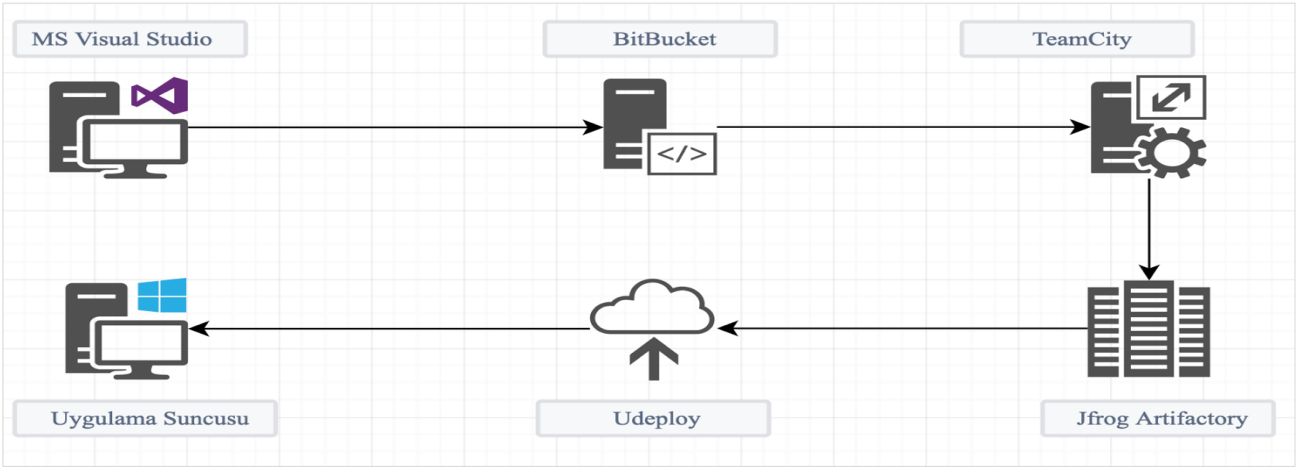
Bu çalışma, geliştirilen yazılımların dağıtımında manuel yaygınlaştırma yönteminin kullanımından kaynaklı sorunların çözümünde yararlanılan otomatik dağıtım yöntemini ele almaktadır. Otomatik yaygınlaştırmada kullanılan temel terim ve kavramlar açıklanarak süreç tasarlanmıştır. Uygulama aşamasında manuel dağıtımı yapılan bir web yazılım paketinin otomatik dağıtım süreci çalışılmıştır. Otomatik dağıtım aracı olarak yaygın kullanılan IBM UrbanCode Deploy (UDeploy) uygulaması tercih edilmiştir. Uygulanmasında, UDeploy üzerinde yeni bir proje oluşturularak çalışma amacına uygun senaryo üzerinden planlama, tasarım, dağıtım, çalıştırma ve izleme aşamaları yürütülmüştür. Otomatik dağıtım süreci, kullanılan yöntem, tasarım ve uygulama örnekleriyle birlikte sunulmaktadır. Çalışma ana süreç aşamaları Şekil 3'te gösterilmektedir.



Şekil 3. Çalışma ana süreç aşamaları

3.1. Çalışma Ortamı, Varsayım ve Sınırlılıklar

Bu çalışmada *IBM Urban Code Deploy* uygulama aracı kullanıldı. Uygulama sunucuları üzerinde *UDeploy* servisleri kurularak kullanıcı rol ve yetki tanımlandı. Uygulama kaynak kodları Bitbucket sunucuları üzerinde kullanılabilir şekilde hazırlandı. Derleme aşamasında kullanılan *TeamCity, .Net platformu, Microsoft SQL sunucu* veri tabanı uygulamalarının uygulama sunucusu üzerinde kurulu olduğu, dağıtım çıktılarının gönderileceği ve saklanacağı *Jfrog Enterprise Artifactory* platformunun çalışır şekilde hazır olduğu var sayıldı. *TeamCity* üzerinden derlenen paketin dağıtımı için *Jfrog* platformunda erişim yetkileri tanımlandı. Uygulamanın farklı uygulama ve *VT* sunucu ortamlarına ortak görevler üzerinden dağıtımı sırasında kullanılan dosya sunucusu ('*file server*'- *FS*) üzerinde lisanslı UDeploy servisi kurulmuştur. Ayrıca veri tabanı dağıtım sürecinde kullanılacak *SQL Server Data Tools (SSDT*) aracı *FS* üzerine kurulu şekilde hazırdı. Uygulama, veri tabanı ve dosya sunucuları üzerinde *PowerShell* ve. *NET Framework, .NET Core, .NET 5* ve üzeri sürümlerin kurulu ve kullanılabilir olduğu varsayılmaktadır. Çalışma ortam ve ilişkileri Şekil 4'te sunulmaktadır.



Şekil 4. Çalışma ortamı

Kullanılan araçların sistemsel özellikleri; ProLiant DL380 Gen10 Sunucu, Windows Server 2016 64 bit işletim sistemi, Intel(R) Xeon(R) Gold 5118 işlemci, CPU @ 2.30 GHz, x64 tabanlı

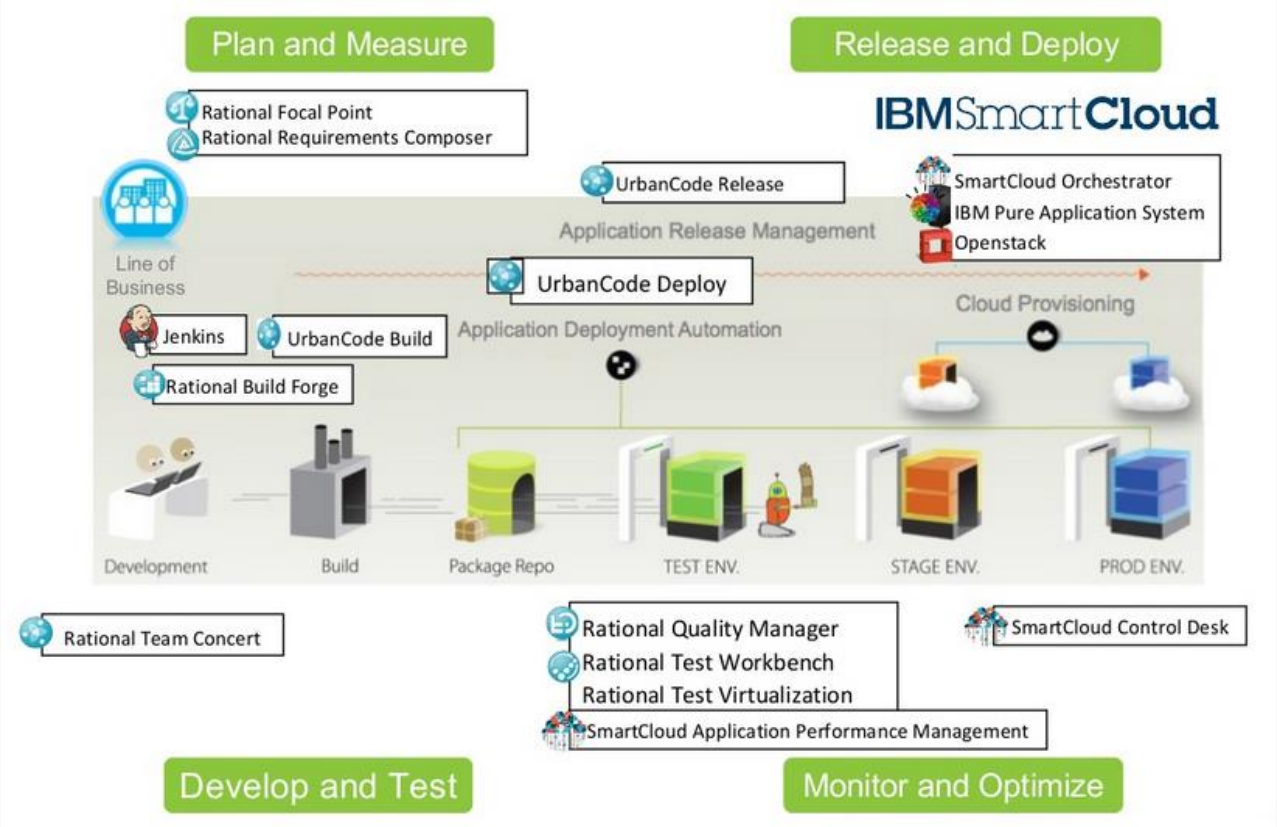
işlemci mimarisine ve 96 GB yüklü belleğe (RAM) sahip olan bir bilgisayar üzerinde gerçekleştirilmiştir. Uygulama geliştirme, kodlama ve test işlemlerinde Microsoft Visual Studio 2019 Enterprise Edition editörü ve C# 64 bit programlama dili kullanılmıştır.

Manuel dağıtım konusunda uzman bir insan kaynağı çalışmaya dâhil edilemediği için standart el yordamıyla(manuel) bir dağıtımın ortalama tamamlanma süresi hesaplanamamıştır. Bu nedenle süre bakımından otomatik dağıtım süreci ile kıyaslama yapmak mümkün olmamıştır.

4. UYGULAMA

4.1. Uygulama Mimarisi

IBM UrbanCode Deploy, uygulama dağıtım ve çalıştırma süreçlerinde yükleme, çalıştırma, izleme ve kontrol işlevleri olan bir yayımlama çözümüdür. Uygulamaların dağıtımını otomatikleştirebilen derleme ve test araçlarını içerir. Şirket içi, bulut ve ana bilgisayar uygulamalarıyla uyumlu çalışır. Bu uygulamaya ait dağıtım mimarisi Şekil 5'de gösterilmektedir.

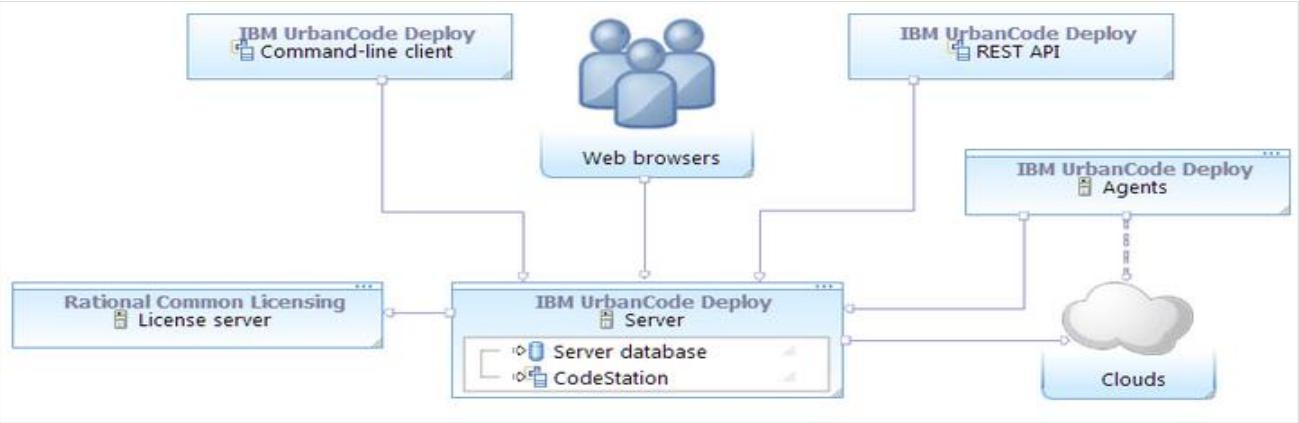


Şekil 5. Dağıtım mimarisi (UrbanCode, 2022)

Geliştirilen veya güncellenen yazılımlar 'Jenkins', 'TeamCity', 'MS TFS Build' gibi derleme araçları ile derlendikten sonra dağıtımı yapılacak bir depolama kaynağına transfer edilir. IBM udeploy aynı zamanda kendi depolama kaynağına sahiptir. Ancak bu çalışmada harici bir depolama kaynağı tercih edilmiştir. Derlenen uygulama kodu, dağıtıma hazır bir yazılım çıktısı olarak depolama kaynağında saklanır. Dağıtım aşamasında, IBM UrbanCode Deploy aracı önceden dağıtım hazırlanan dağıtım diyagramına göre süreçleri çalıştırır, dağıtımı yapılacak uygulamayı test, üretim (PROD) gibi önceden hedeflenen ortamlarda gerçekleştirir.

Uygulamanın temel kurulumunda bir sunucu, ajanlar ve lisans sunucu ihtiyacı vardır. Kullanılan ajanların iki tipi vardır. Uygulamanın sürüm 7 ve sonrasında, sunucuya WebSocket ve güvenli (*https*) bağlantı yöntemini kullanır. Önceki sürümlerinde ise JMS ve güvenli (*https*) bağlantısı kullanır. Bu ajanlar bulut ortamlarına, sanal makinelere (VM'ler), kapsayıcılara veya fiziksel

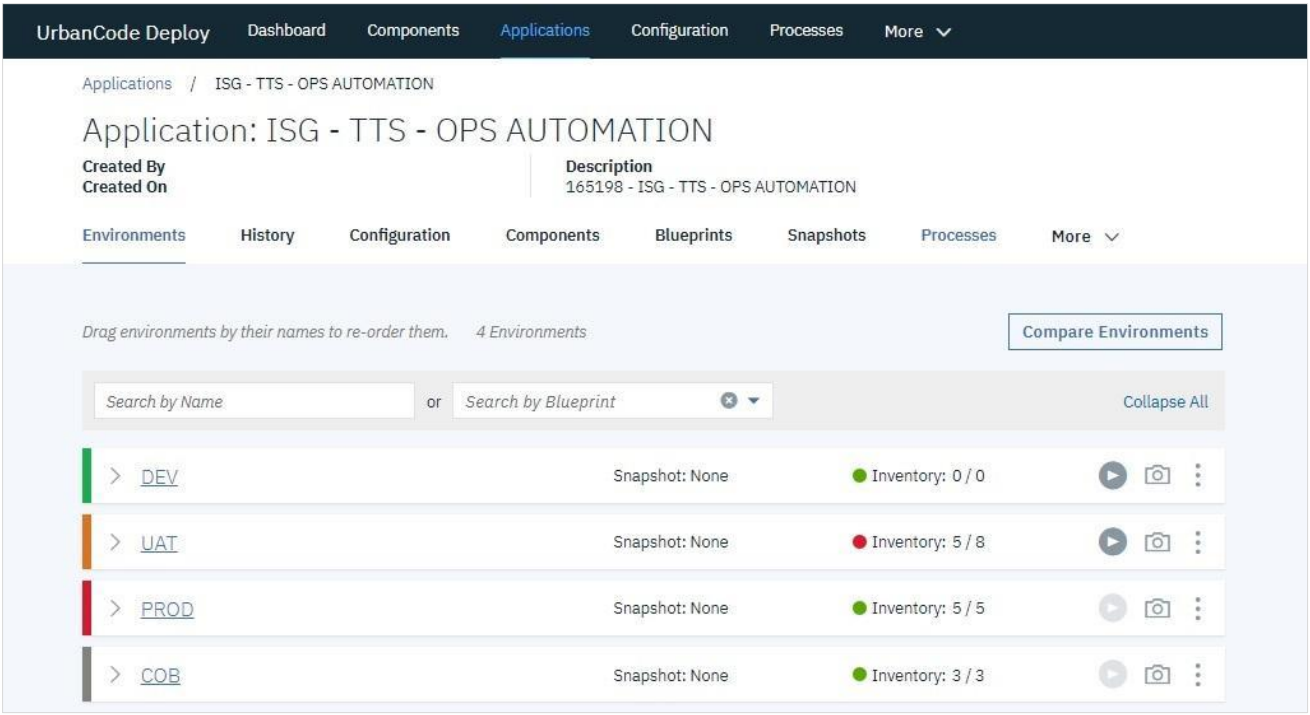
sistemlere yüklenebilir, birçok farklı türdeki sisteme kurulabilir. Uygulamaya ait çekirdek topolojisi Şekil 6'da gösterilmektedir.



Şekil 6. Çekirdek topoloji (IBM_Urbancode, 2022)

4.2. Uygulama Genel Görünümü

UrbanCode Deploy dağıtım uygulaması çalıştırıldığında Şekil 7'de gösterilen şekilde bir ekran ara yüzü elde edilmektedir.



Şekil 7. Uygulama genel görünümü

Uygulama kullanım kılavuzu, entegrasyon, destek gibi ana başlıklar yanında oluşturulan bir projeye ait uygulama geliştirme ortamı (DEV), kullanıcı kabul testi (UAT), üretim ortamı canlı (PROD) ve iş sürekliliği (COB) çalışma ortamları yer almaktadır. Yine uygulamaya ait dağıtım geçmişi, konfigürasyon yönetimi, bileşen tanımları, takvim, değişiklikler, geçmiş işlemler gibi alt menü başlıkları yer almaktadır.

4.3. Planlama

Bu çalışmada otomatik dağıtım aracı olarak yaygın kullanılan IBM UrbanCode Deploy (UDeploy) uygulaması kullanılmıştır. Bu başlık altında yetkilendirme, uygulama bileşen ve konfigürasyonu gibi uygulama alanlarına ilişkin plan ve ön tanımlamalar gerçekleştirilmiştir.

4.3.1. Yetkilendirme

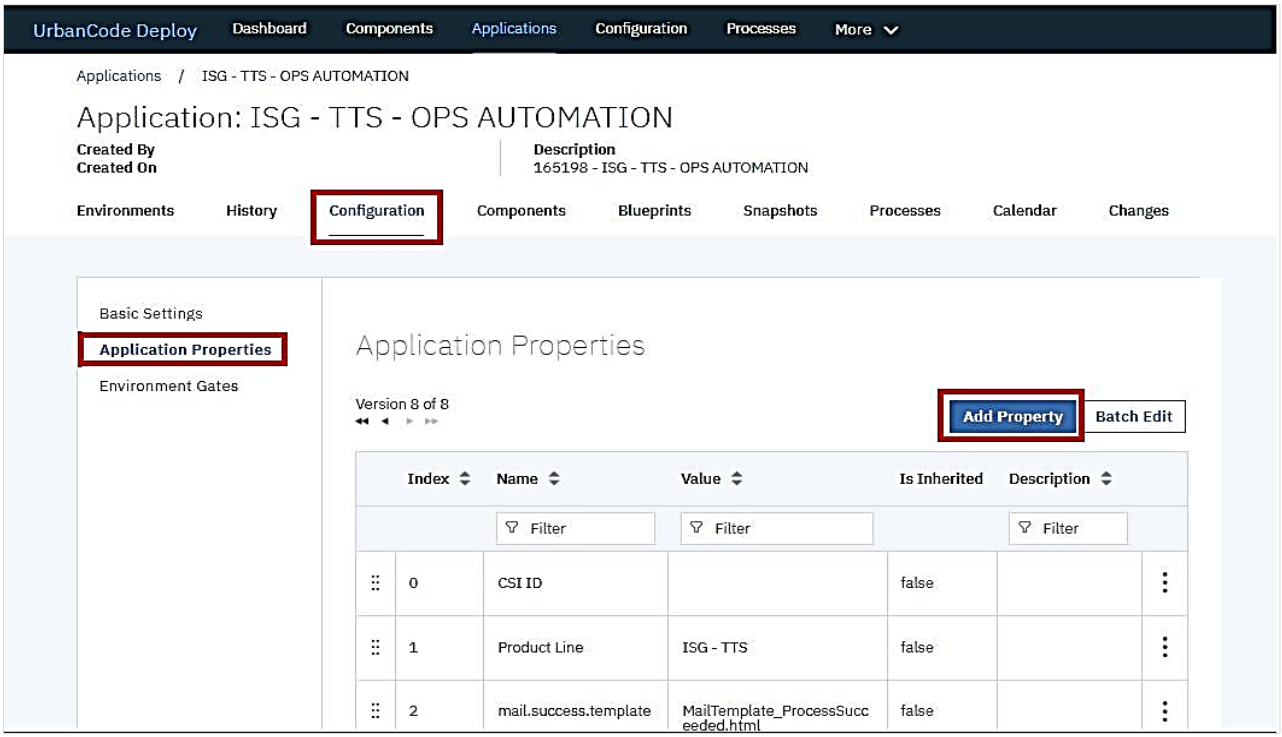
Bir dağıtım sürecinde ihtiyaç duyulan uygulama kullanıcı ve yetkilerine ilişkin örnek rol ve tanımları Tablo 2'de sunulmaktadır.

Tablo 2: Kullanıcı Yetkilendirme Rol ve Tanımları

Rol	Yetki Tanımı
Yönetici ('*Udeploy Admin*')	En üst yetkiye sahip kullanıcı rolüdür. Yeni bir uygulama oluşturma, diğer rollerin yönetimi ile canlı üretim(*production*) ve COB ortam sunucu ilişkilendirmelerini gerçekleştirir.
Konfigürasyon Yöneticisi ('*Configuration Manager*')	Süreç tasarımı, yapılandırma tanımlama ve düzenleme, uygulamanın dağıtımı ve UAT ortamına onay verme yetkisine sahiptir. Ancak canlı üretim ve COB ortamına dağıtım başlatamaz ve onaylayamaz.
Uygulama Yöneticisi ('*Application Manager*')	UAT ortamına dağıtım(*deployment*) başlatma ve onay yetkisi bulunur. Süreç tasarım ve yapılandırma, Canlı ve COB ortamına dağıtım yetkisine sahip değildir.
Ürün Destek ('*Production Support*')	Canlı ve COB ortamına onay gerekecek şekilde dağıtım başlatabilecek yetkiye sahip bir roldür. Süreç tasarım ve yapılandırma yapma yetkisine sahip değildir.

4.3.2. Konfigürasyon Tanımlama

Otomatik dağıtım aşamasında çalışacak görevlere ilişkin tanımlamalar '*administrator'* kullanıcısı tarafından uygulamanın 'Home/Applications/Configuration' sekmesinde yer alan 'Add Property' butonu kullanılarak tanımlanır. Bu özellikler ilgili görevin çalışma anında kullanılır. Çalışmada kullanılan örnek uygulama tanımların yer aldığı ekran görüntüsü Şekil 8'de yer almaktadır.



Şekil 8. Uygulama konfigürasyon tanımları

Uygulama özellikleri bölümünde yer alan değişkenler ve kullanım amaçları Tablo 3'de açıklanmaktadır. Yeni bir nitelik ekleme '*Add Property*', daha önce tanımlanmış bir özelliği güncelleme '*Edit*', silme ise '*Delete*' alanları üzerinden gerçekleştirilir.

Tablo 3: Uygulama Değişkenleri

Özellik İsmi	Açıklama
CSI ID	Uygulamanın kurum envanter kaydı numarası
mail.alldisable	E-posta gönderim özelliğinin aktif - pasif yapılması
mail.allfrom	Gönderilecek e-postanın 'kimden' alanı
mail.allsmtp	E-posta SMTP sunucu adresi
mail.allto	Gönderilecek e-postanın 'kime' alanı
mail.approved.subject	E-posta 'başlık' şablonu
mail.approved.template	E-posta onaylanmış dağıtım için bildirim maili 'gövde' şablonu
mail.fail.subject	Hata almış dağıtım işleminin bildirim e-posta 'başlık' şablonu
mail.fail.template	Hata almış dağıtım için bildirim e-posta 'gövde' şablonu
mail.rejected.subject	E-posta onaylanmamış dağıtım için bildirim e-posta gövde şablonu
mail.rejected.template	Onaylanmamış dağıtım için bildirim e-posta 'gövde' şablonu
mail.review.subject	Dağıtımında çalıştırılacak kod parçasının gözden geçirme bildirim başlığı
mail.review.template	Dağıtımında çalıştırılacak kod parçasının gözden geçirilme bildirim içeriği
mail.start.subject	Başlatılan dağıtım işleminin bilgilendirme e-postası için başlık şablonu
mail.start.template	Başlatılan dağıtım işleminin bilgilendirme e-postası için gövde şablonu
mail.success.subject	Başarılı tamamlanan dağıtım işleminin bilgilendirme e-postası için başlık şablonu
mail.success.template	Başarılı tamamlanan dağıtım işleminin bilgilendirme e-postası için gövde şablonu
Product line	Uygulamanın kurum envanterinde bağlı bulunduğu bölüm

4.3.3. İş Akışlarının Planlanması

Bu çalışmada iş akışlarının uygulamada var olan hazır ya da hazırlanan şablonlardan yararlanarak oluşturulması planlanmıştır. Aynı tür bileşenlerin dağıtım ihtiyaçlarının merkezi bir şekilde yönetimi amaçlanmıştır. Ana şablonda yapılacak bir güncellemenin bu şablondan türetilen iş akışlarına otomatik bir şekilde uygulanabilmesine imkân sunacak bir yapı kurgulanmıştır.

Bu çalışma kapsamı bir web uygulamasının farklı ortamlara dağıtımının otomatik yapılması ile sınırlandırılmıştır. Bu amaçla IIS, WEB, UD iş akış şablonları oluşturulmuştur.

İnternet bilgi servisi (IIS): Bu çalışmada uygulamanın internet üzerinden servis edileceği sunucu Microsoft IIS olarak planlanmıştır. Otomatik dağıtımda, IIS üzerinden yapılacak görevler için planlanan iş akışları Tablo 4'te sunulmaktadır.

Tablo 4: Microsoft IIS İş Akışları

İş Akışı İsmi	Açıklama
Backup Configuration	Mevcut IIS yapılandırma yedeğinin alınması
Configure Site	İnternet sitesinin yapılandırılması
Create Site	Yeni bir internet sitesinin oluşturması
Delete Site	Mevcut internet sitesinin silinmesi
Recycle Site	İnternet sitesinin yeniden başlatılması
Start Site	İnternet sitesinin başlatılması
Stop Site	İnternet sitesinin durdurulması

İnternet (WEB): İnternet uygulamasının otomatik dağıtımında kullanılacak görev ve işlemlere ait akışların oluşturulmasında hazır WEB şablonlarının kullanılması planlanmıştır. Bu görevler IIS haricinde uygulamaya özel ihtiyaçlardan türetilmiştir. Bu aşamada planlanan işlemler ve kullanım amaçları Tablo 5'te tanımlanmıştır.

Tablo 5: WEB Dağıtım İş Akışları

İş Akışı İsmi	Açıklama
Add to Inventory	Mevcut versiyonun envanter kaydına eklenmesi
Backup Artifacts	Uygulamanın aktif sürümünün yedeğinin alınması
Clean Backups	Önceden alınmış yedeğin temizlenmesi
Deploy Artifacts	Dağıtım işlemi
Encrypt Config	'web.config dosyası'nın şifrelenmesi
Remove from Inventory	Envanter kaydından silinmesi
Replace Configuration Tokens	Ortama göre değiştirilecek dosyaların hazırlanması
Restore Artifacts	Daha önceki yedek üzerinden geri dönülmesi
Set FID	IIS üzerinde uygulama havuzuna 'Functional ID' tanımlanması.

Uygulama Dağıtım (UD): Otomatik dağıtım görevinin çalıştığı esnada e-posta bilgilendirmeleri, hata alınması durumunda hata kaydının tutulması gibi işlemlerin hazırlanmasında UD şablonları kullanılmıştır. Bu aşamada planlanan işlemler ve kullanım amaçları Tablo 6'de tanımlanmıştır.

Tablo 6:UD İş Akışları

Process	Description
Add Latest Deploy to Inventory	Son dağıtıma ait sürümün döküm kaydı.
Check Process Failure	İş akışındaki hata kontrolü
Create Snapshot	Mevcut sürüm bilgisinin kayıt altına alınması
Get Request Details	Dağıtım isteği detay bilgisi
Send Notification Mail	Bilgilendirme mailinin gönderilmesi
Set Process Failure Flag	İş akışındaki hata kontrolü bayrağına değer atanması
Set Process Start Time	İş akışının başlangıç zamanının kaydedilmesi

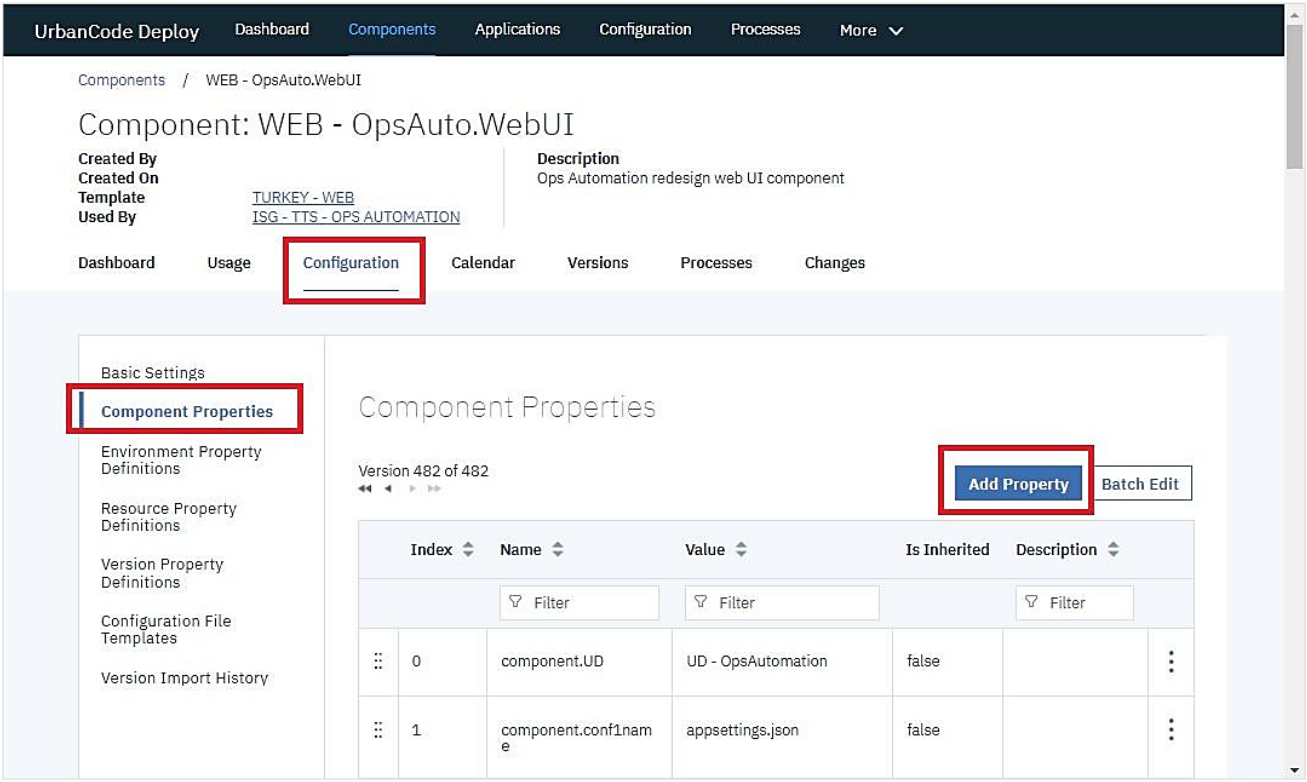
4.4. Tasarım

Bu çalışmanın bileşen('*component*') ve iş akış tasarımları UDeploy uygulaması üzerinden gerçekleştirilmiştir. Bileşen kavramı, otomatik dağıtım yapılacak her bir tipteki uygulama ya da nesneyi ifade etmektedir. Bileşenleri ortamlara dağıtmak için uygulama süreçleri ('*application process*') tasarlanır ve daha sonra çalıştırılır. Bileşen ve uygulama dağıtım tasarımı bu kısımda ele alınmaktadır.

4.4.1. Bileşen Tasarımları

Bileşen, dağıtımı yapılacak olan her bir uygulama parçacığının ayrılması için kullanılmaktadır. Bileşenler daha önceden hazırlanmış olan şablonlardan türetilmiştir.

İnternet (WEB) Bileşeni: İnternet bileşenlerine ait iş akış tasarımları hazır şablonlar üzerinden gerçekleştirilmiştir. Dağıtımı yapılacak uygulama paketi otomatik dağıtım sürecinin başlamasından önce derleme aracı çıktısı olarak depolama amacıyla kullanılan Linux tabanlı *Jfrog Artifactory* uygulamasına yüklenir. Udeploy, periyodik aralıklarla ya da manuel kullanıcı tetiklemesiyle bu uygulamaya bağlantı oluşturarak depo yerinde uygulamanın yeni bir sürümünün bulunup bulunmadığını kontrol eder. Eğer yeni bir sürümü varsa bu sürüm otomatik olarak bileşenin güncel sürüm bilgisi kaydedilir. Uygulama sürüm bilgilerine dair ekran görüntüsü Şekil 9'da sunulmaktadır.



Şekil 9. WEB bileşen sürüm örnekleri

Web bileşenleri aynı zamanda uygulama yapılandırma bilgilerini dağıtım yapılacak ortamlara göre farklılaştırmaya katkı sunar. Örneğin: Veri tabanı bağlantı bilgisi test ve canlı ortamlar için farklı değerler içerir. Bu değerler ortam değişkenleri kısmında değişken olarak tanımlanmaktadır. Değişkenlerin değerleri ise her ortam için ayrı ayrı girilmektedir. Dağıtım aşamasında yapılandırma dosyası girilen değerlere göre değiştirilerek ilgili uygulama klasörüne kopyalanır. Kullanılan internet bileşen özellikleri Tablo 7'de, ortam değişken ve tanımları Tablo 8'de sunulmaktadır.

Tablo 7:WEB Bileşen Özellikleri

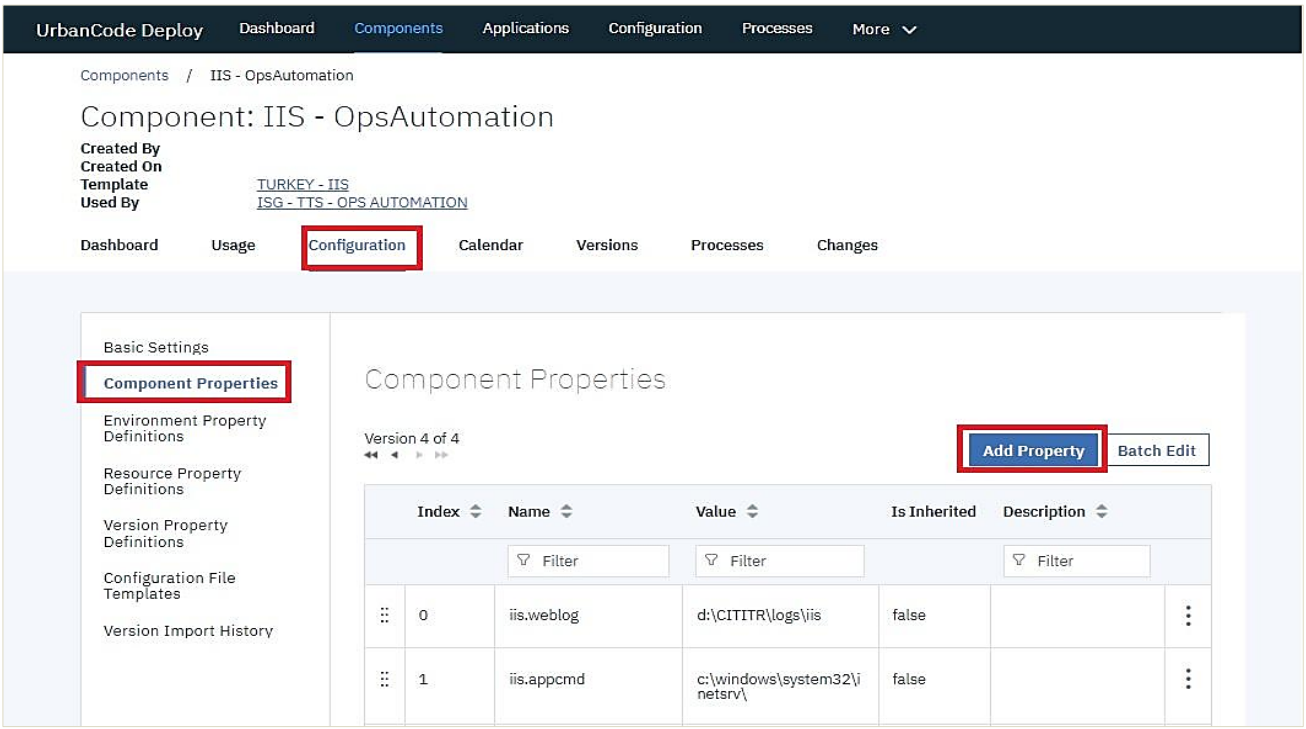
Özellik ismi	Açıklama
art.zippedartifact	Dağıtımı yapılacak uygulama paketinin sıkıştırılmış olup olmadığı bilgisi.
component.artifactoffset	Dağıtımı yapılacak uygulama dosyalarının klasör içerisindeki ofset bilgisi.
component.backupfolder	Yedek alınacak klasörün sunucu üzerindeki yolu.
component.clearstaging	Çalışma klasörünün operasyon sonrasında temizlenme seçeneği yer alır.
component.conf1diroffset	Tokenize edilecek dosyanın klasör içerisindeki ofset bilgisi.
component.conf1name	Tokenize edilecek dosyanın ismi
component.confdelimitend	Tokenize edilecek dosya içerisindeki ilgili bölümün başlangıç karakteri
component.confdelimitstart	Tokenize edilecek dosya içerisindeki ilgili bölümün bitiş karakteri.
component.latestbackuppath_COB	COB ortamında son alınan yedekleme(backup) konumu
component.latestbackuppath_PROD	Üretim (PROD) ortamında son alınan yedek konumu
component.latestbackuppath_UAT	UAT ortamında son alınan yedek konumu
component.latestbackupversion_COB	COB ortamında son alınan yedeğin sürüm bilgisi
component.latestbackupversion_PROD	Üretim (PROD) ortamında son alınan yedeğin sürüm bilgisi
component.latestbackupversion_UAT	UAT ortamında son alınan yedeğin sürüm bilgisi
component.type	Component Türü
component.UD	UD bileşen ismi
site.enable32bitapp	IIS üzerindeki uygulama havuzunun 32 bit olması bilgisi yer alır.
site.encryptconfig	web.config dosyasının şifrelenmesi bilgisi yer alır
site.IISComponent	IIS bileşeninin ismi yer alır.
site.iisfolder	Uygulamanın sunucu üzerindeki klasörü
site.managedpipeline	Uygulama havuzu üzerindeki yönetilen işlem hat bilgisi

site.managedruntime	Uygulama havuzu üzerindeki yönetim çalışma zamanı bilgisi
site.name	Uygulamanın IIS üzerindeki web site ismi
site.poolname	Uygulamanın IIS üzerindeki uygulama havuz ismi
site.rapidfailprotection	Hızlı hata koruma bilgisi
tfs.latestdeploy_COB	COB ortamına yapılan son dağıtımın sürüm bilgisi
tfs.latestdeploy_PROD	Üretim (PROD) ortamına yapılan son dağıtımın sürüm bilgisi
tfs.latestdeploy_UAT	UAT ortamına yapılan son dağıtımın sürüm bilgisi

Tablo 8: Ortam Değişkenleri

Özellik İsmi	**Açıklama**
site.certfriendlyname	İnternet sitesi sertifika tanımı
site.port	İnternet sitesi port bilgisi
site.protocol	İnternet sitesi protokol bilgisi (http/https)
site.certlocation	Sunucu üzerindeki sertifikanın konumu
site.FIDType	İnternet sitesinin çalışacağı işlevsel (functional) ID tipi
site.FIDName	İnternet sitesinin çalışacağı işlevsel ID ismi
site.FIDPassword	İnternet sitesinin çalışacağı işlevsel ID şifresi
ART_SOURCE_REPO	Dağıtım edilecek paketin 'artifactory' uygulama kaynak konumu
ART_TARGET_REPO	Dağıtım yapılacak paketin taşınacak hedef konumu
conf.environment	Kişiselleştirilecek ortam verisine ait değişken
conf.connectionString	VT bağlantı tanımı gibi kişiselleştirilecek bağlantı değişken tanımı.

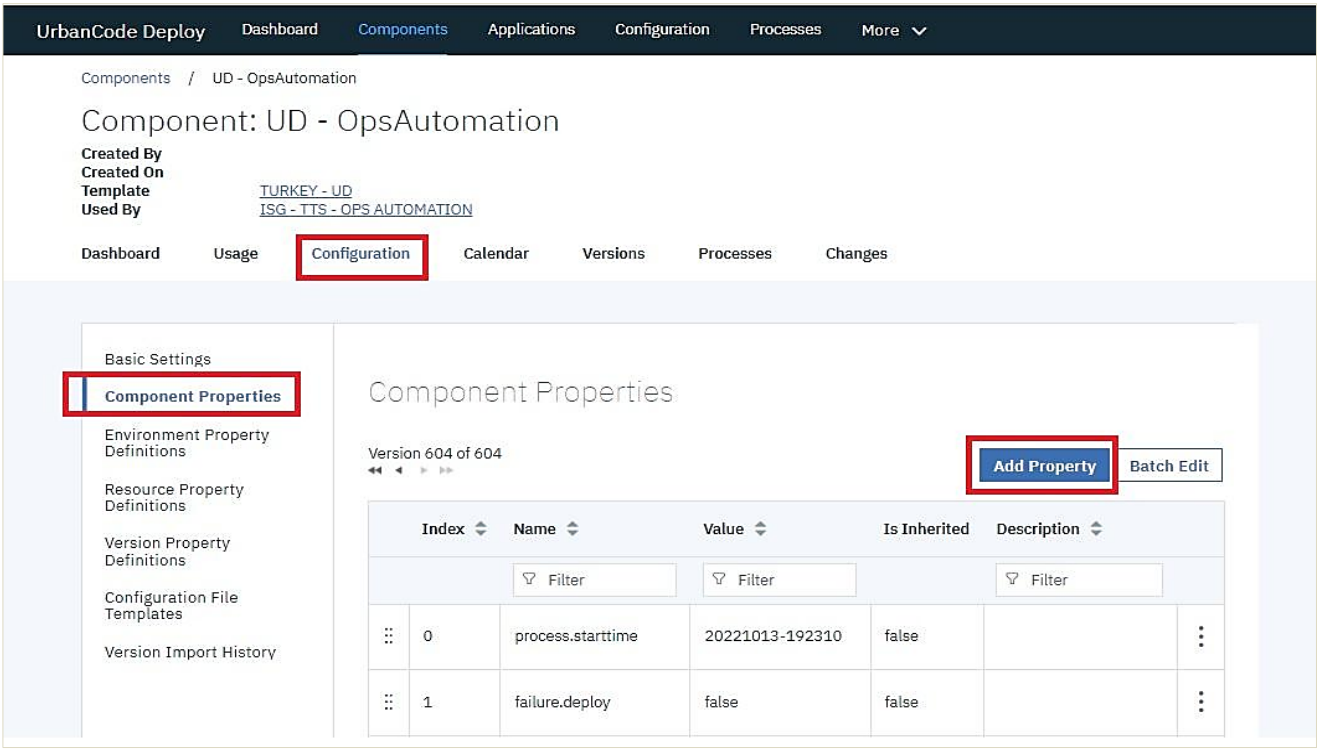
ISS Bileşeni: Uygulama dağıtım sürecinde derlenen uygulama paketinin sunucuya konumlandırılması aşamasında IIS tarafında yapılması gereken işlem adımları bulunmaktadır. Bu bileşen IIS hazır şablonu üzerinden türetildiği için şablonda tanımlanan iş akışları dağıtım aşamasında iş akışı tasarımında yer alır ve çalıştırılır. Bu bileşen üzerinde uygulama ya da parçası bulunmadığından sürüm yapısı söz konusu değildir. IIS bileşen ve tanımları Şekil 10'da yer almaktadır.



Şekil 10. IIS Bileşen Özellikleri

UD Bileşeni: Otomatik dağıtım sürecinde kullanılması planlanan yardımcı bir bileşen olarak tasarlanmıştır. Bileşen, UD iş akış şablonlarından tasarlanır. Üzerinde herhangi bir uygulama

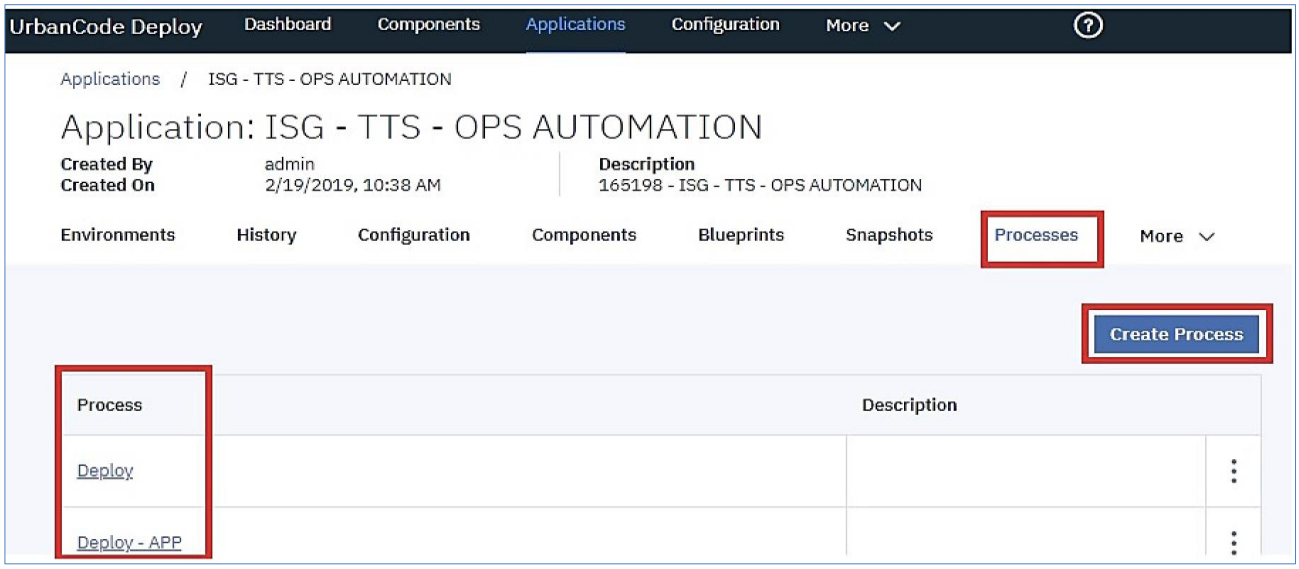
bulunmadığı için bu bileşen de yardımcı bir bileşen olup IIS bileşeni gibi sürüm yapısı söz konusu değildir. Ortam değişkenleri bulunmamaktadır. Bu bileşene ait tasarım özellikleri Şekil 11'de gösterilmektedir.



Şekil 11. UD bileşen özellikleri

4.4.2. Uygulama İş Akışı Tasarımı

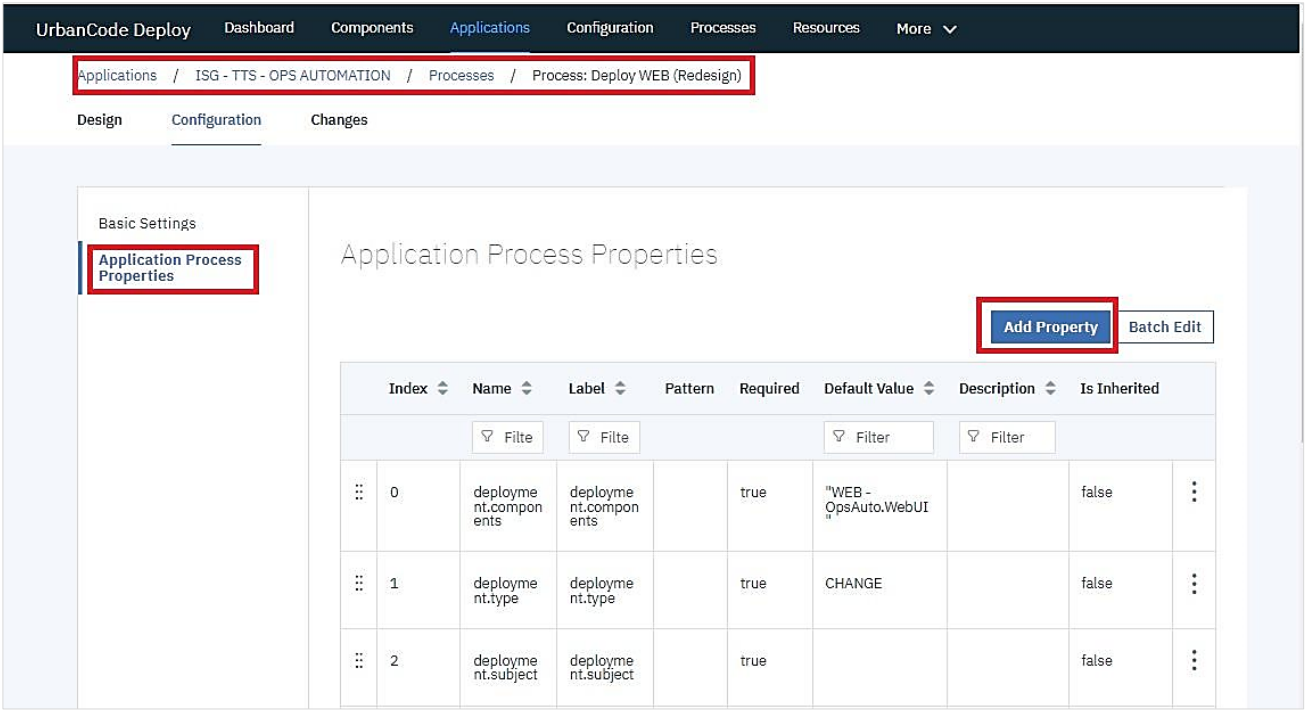
Bileşen hazır şablonları ve üzerlerinde tanımlı bulunan işlemler dağıtım sürecinin iş akışını oluşturmaktadırlar. Oluşturulan bu ana süreç ve bileşenler birbirleriyle ilişkili biçimde tasarlanır. Dağıtım aşamasında kullanılacak özelliklere ait değişkenlerin tanımı da bu bölümde gerçekleştirilmektedir. Uygulamaya ait iş akış listesinin örnek ekran Şekil-12'de gösterilmektedir.



Şekil 12. İş akış listesi

İş akışı tasarımı UDeploy süreç tasarımı menüsü bileşenler('*components*') menüsü altında yer alan süreçler ('*processes*') sekmesi '*Create Process*' düğmesi kullanılarak gerçekleştirilir. Mevcut bir

akışın güncellenme, kopyalama veya silme işlemi de bu ekran aracılığıyla gerçekleştirilmektedir. Uygulamada kullanılan iş akışı özelliklerine ait örnek ekran görüntüsü Şekil 13'te sunulmaktadır.



Şekil 13. Uygulama iş akışı özellikleri

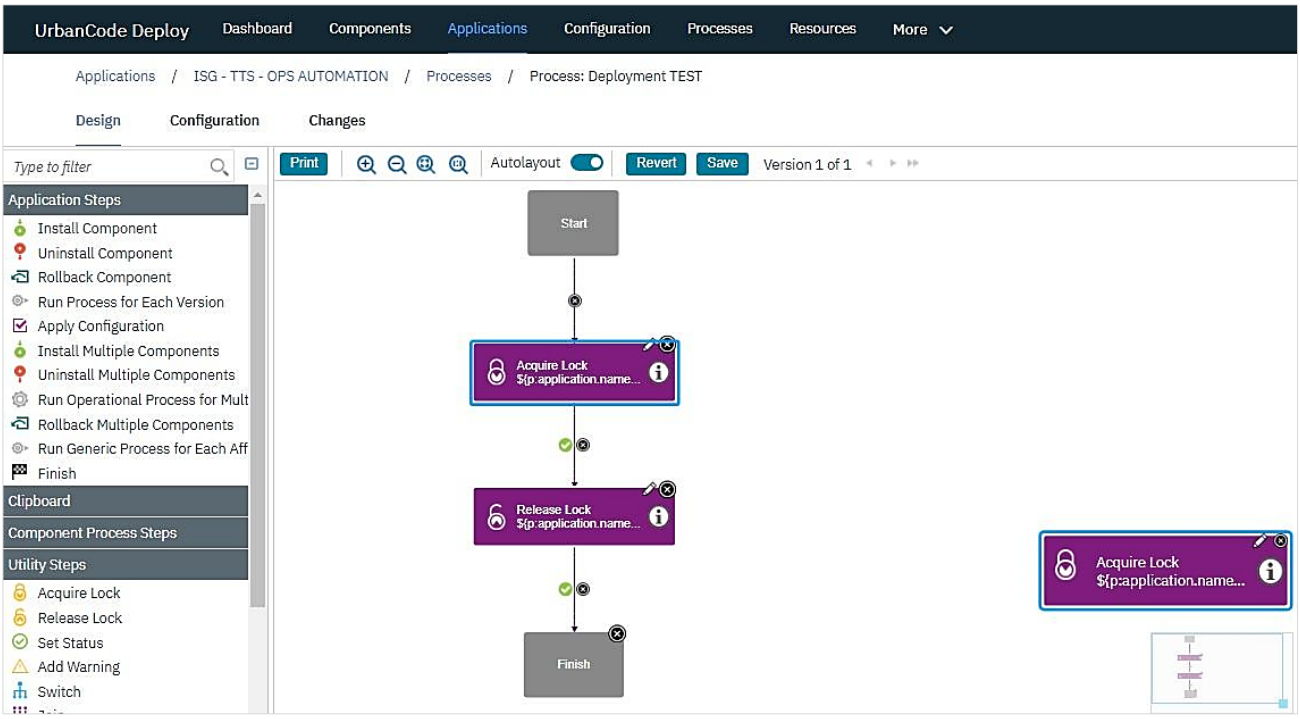
Otomatik dağıtım sürecinin tasarımında Tablo 9'da yer alan işlem adımları belirlenen sırada çalışacak şekilde tasarlanmıştır.

Tablo 9: Dağıtım Süreç İşlem Adımları

Görev İsmi	**Açıklama**
Start	Sürecin Başlangıç Adımı
Aquire Lock	Aynı sürecin birden fazla çalıştırılmasının engellenmesi
Sen Notification Mail(started)	Başlandı bilgisinin e-mail ile iletilmesi
Environment Selection	Deployment yapılacak ortamın bilgisi kontrolü
Ticket Validation	Canlı ortam için değişiklik kaydının sorgulanması.
Set Process Failure Flag	Süreçte hata takibi için bayrağın başlangıç değeri atama
Set Process Start Time	Sürecin başlangıç zamanının atanması
Create Rollback Snapshot	Mevcut sürümüne ait görüntünün kaydedilmesi
Backup	Mevcut uygulamanın yedeğinin alınması.
Stop Site	IIS üzerindeki site ve uygulama havuzunun durdurulması
Deploy	Dağıtım işlemi.
Add to inventory	Envanter kaydının güncellenmesi
Create Web	Deployment türü INITIAL olarak seçilmişse IIS altında site ve diğer bileşenlerin oluşturulması
Check Process Failure	Süreçte hata kontrolü adımı
Start Site	Deployment sonrası uygulamanın başlatılması
Send Notification Mail	Tamamlandı bilgisinin e-mail ile alıcılara iletilmesi
Release Lock	Süreç kilidinin kaldırılması ve yeni dağıtıma izin verilmesi
Finish	Sürecin bitiş adımı

Bu işlem adımları çalışırken herhangi bir hata oluşması durumunda tasarımda karar verilen önceki sürüme otomatik geri dönülür. Bu hata dağıtım işlemine başlamadan önce gerçekleşmesi durumunda uygulama henüz dağıtım gerçekleştirilmediğinden süreç hata oluştuğu bilgisi

işaretlenerek sonlandırılırmıştır. Süreç tasarımı, UDeploy ürünü üzerinde yer alan tasarım (‘*design*’) menüsü aracılığıyla yapılmaktadır. Süreç tasarımı örnek ekran görüntüsü Şekil 14’te sunulmaktadır.



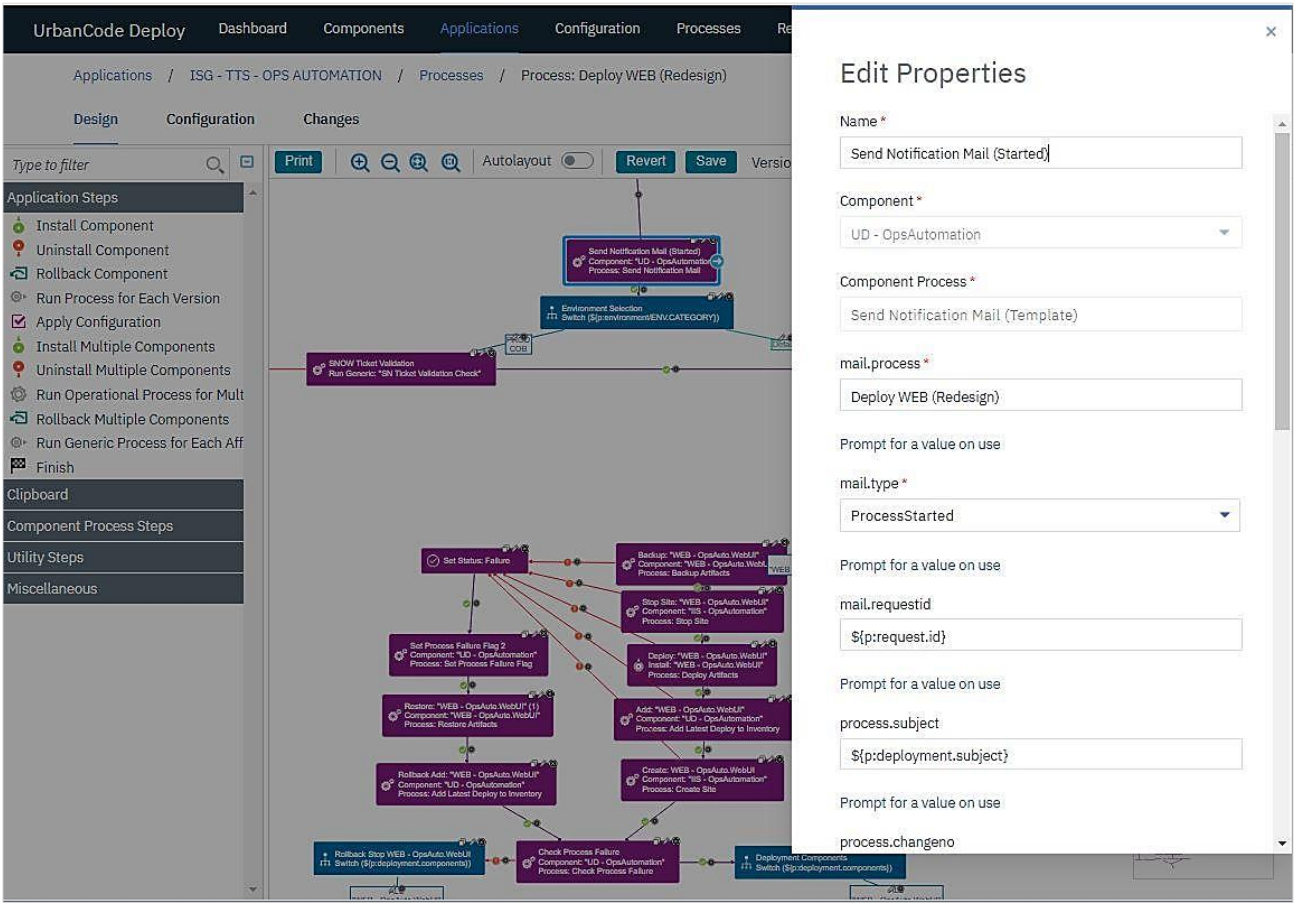
Şekil 14. Süreç tasarım örneği başlangıç bölümü

Tasarım menüsünde sunulan hazır bileşen şablonları ya da menünün sol bölümünde yer alan hazır görev listesi üzerinden sürükle-bırak yöntemi ile form üzerine yerleştirilir. Ardından bileşen üzerinde ilk olarak çalıştırılacak işleme ait bilgi tanımlanır. Başarıyla tamamlanan bir işlemden sonra süreç akışının devam edeceği bir sonraki işlem adımına bağlantı tanımlanır. Bu adıma geçişte bir hata ile karşılaşması durumunda yönleneceği görev ise bağlantı ok işaretinin ortasında bulunan ‘*check’* sembolü tıklandıktan sonra hata okuna dönüştürülerek gerçekleştirilir. Her bir süreç ‘*start*’ adımıyla başlar ve ‘*finish*’ adımıyla sonlanır.

Bir bileşen işlemi (‘*component process*’) eklendikten sonra üzerinde, görev ismi, hangi bileşene ait olduğu ve işleme ait isim bilgisi görülmektedir. Buna ait örnek görüntü Şekil 15’te yer almaktadır.

Şekil 15 Bileşen işlemine ait örnek görünüm

Tasarım aşamasında bileşene ait değişen değerlerinin girilmesi gerekmektedir. Her bir işlem sağ üst köşesinde bulunan kalem sembolü ile düzenleme işlemi yapılmaktadır. Bu değişkenlerin değerleri başka bir değişkenin değerinden atanması şeklinde olması durumunda ${p: *değişken*} şeklinde tanımlanır. Bu değişken değeri, süreç adımı başlatılırken kullanıcı tarafından girilen değerdir. Değişken yerine sabit değer verilmek istendiğinde değişken yerine sabit değer yazılır. Her iki türe ait tanımlama örneği Şekil 16’da gösterilmektedir.



Şekil 16. 'Send Notification Mail' parametre değerlerinin tanımlanmasına dair örnek ekran görüntüsü

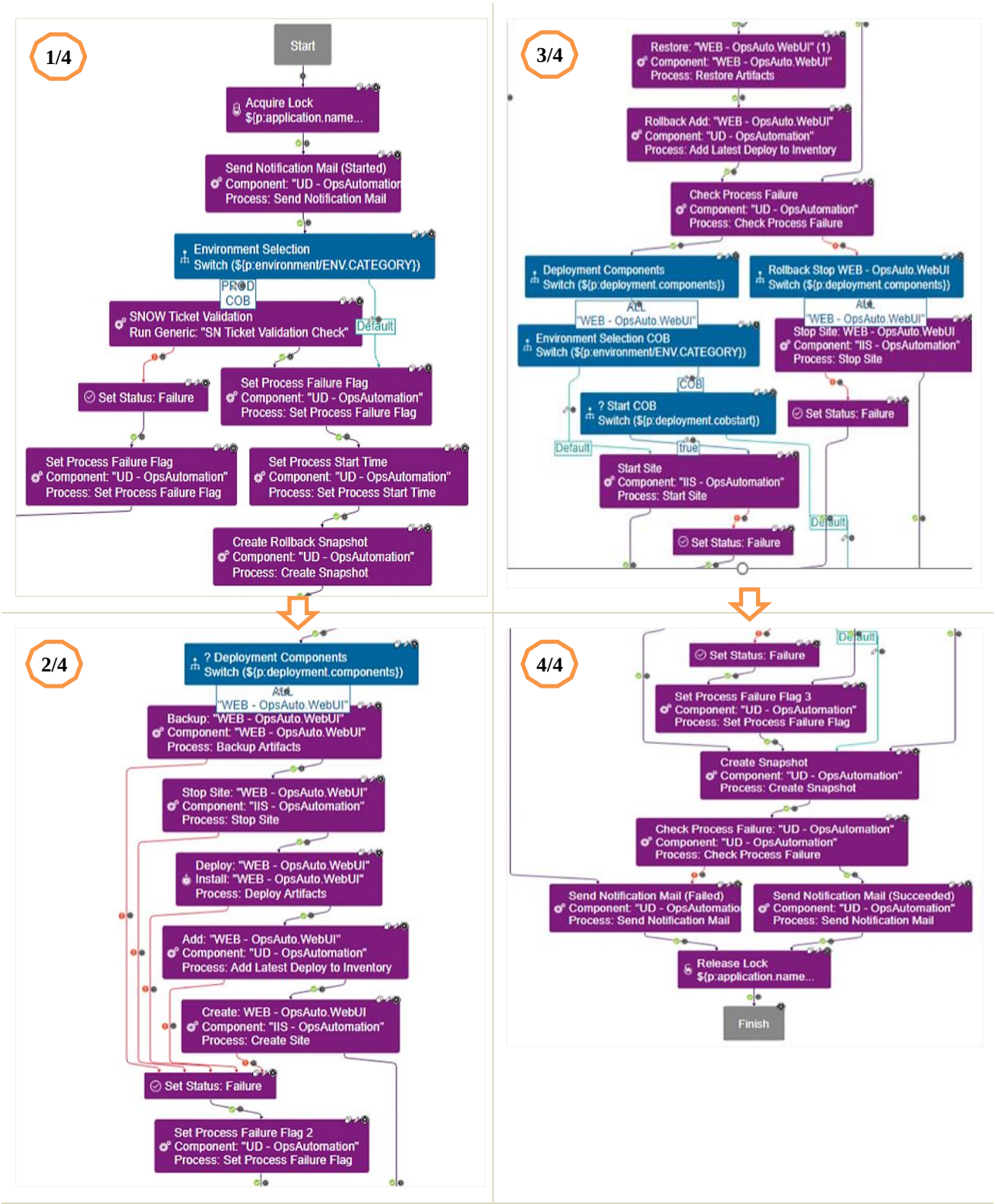
Tasarım ekranında görünen oklar ile bileşenler bir iş akışı oluşturacak şekilde birbirlerine bağlanmıştır. Ok işaretlerinin üzerinde bulunan yeşil ve kırmızı ibareler bir sonraki bileşene iş akışını yönlendirirken hata ile karşılaşıldığında ya da önceki adımın başarılı çalışması sonucu devam edeceğini ifade etmektedir.

Tasarımda karar verme ('*switch'*) bileşenleri kullanılmıştır. Bu bileşen iş akış akında, daha önceden tanımlanmış olan bir değişkenin değerine göre karar verip akışı istenilen yönde sürdürülmesi amaçlanmıştır. Örneğin, Şekil 16'da görünen '*Environment Selection'* karar verme nesnesi dağıtımı yapılacak olan ortama (UAT, COB, PROD) göre iş akışının yönlendirilmesi sağlanmıştır. Bu kullanımda, COB ve PROD için akış sol tarafta bulunan '*SNOW Ticket Validation'* adımına yönlendirilirken, diğer bütün değerler için ikinci sağdaki ok ile devam eden akışı takip edecek şekilde tasarlanmıştır.

UD Bileşeni altında tasarlanmış olan '*Send Notification Mail*' süreci, tasarım anında şablon olarak kullanılması planlanan değişkenleri içerir. Belirlenen değişken tipine göre bazı alanlar metin olarak girilebileceği gibi bazı alanların sadece belirlenen değerlerin girilmesi şeklinde tasarlanmıştır.

Değişkenlerin altlarında bulunan '*Promt for value on use'* seçeneği ise, değeri Şekil 16'da görünen özellik güncelleme ekranından girilmesi yerine, iş akışının başlatıldığı ekrandan çalıştırılma anında girileceğini ifade eder. Bu çalışmaya konu olan uygulamada sürece insan müdahalesinin mümkün olduğunca az olmasına özen gösterilmiştir. Dağıtım süreç tasarımı, eldeki kaynaklar kullanılarak olabildiğince otomatik işletilecek şekilde gerçekleştirilmiştir. Süreç tasarımı iş ve kurum hedeflerine uygun şekilde belirlenmelidir.

Çalışmada tasarımı gerçekleştirilen dağıtım sürecinin bütünü Şekil 17'de gösterilmektedir.

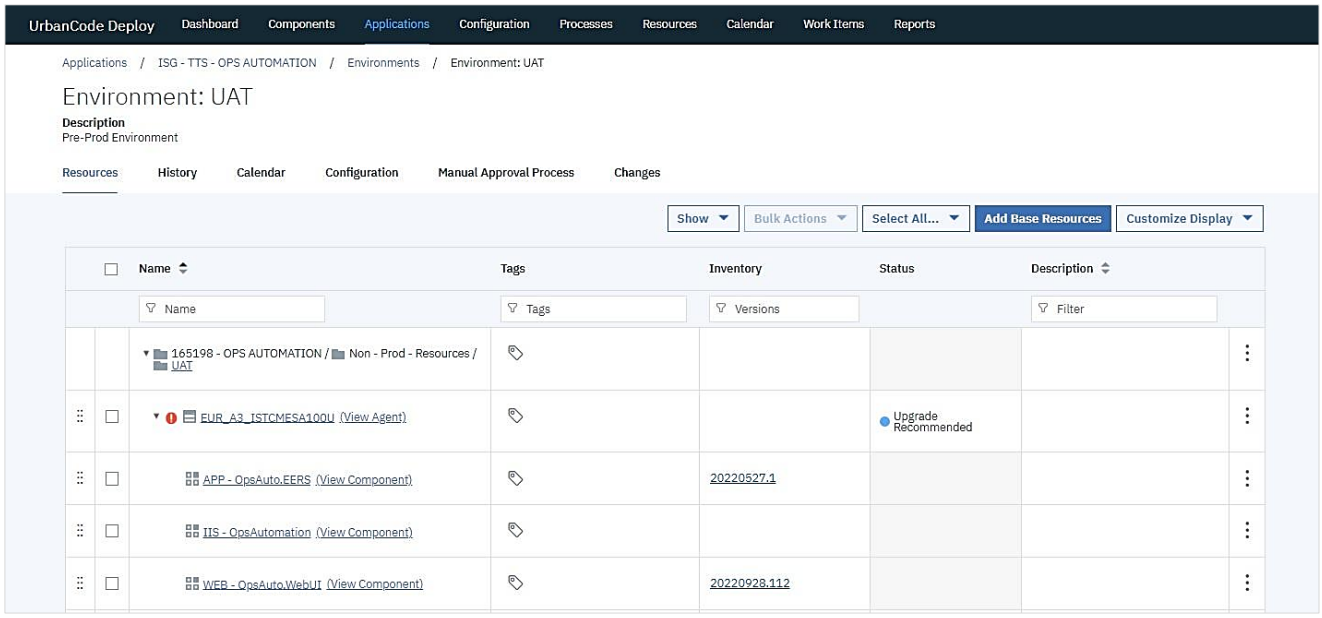


Şekil 17. Otomatik Dağıtım Uygulama Süreç Tasarımı

Süreç işlem adımlarının çalıştırılma aşamasında herhangi bir hata ile karşılaşılması durumunda hata adımlarına yönlenerek, bir önceki adımda alınan yedeği tekrar uygulama klasörüne taşıyacak (‘*restore*’) şekilde tasarım gerçekleştirilmiştir. Bir hata sebebiyle akış yedekten dönüldüğü için uygulama otomatik olarak yeniden başlatılmaz. Sürecin tasarımında bir hata ile karşılaşıldığı durumda sürecin nasıl bir yöntemle işletileceğine önceden karar verilmelidir. Bu karşılaşılan hatanın çözümlenip yeniden derlenmesi ya da yedekten değişiklik öncesi duruma geri dönüş şeklindedir.

Uygulama (‘*Application*’) menüsünden kaynak (‘*Environment / Resources*’) alt menüsü kullanılarak ilgili uygulama seçilerek açılan ekrandan sürükle-bırak yöntemi ile ilgili bağlantı tanımı

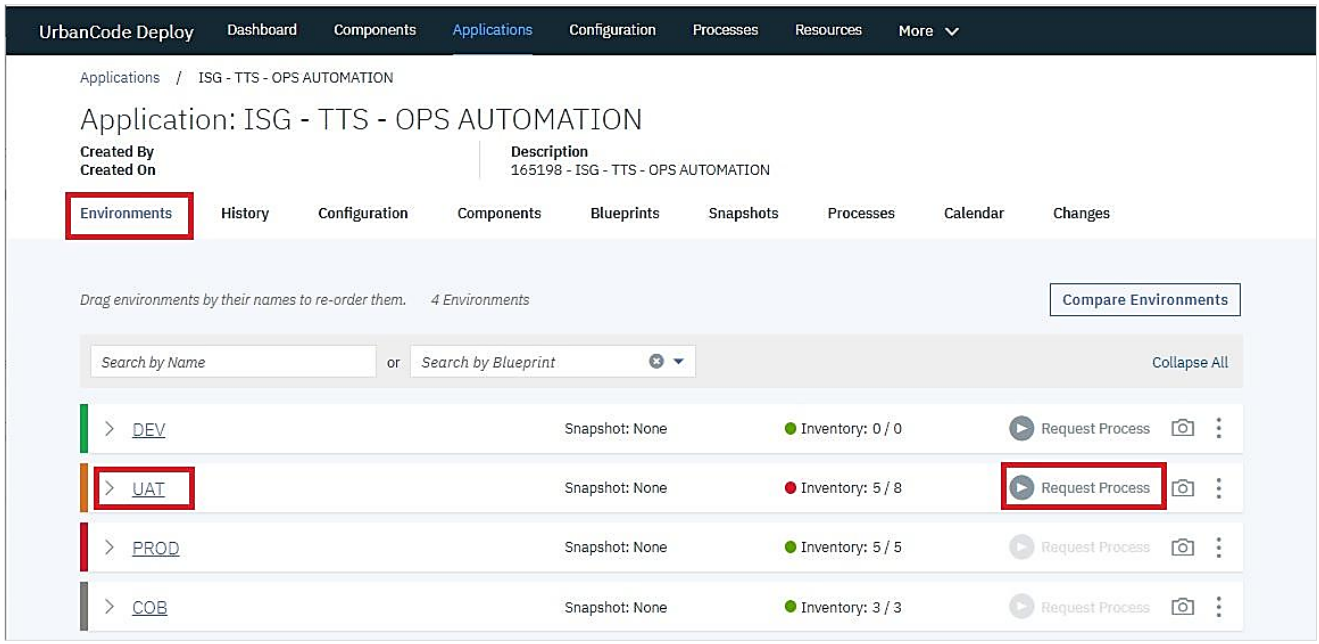
yapılır. Uygulamanın dağıtımının yapılacağı sunucu vb. bileşen bağlantı tanımları Şekil 18'de yer alan uygulama ekranı üzerinden gerçekleştirilir.



Şekil 18. Sunucu ve bileşen ilişkilendirme ekran ara yüzü

4.5. Dağıtım

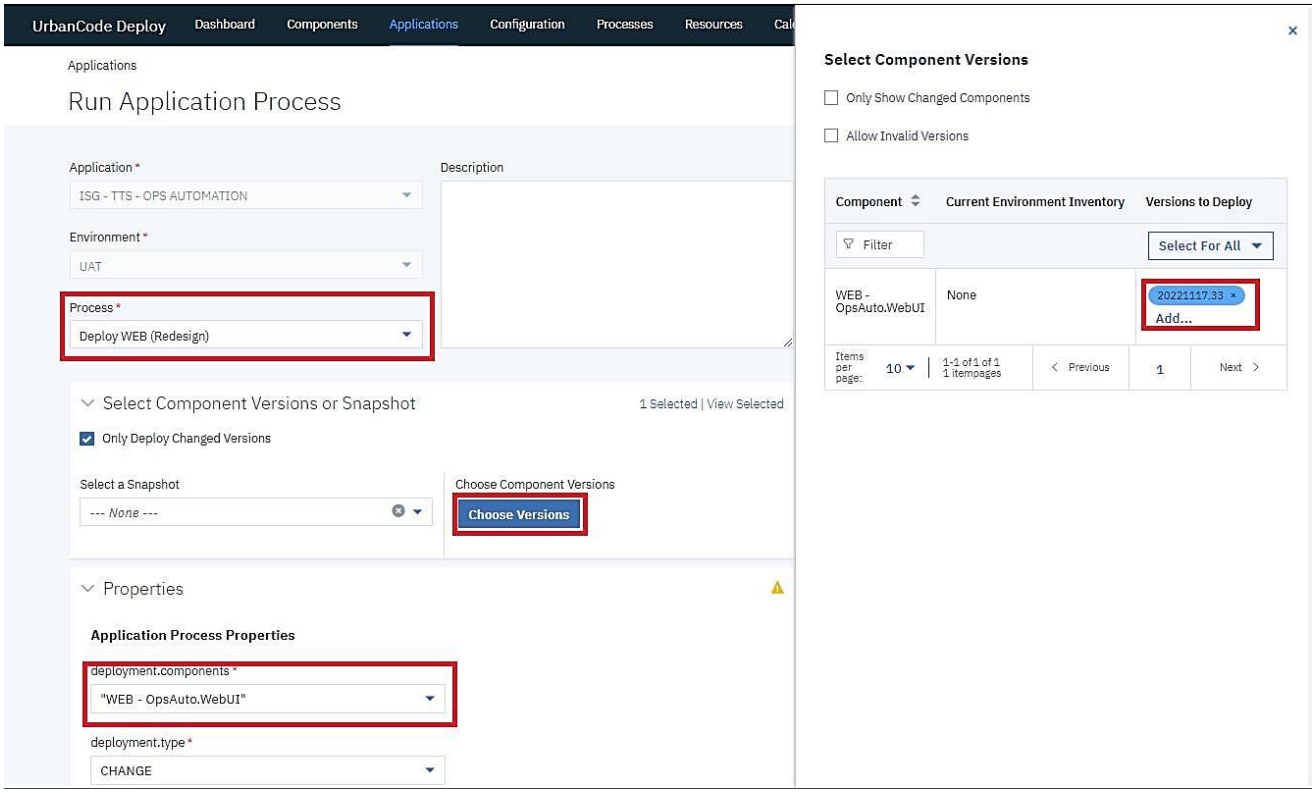
Bu aşama uygulamanın çalıştırılacağı ortamlara güvenli bir şekilde taşıma ve yaygınlaştırma aşamasıdır. Dağıtım yapılacak ortam seçimi 'Application' menüsü üzerinden ilgili uygulama seçildikten sonra '*Application/Environment*' alt menüsü üzerinden dağıtım edilecek ortam satırındaki '*Request Process*' komutu verilir. Dağıtım yapılacak ortamlara ilişkin uygulama ekran görüntüsü Şekil 19'da gösterilmektedir.



Şekil 19. Dağıtım yapılacak ortam seçimi

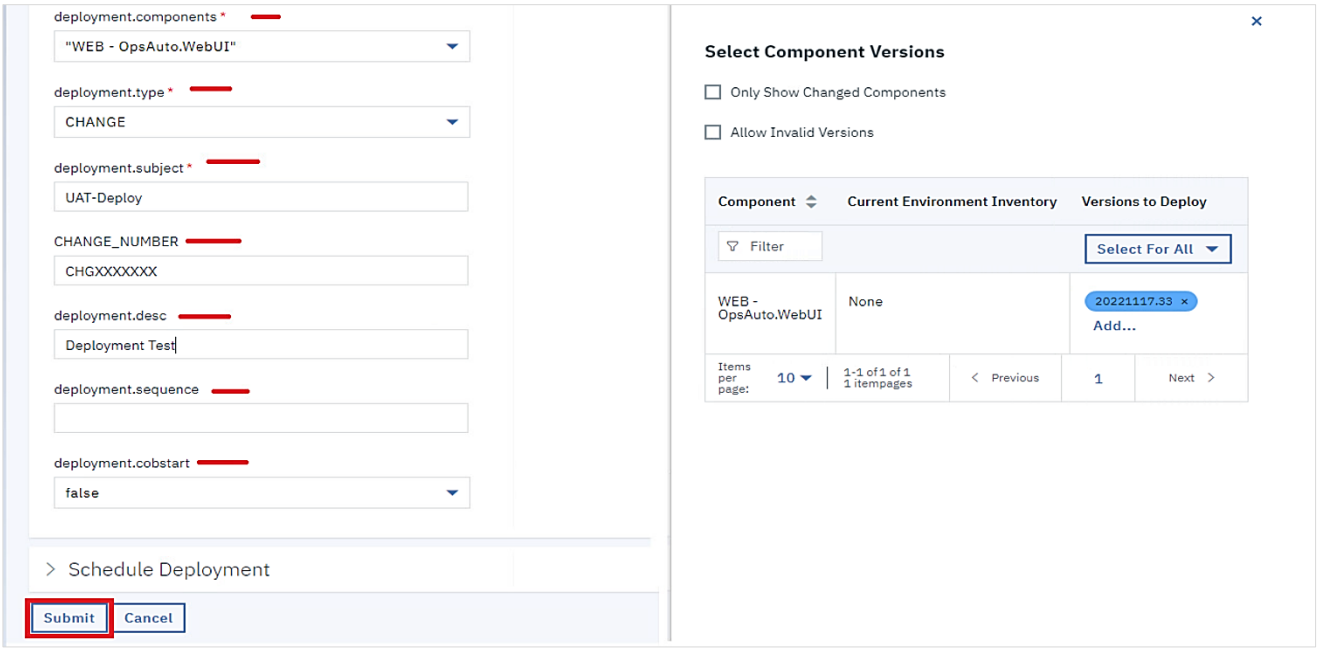
4.6. Çalıştırma

Dağıtım işleminden hemen önce uygulama sürüm bilgisi ve özellik tanımları Şekil 20'de yer alan uygulama ekranı ara yüzü üzerinden gerçekleştirilir. Bileşen sürüm seçimi ('*Choose Versions*') düğmesine tıklandıktan sonra ekranın sağ tarafında yer alan bileşen sürümü üzerinden gerçekleştirilir.



Şekil 20. Dağıtım süreci ön tanımlama

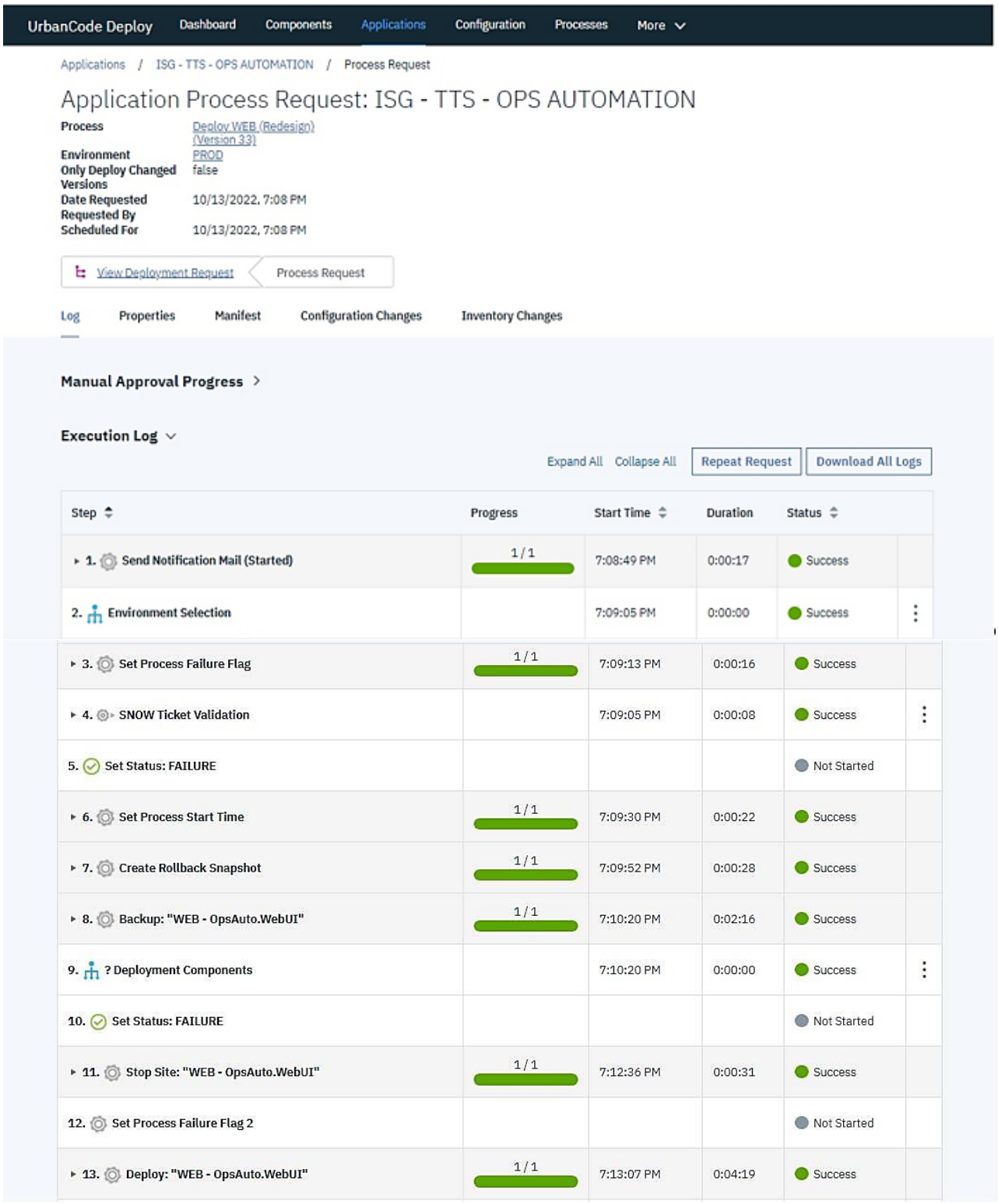
Sürüm seçiminden sonra dağıtım işleminde kullanılacak diğer değişkenlerin değerleri Şekil 21'de yer alan uygulama ekran ara yüzü üzerinden girilir. Ardından dağıtım süreci '*Submit*' komutu verilerek çalıştırılır.

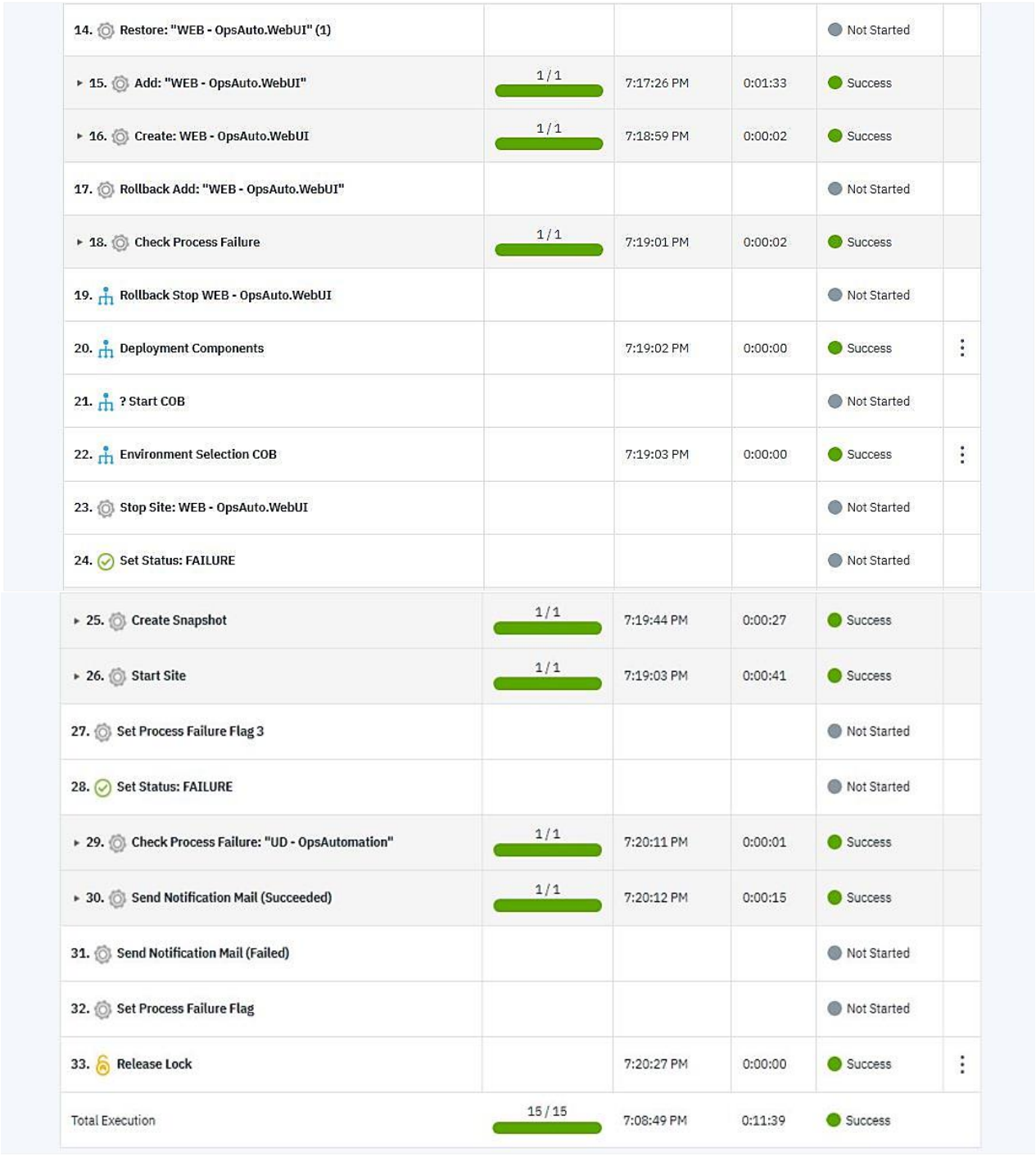


Şekil 21. Dağıtım işlemine ait değişken girişleri

4.7. İzleme

Dağıtım süreci çalıştırılmaya başlatıldıktan sonra çalışan her bir adımda gerçekleşen işlemlere ait iz kaydı(‘*log*’) bilgileri uygulama ekranları aracılığıyla anlık olarak takip edilmektedir. Çalıştırılan her bir adımın ne zaman başladığı ne kadar sürdüğü, durumu gibi bilgilerin yer aldığı izleme ekranına ait görüntü Şekil 22’de yer almaktadır.





Şekil 22. Dağıtım işlemlerine ait izleme ekranı

4.8. Kontrol

Dağıtımı tamamlanan bir uygulamaya yönelik erişim yetkileri, güvenlik gibi kurum ihtiyaç ve beklentilerine dair ek kontroller uygulanır. İzleme ekranından aynı zamanda çalıştırılan adımların yanlarında bulunan ok işaretleriyle o adıma dair çalıştırılan alt süreçler var ise onlara ait detaylar görüntülenebilir. İzleme ekranında karar verme mekanizması işletildiği için her bir adımın çalıştırılması ya da '*Success*' olarak tamamlanması beklenmez. Örneğin '*Send Notification (Failed)'* adımı sadece süreç hata aldığında tetiklenecek şekilde tasarlanmıştır.

5. BULGU VE TARTIŞMA

Manuel dağıtım ele alındığında insan hatası ve dağıtım süresi bakımından iki temel sorunla karşılaşılmaktadır. Manuel dağıtım işlemi her ne kadar işinde bu yetkinliğe sahip kişiler tarafından yapılıyor olsa da insanı etkileyen iş yoğunluğu, stres, duygusal faktörler ve çeşitli çevresel faktörler sebebiyle defalarca tekrarlanan işlerde bile insan hataları kaçınılmaz bir gerçektir. Manuel dağıtım süresi bakımından otomatik dağıtımlara göre göreceli olarak uzundur. Telekomünikasyon, banka, hastane gibi çok farklı sistem ve uygulamaların çevrimiçi bir arada çalıştığı bilişim altyapılarında manuel bir dağıtım ve güncelleme otomatik dağıtıma göre kıyaslanamayacak kadar uzun ve karmaşıktır. Manuel dağıtım, otomatik dağıtımdan çok daha fazla sorunlu ve kesintilere daha yatkındır. Otomatik sistemlerin yapabileceği görevleri tam olarak yapamazlar. Azami özen ve gayret gösterilse bile insan hatalarının olması kaçınılmazdır. Manuel dağıtımda işler yavaş ilerlediği ve hata yapmaya daha açık olduğundan dağıtım işi saatler ve hatta günlerce sürebilir. İlgili ekiplerin dağıtım süresi boyunca görevde kalması gerekir, bu da çalışanların ofis ortamında stres altında daha uzun zaman geçirmesi anlamına gelmektedir. Üstelik bu alanda yetkin bir uzmana sahip olmak da her zaman mümkün olmayabilir. Diğer taraftan bir uygulamanın dağıtımının uzun ve problemli geçme ihtimalinin varlığı uygulama geliştiricilerinde kariyerini kısaltabilecek bir soruna neden olacağı endişesine neden olmaktadır. Ayrıca manuel dağıtım sırasında bir kesinti ya da hata gerçekleştiğinde, daha önceki başarılı olan bir sürüme geri dönmek son derece zor ve riskli bir taşımadır. Bu tür nedenler manuel dağıtımı yetersiz kılmaktadır. Dağıtım sürecinin daha hızlı ve hatasız gerçekleştirilmesinde otomatik dağıtım araçlarının kullanımı bir çözüm sunmaktadır.

Otomatik dağıtım araçlarının kullanılması dağıtım sürecinde yetkin kişinin sürekli olarak bulundurulması ihtiyacını ortadan kaldıracaktır. Otomatik dağıtım süreci, tasarım kısmı hariç, bunların çalıştırılması için yetkin insan kaynağına ihtiyaç duymamaktadır. Böylece yetkin insan kaynağının kurum içerisinde daha verimli olarak değerlendirilmesi, kuruma sağladığı faydanın yanı sıra çalışan uzman kişinin çalışma motivasyonunu da arttıracaktır. Karmaşık gibi görünen dağıtım süreç tasarımları ile bu çalışmada olduğu gibi başta şablonlar oluşturulup sonraki kullanımlar basitleştirebilir. Özellikle sürekli tekrarlanan uygulama dağıtımlarında dağıtım sürecinin baştan bir defa tasarlandıktan sonra istenildiği kadar kullanılması imkânı vardır. Aynı zamanda bu kullanım modeli çok fazla sayıda uygulama barındıran bir kurumda merkezi yönetim ve daha hızlı süreç tasarımı ortaya çıkarma gibi yararlar da sağlamaktadır. Kurum kendi içerisinde şablonlar ve kullanım için eğitim materyalleri hazırlayarak işe yeni başlayan çalışanlar için sunulan oryantasyon sürecini kısaltabilir. Böylece bu iş için gerekli yetişmiş teknik insan kaynağına olan talebi de azaltabilir.

Güvenlik bakımından, bir sunucuya her bir erişim bir risk teşkil etmektedir. Erişim yetkisi olan kullanıcı her ne kadar bu yetkiye kontrollü olarak sahip olsa da bilerek ya da bilmeyerek uygulamaya zarar verebilir, iş sürekliliği kesintiye uğrayabilir. Otomatik dağıtım sürecinde bu yetkinlikte bir erişime ihtiyaç olmadığı için bu risk bütünüyle ortadan kalkmaktadır.

Telekomünikasyon, sigortacılık ve bankacılık gibi birçok sektörde yazılımların dağıtım ve güncellenmesine yönelik yasal düzenlemeler yanında COBIT, ITIL ve CMMI gibi kalite standart gereksinimlerine dair sıkı kurallar söz konusudur. Bağımsız otoriteler tarafından bu kurallara uyuma ilişkin kontrol ve denetimler gerçekleştirilmektedir. Bu kontrol ve denetimlerde sürece ilişkin detaylı veri ve iz kayıtlarına ihtiyaç duyulmaktadır. Bu isteklerin karşılanması otomatik dağıtım aracı kullanılarak gerçekleştirilen dağıtımlarda oldukça kolaydır. Dağıtıma ilişkin iz kayıtlarının sistem üzerinde uçtan uca görülebilir olması denetleyen taraf için de kolaylık ve güvence sağlamaktadır. Manuel gerçekleşen süreçler ise genellikle kimin neyi, ne zaman, nerede ve neden yaptığını açıklayan, merkezi belgelerden ve delillerden yoksundur. Bu süreçte herhangi bir kayıt yapısı bulunmamaktadır. Bu nedenle bazı kurumlarda yapılan aktivitenin kayıt altında tutulması için manuel olarak ekran görüntüleri alınıp saklanması istenmektedir. Bu durumda manuel olarak dağıtımı yapan kişinin bu konu için de çalışması, zaman ayırması gerekir ve süreç daha da uzar. Diğer taraftan bir bankada iç denetim, bağımsız dış denetim, kalite denetimi gibi birçok kontrol ve denetim faaliyeti yürütülmektedir. Sık gerçekleşen denetimler bilgi teknolojileri tarafında ciddi iş yüküne sebep

olmaktadır. Bilgi teknolojileri yönetimleri otomatik dağıtım araçlarını kullanarak denetim için harcanan süreyi kısaltarak elindeki kaynakları daha verimli kullanabilme imkânına sahip olacaktır. Özetle otomatik dağıtım araçlarının kullanımı orta ve büyük ölçekli kurumsal firmalarda bilgi teknolojilerinin kontrol ve yönetimine katkı sunmaktadır.

6. SONUÇ

Bu çalışmada yazılım dağıtım süreci, manuel dağıtımdan kaynaklı sorunlar, bu sorunların kurum ve çalışanlara olan olumsuz etkileri, sorunların çözümünde kullanılan otomatik dağıtım araçları, otomatik dağıtım çözümünün sunduğu katkılar araştırılmıştır. Bu amaçla sorunun çözümünde kullanılan otomatik dağıtım araçları incelenmiş, yazılım dağıtım sürecinin otomatikleştirilmesine yönelik örnek bir WEB uygulaması ele alınarak IBM UrbanCode Deploy ürünü üzerinde uygulamalı olarak çalışılmıştır. Otomatik dağıtımda yer alan kavramlar açıklanmış, dağıtım tasarımının yapılabildiği ekranların kullanımına değinilmiş, hazırlanan şablonlar kullanılmıştır. Uygulamalı çalışmanın tüm safhaları örnek ekran görüntüleriyle ortaya konulmaya çalışılmıştır. Benzer çalışma ve uygulamalar için öğretici olmasına özen gösterilmiştir.

Manuel dağıtım süreci oldukça hataya açıktır. Tanımlanmış bir süreçteki her bir işlem adımının her zaman aynı standartta insan eliyle yapılması çok zordur. Bir dağıtım sürecinin sürekli tekrarı, yazılı olması ve süreci yürüten kişilerin son derece yetkin olması bile manuel olarak yürütülen süreçlerde hata olasılığını ortadan kaldıramaz. Dağıtım sürecinin önemli bir adımı yanlışlıkla atlanabilir, sürüm sırasında meydana gelen bazı hatalar tespit edilemeyebilir, yanlış yazılım sürümleri devreye alınabilir, dağıtım yapılacak ortam ve uygulamalar yeterince analiz edilmeden dağıtıma başlanmış olabilir. Bu ve benzeri birçok nedenden dolayı hatalı, eksik ya da iş kesintisine sebep olan bir dağıtım pek çok sorun ve kayıplara neden olmaktadır. Ayrıca dağıtım işlemini yapan kişinin dağıtımı yapılacak uygulamanın fonksiyonu hakkında yeterli bir bilgi sahibi olması beklenmez. Bu nedenle ciddi etkilere sahip bir uygulamanın dağıtımda bir sorun ile karşılaşma ihtimaline karşı yazılım geliştiricisi, proje yöneticisi, iş analisti ve ilgili diğer teknik personelin bir kriz masası etrafında bir arada çalışmaları gerekir. Uzun süren bir manuel dağıtım sürecinde çok sayıda uzman stres altında çalışır. Dağıtımda herhangi bir hata geliştiğinde iş kesintilerinin uzun sürme ihtimali de yüksektir. Çok fazla yazılım ürünü bulunan ve sürüm güncellemelerinin sıklıkla yapıldığı bir kurumda bu tür hata ve problemlerin yaşanma ihtimali daha da yüksektir. Diğer taraftan sıkı denetim ve kontrollere tabi olan kurumlarda dağıtım sürecinin işletilmesi yanında bunun kontrolü de oldukça zordur. Manuel dağıtım sürecinden kaynaklı bu ve benzeri birçok problemin çözümünde otomatik dağıtım araçlarının kullanımı bir çözüm sunmaktadır.

Günümüzde orta ve büyük ölçekli kurumların bilgi teknolojileri kuruluşlarında uygulama dağıtım süreçlerinin otomasyonu, yasal uyum yükümlülüğünün ötesinde sunduğu birçok imkân ve katkı nedeniyle önemli bir ihtiyaç haline gelmiştir. Bu çalışmada henüz otomatik dağıtım sürecini kullanmayan ve manuel işlemden dolayı birçok sorun yaşayan kurum veya kuruluşlar için yol gösterici öneriler bulunmaktadır. Manuel dağıtım sürecinden kaynaklanan birçok soruna karşı otomatik sürecin sunduğu çözüm ve katkılara dikkat çekilerek belli bir farkındalık oluşturulması hedeflenmiştir. Hali hazırda otomatik dağıtım ürünü kullanan kurumlar için ise dağıtım sürecinde tekrarlanacak standart operasyonları şablon haline getirmenin merkezi yönetim için sağladığı katkılara dikkat çekilmiştir. Tekrar eden yeni uygulamalarda bu yöntem kullanılarak dağıtım süreç tasarımlarının çok daha kısa sürede nasıl yapılabildiği anlatılmıştır. Yazılım yaşam döngüsünde önemli bir yere sahip olan yazılım uygulamalarının dağıtım süreci ve bu sürecin otomatikleştirilmesine yönelik uygulamalı çalışma literatüre kazandırılmıştır. İlgili araştırmacılara ve teknik kullanıcılara bir referans kaynağı olarak sunulmaktadır.

KAYNAKÇA

Ali, M., Aftab, A. & Buttt, W.H. (2020). "Automatic Release Notes Generation". 2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS), pp. 76-81, doi: 10.1109/ICSESS49938.2020.9237671.

Ali, M. Tarar, M.I.N. & Butt, W.H. (2020). "Automatic Release Notes Generation: A Systematic Literature Review," 2020 IEEE 23rd International Multitopic Conference (INMIC), pp. 1-5, doi: 10.1109/INMIC50486.2020.9318191.

Arcangeli, J-P., Boujbel, R., & Leriche, S. (2015). "Automatic Deployment of Distributed Software Systems: Definitions and State of the Art". Journal of Systems and Software. Vol.103, pp. 198-218, https://doi.org/10.1016/j.jss.2015.01.040

AWS Code Deploy (2022). https://aws.amazon.com/tr/codedeploy/ (14.10.2022).

Bamboo (2022). Continuous Delivery, from Code to Deployment, https://www.atlassian.com/software/bamboo (14.10.2022).

Borandag, E. & Yücalar, F. (2020). "Artırılmış Gerçeklik ile Scrum Task Board Uygulaması". Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi, 4 (1) , 1-12. Doi:10.33461/uybisbbd.652366

Capistrano (2022). What is Capistrano?, https://capistranorb.com/documentation/overview/what-is-capistrano/ (14.10.2022).

CircleCI (2022). Product Overview, https://circleci.com/product/, (14.10.2022).

Chen, M., Zhang, S., Deng, H., Chen, B., Xing, C., & Xu, B. (2020). "Automatic deployment and control of network services in NFV environments". Journal of Network and Computer Applications. https://doi.org/10.1016/j.jnca.2020.102677

CloudBees (2022). https://www.cloudbees.com/, (14.10.2022).

Codar (2022). Deploy and redeploy packages, https://docs.microfocus.com/Codar/1.80/Content/Concepts/devopsUseCase-DeployRedeploy.htm (14.10.2022).

DeployBot (2022). DeployBot Code Deployment Guides, https://deploybot.com/ (14.10.2022).

DergiPark (2022). DergiPark Akademik, https://dergipark.org.tr/tr/ (20.03.2023).

DevOps (2022). Azure DevOps Services , https://learn.microsoft.com/tr-tr/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops, (14.10.2022).

Digital (2022). Deploy Description, https://digital.ai/products/deploy/ (18.10.2022).

Eloranta, M. (2018). "Continuous development and release automation of web applications", Master Thesis, School of Electrical Engineering, Aalto University.

Farley, D., & Humble, J. (2010). "Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation", Addison-Wesley Professional, ISBN: 9780321670250

GoCD (2022). GoCD Features, https://www.gocd.org/why-gocd/ (16.11.2022).

Gradle (2022). Gradle Build Tool, https://gradle.org/ (16.11.2022).

Highsmith, J. & Cockburn, A. (2001). "Agile Software Development: The Business of Innovation", in Computer, vol. 34, no. 9, pp. 120-127, IEEE, doi: 10.1109/2.947100.

Hofmann, P., Samp, C., & Urbach, N. (2019). Robotic process automation. Electronic Markets. https://doi.org/10.1007/s12525-019-00365-8.

IBM (2022). What is IBM® UrbanCode® Deploy?, https://www.ibm.com/cloud/urbancode/deploy (25.08.2022).

IBM_Urbancode (2022). Systems and topology overview, https://www.ibm.com/docs/en/urbancode-deploy/7.0.5?topic=deploy-systems-topology-overview#ov_systems__core (12.08.2022).

Jenkins (2022), What is Jenkins?, https://www.jenkins.io/doc/ (16.09.2022).

Lascu, T., Mauro, J., & Zavattaro, G. (2015). "Automatic Deployment of Component-Based Applications". Science of Computer Programming. https://doi.org/10.1016/j.scico.2015.07.006

Mohd Nordin, A.A., Latih, R. and Ali, N. M., "Software Development Productivity Model: Validation through Expert Review". 2021 International Conference on Electrical Engineering and Informatics (ICEEI), Kuala Terengganu, Malaysia, 2021, pp. 1-6, doi: 10.1109/ICEEI52609.2021.9611151

Morris, K. (2016). "Infrastructure as Code: Managing Servers in the Cloud", O'Reilly Media.

Octopus (2022), Deployment automation, https://octopus.com/, (16.09.2022).

Pai, W. (2002). "A Quality-Enhancing Software Function Deployment Model". Information Systems Management, 19:3, 20-24, doi: 10.1201/1078/43201.19.3.20020601/37166.3

PiriKesif (2022). Piri Keşif Aracı, https://kesifaraci.com/, (20.03.2023)

Poppendieck, M., & Poppendieck, T. (2007). "Implementing Lean Software Development: From Concept to Cash", Addison-Wesley Professional, ISBN: 0321437381.

Ruiz-Rube, I., ManuelDodero, J., & Colomo-Palacios, R. (2015). A framework for software process deployment and evaluation. Information and Software Technology. https://doi.org/10.1016/j.infsof.2014.12.001

Şahinaslan, E. (1998). Yazlımda Kalite Modellerinin Değerlendirilmesi, Bilgisayar Mühendisliği YL Tezi, Enstitüsü Mühendislik ve Fen Bilimleri Enstitüsü, Gebze Teknik Üniversitesi, İzmit, Türkiye

TeamCity, (2022). Powerful continuous integration for DevOps-centric teams, https://www.jetbrains.com/teamcity/ (16.10.2022).

Travis (2022). About Us, https://www.travis-ci.com/about-us/ (16.10.2022).

UrbanCode(2022). Accelerate Software Delivery with IBM UrbanCode, https://www.royalcyber.com/technologies/urbancode/ (20.10.2022).

Zhai, H., & Wang, J. (2021). "Automatic deployment system of computer program application based on cloud computing". International Journal of System Assurance Engineering and Management. https://doi.org/10.1007/s13198-021-01068-0

Zhang, C., Sun, E., Cheng, X., Zang, D., & Chen, Z. (2021). A Scheme and Implementation of Automatic Deployment of Multilingual Industrial Mechanism Model Based on OpenStack. *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*. https://doi.org/10.1109/aeeca52519.2021.9574124