T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BLOKZİNCİRİ TEMELLİ BULUT İMALAT MODELİ

DOKTORA TEZİ

Baran KAYNAK

Enstitü Anabilim Dalı	**:**	**ENDÜSTRİ MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Özer UYGUN**

Aralık 2020

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BLOKZİNCİRİ TEMELLİ BULUT İMALAT MODELİ

DOKTORA TEZİ

Baran KAYNAK

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Bu tez 29/12/2020 tarihinde aşağıdaki jüri tarafından oybirliği/oyçokluğu ile kabul edilmiştir.

Jüri Başkanı **Üye** **Üye**

Üye **Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Baran KAYNAK
29.12.2020

TEŞEKKÜR

Doktora eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışmanım Doç. Dr. Özer Uygun hocama, aynı zamanda destekleri için tez izleme komitesi üyeleri Dr. Öğretim Üyesi Gültekin Çağıl ve Prof. Dr. Ahmet Özmen hocalarıma teşekkürlerimi sunarım.

Hayatım boyunca bana her türlü desteği sağlayan aileme, doktora sürecinde tüm zorlukları benimle göğüsleyen ve hayatımın her evresinde bana destek veren sevgili eşim Sümeyye Kaynak'a, varlığıyla beni onurlandıran sevgili oğlum Taha Kaynak'a minnet duygularımı sunarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

ABI	: Application Binary Interface
AHP	: Analitik Hiyerarşi Prosesi
AMRG	: Mevcut imalat kaynakları grubu
AWS	: Özerk çalışma sistemi (Autonomous Work System)
BC	: Blokzinciri destekli imalat
BC-DApp	: Blokzinciri destekli merkezi olamayan uygulamalar
BCmfg	: Blokzinciri
BHT	: Bizans hata toleransı
BT	: Bilgi teknolojileri
C	: Maliyet
CBDM	: Bulut tabanlı tasarım ve imalat
CM	: Bulut imalat
CPS	: Siber fiziksel sistemler
DApp	: Merkezi olmayan uygulamalar
DCMApp	: Dağıtılmış bulut imalat uygulaması
DGA	: Dağıtık genetik algoritma
DPoS	: Temsili hisse kanıtı (Delegated Proof of Stake)
ED	: Erken teslim
HABC	: Hibrit yapay arı kolonisi
IoT	: Nesnelerin interneti (Internet of Things)
IPFS	: Gezegenler arası dosya sistemi (InterPlanetary File System)
JC	: İş sözleşmesi
JSC	: İş akıllı sözleşmesi
KOBİ	: Küçük ve orta büyüklükteki işletmeler
LD	: Geç teslim
LPoS	: Kiralık hisse kanıtı (Leased Proof of Stake)

MRG	: Tüm imalat kaynakları
NFS	: Ağ dosya sistemi (Network File System)
NIST	: Ulusal standartlar ve teknoloji enstitüsü
OWL	: Web ontoloji dili (Web Ontology Language)
PC	: Kişisel bilgisayarlar
PCO	: Parçacık sürü optimizasyonu
P	: Önceden seçilmiş şirketler
P2P	: Noktadan noktaya bağlantı
PoW	: Emek kanıtı (Proof of Work) uzlaşma protokolü
PoS	: Hisse kanıtı (Proof of Stake) uzlaşma protokolü
Q	: Kalite
QoS	: Hizmet kalitesi
R	: Risk
ReST	: Temsili durum transferi (Representational State Transfer)
RFID	: Radyo frekansı ile tanımlama
RSC	: Kaynak akıllı sözleşme
RT	: Tüm kaynak türleri
RU	: Kaynak birimleri
SC	: Akıllı Sözleşmeler
SMTP	: Basit posta aktarım protokolü (Simple Mail Transfer Protocol)
SOA	: Servis yönelimli mimari
SOAP	: Basit nesne erişim protokolü (Simple Object Access Protocol)
T	: Zaman
TCQR	: Zaman, Maliyet, Kalite, Risk
TCQS	: Zaman, Maliyet, Kalite, Hizmet
TQCSEFK	: Zaman, Kalite, Maliyet, Hizmet, Çevre, Esneklik, Bilgi
UDDI	: Evrensel tanımlama keşfi ve entegrasyonu (Universal Description Discovery and Integration)
WSDL	: Web hizmetleri tanımlama dili
XaaS	: Servis olarak X
XML	: Genişletilebilir işaretleme dili (Extensible Markup Language)
2FA	: İki faktörlü kimlik doğrulama

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Bulut imalat, bulut tabanlı tasarım ve modelleme, blokzincir, merkezi olmayan uygulamalar, blokzincir tabanlı imalat, QoS, bulut bilişim

Bu çalışmada, Endüstri 4.0 teknolojilerinden biri olan CBDM (Bulut Tabanlı Tasarım ve Modellem) üzerine araştırmalar yapılmıştır. Literatürde yer alan geliştirilmiş CBDM modelleri incelenmiş ve örnek bir CBDM modeli geliştirilmiştir. CBDM modellerinin merkezi bir yapıya sahip olmasından kaynaklanan problemler incelenmiş ve bu problemlerin imalat platformları için oluşturdukları tehlikeler anlatılmıştır. Bu çalışma kapsamında SC (Akıllı Sözleşme) destekli DApp (Merkezi Olmayan Uygulamalar) ile bahsedilen problemlere çözüm sunulmuştur. Geliştirilen DCMApp (Merkezi Olmayan Bulut İmalat Uygulaması) CBDM platformlarının temel döngülerini barındırmaktadır. BC (Blokzinciri) ağındaki yeni teknolojik gelişmeler, daha uygun maliyetli yeni BC ağlarının gelişimi, SC'ların geliştirilmesi; daha ekonomik DCMApp uygulamalarının geliştirilmesine ve diğer imalat hizmetlerinin sisteme doğrudan entegre edilebilmesine imkân sağlayacaktır.

Geliştirilen bulut imalat modeli genel bir yapıya vurgu yapmaktadır. Geleneksel CBDM platformlarında olduğu gibi tek bir sektöre hizmet veren bir model yerine; sunulan genel yapı ile birden fazla sektörü içerisinde barındıracak modeldir. CBDM modeli temel alınarak geliştirilen DCMApp uygulaması hibrit bir yapıya sahiptir. Bu özelliği DCMApp uygulamasını diğer DApp uygulamalarından farklı kılar. DCMApp uygulamasında halka açık Ethereum BC ağı kullanılmaktadır. Ethereum ağı 2. büyük BC ağıdır ve halka açıktır. Bu özellikleri DCMApp'a güvenirlik, güvenlik, şeffaflık, sağlamıştır. DCMApp uygulaması ile sunucu altyapı ihtiyacı ortadan kalkmıştır ve herhangi bir kişi; herhangi bir aracıya ihtiyaç duymadan, herhangi bir yatırım yapmadan, komisyon ücreti, sürdürülülebilirlik, bakım gibi maliyetleri ödemeden birbirleriyle imalat anlaşmaları yapabilmektedir. DCMApp'da tüm bilgiler işlem ücreti geretiren genel bir Ethereum BC ağında saklanmaz. Ürün tasarımı, kaynak bilgileri, özel mesajlar gibi detaylı bilgiler yerel veritabanlarında, dağıtılmış bir sunucuda veya bulut hizmetlerinde tutulabilir. DCMApp'ın hibrit bir yapıya sahip olması daha ekonomik bir uygulama olmasını sağlamaktadır.

Merkezi olmayan ve dağıtık yapıdaki bir uygulamada kaynak yönetiminin gerçekleştirilmesi oldukça zor bir görevdir. DCMApp uygulamasında kaynak yönetimi için AHP ve genetik algoritmayı kullanan yeni bir hibrit model geliştirilmiştir. Model; TCQS, ED, LD, P gibi parametreleri ve bu parametrelerin birbirleriyle olan ilişkilerini dikkate alır. Müşteri önceliklerine göre kullanıcıya çözüm seçenekleri sunar. Geliştirilen model ile DCMApp için bir karar destek sistemi önerilmiştir.

BLOCKCHAIN BASED CLOUD MANUFACTURING MODEL

SUMMARY

Keywords: Cloud manufacturing, cloud-based design and modelling, blockchain, decentralized applications, blockchain based manufacturing, QoS, cloud computing

In this study, CBDM (Cloud Based Design and Manufacturing), one of the Industry 4.0 technologies were investigated. The developed CBDM models in the literature were examined and a sample CBDM model has been developed. Problems arising from the fact that CBDM models have a central structure are examined and the dangers these problems pose for manufacturing platforms are explained. Within the scope of this study, solutions to the mentioned problems are presented with SC (Smart Contract) supported DApps (Decentralized Applications). The developed DCMApp (Decentralized Cloud Manufacturing Application) contains the basic steps of CBDM platforms. New technological developments in BC (Blockchain) network, development of new more cost-effective BC networks, development of SCs; it will enable more economical DCMApp applications to be developed and other manufacturing services to be directly integrated into the system.

The developed cloud manufacturing model emphasizes a generic structure. As in traditional CBDM platforms, instead of a model serving a single sector; It is a model that will accommodate more than one sector with the generic structure presented. DCMApp developed on the basis of CBDM model has a hybrid structure. This feature makes DCMApp different from other DApp applications. In the DCMApp, public Ethereum BC network is used. Ethereum network is the 2nd largest BC network and it is public. These features provided DCMApp with reliability, security, transparency. With the DCMApp, the need for server infrastructure has been eliminated and any person; It can make manufacturing agreements with each other without needing any intermediary, without making any investment, without paying commission fee, sustainability, and maintenance costs. In DCMApp, all information is not stored on a public Ethereum BC network, which requires transaction fees. Detailed information such as product design, source information, private messages can be stored in local databases, on a distributed server or in cloud services. DCMApp is a more economical application.

A new hybrid model has been developed in DCMApp application that uses AHP and genetic algorithm for resource management. The model considers parameters such as TCQS, ED, LD, P and the relationships of these parameters with each other. It offers solutions to the user according to customer priorities. With the developed model, a decision support system for DCMApp is proposed.

BÖLÜM 1. GİRİŞ

İmalat sistemleri ürün veya hizmetin değerini arttırmak, ürün veya hizmetleri oluşturmak için gerekli olan tüm faaliyetleri içerir. Yoğun rekabet, müşteri ihtiyaç ve beklentilerinin hızla değişmesi, teknolojide meydana gelen hızlı gelişme imalat sistemini değişikliğe itmiştir.

1764 yılında James Watt, buhar makinesini geliştirerek Endüstri 1.0'ın başlamasını sağlayan teknolojik bir devrim gerçekleştirmiştir. Bu teknolojik gelişme ürün sayısında ve kalitesinde artışın olmasını sağlamıştır. Merkezi bir yapıya sahiptir. 1776 yılında Adam Smith uzmanlaşma kuramını ortaya atmıştır. 18. yüzyılda ise Charles Babbage uzmanlaşmaya göre iş bölümü kavramını daha da geliştirmiştir. 19. yüzyılda, çizelgeleme, montaj hattı üretimi, stok kontrol kavramları ortaya çıkmıştır. Üretim sistemlerinde elektrik enerjisinin kullanımı ile endüstrileşme hızlanmış ve Endüstri 2.0 dönemi yaşanmıştır. Bu dönemde bekleme süresi azalmış, ürünlerin kalitesi artmış ve üretimin aşırı olmasından kaynaklanan atıklar azalmıştır. Bilgisayar teknolojileri gelişimi ve üretimde kullanılmaya başlaması ile Endüstri 3.0 dönemi başlamıştır. Bu dönem sanayinin sayısallaşması dönemidir. Teknolojiler birbirleriyle kaynaşmış durumdadır, süreçler iletişim halindedir. Tüm bu tarihsel akışta seri üretim, esnek imalat, bilgisayarla tümleşik imalat, yalın üretim, tam zamanında üretim, eş zamanlı mühendislik, çevik üretim, web tabanlı imalat ve sanal imalat teknolojileri geliştirilmiştir.

Endüstri 3.0 dönemine kadar endüstri sektörleri ve geliştirilen teknolojiler merkezi bir imalat sistemi yapısındaydı [1]. İnternetin gelmesiyle birlikte web tabanlı imalat sistemleri, etmen tabanlı imalat sistemleri gibi dağıtık imalat sistemleri benimsenmeye

başladı. Dağıtık bir yapıya sahip olması ürün çeşitliliğini, rekabeti, kaliteyi arttırdı; maliyeti, süreyi azalttı. Fakat yapısal ve işlevsellik bakımından karmaşıklığın artmasını neden oldu, ölçeklenebilirlik kısıtlıydı. Makineler efektif bir şekilde kullanılamıyordu. Otomasyon tam anlamıyla yapılamıyordu. Makinelerin birbirleriyle iletişimi yoktu. Bir ürün veya hizmetin, tasarımdan başlayarak üretilip satışa sunulmasına kadar geçirdiği aşamalar tam anlamıyla bütünleşmiş değildi. Bilgi arka planda tutuluyordu. Gelişmiş imalat teknolojileri ve modelleri, imalat şirketlerinin TQCSEFK (pazara en kısa zamanda girme-fastest Time to market, yüksek kalite-highest Quality, düşük maliyet-lowest Cost, en iyi hizmet-best Service, temiz çevre-cleanest Environment, geniş esneklik-greatest Flexibility ve büyük bilgi-high Knowledge) hedeflerini karşılamamaktaydı [2]. Kullanıcılara; bazı üretim kaynakları ve yetenekleri hizmet olarak sunulamıyordu. Bu anlamda bu ihtiyaçları karşılayacak yeni teknolojilere ve imalat anlayışına ihtiyaç vardı.

Siber fiziksel sistemler, nesnelerin interneti, bulut bilişim, sanallaştırma teknolojileri, büyük veri analiz araçlarının geliştirilmesi Endüstri 4.0 dönemine ait ihtiyaçların karşılanmasında altın değerinde olan araçlardır. Bu araçlar daha fazla verinin toplanmasına, güvenliğin sağlanmasına, ölçeklenebilirliğin artmasına, verinin bilgiye dönüştürülmesine imkân sağlamaktadır. Bu teknolojiler Endüstri 4.0 döneminin ortaya çıkmasında öncü teknolojilerdir. Endüstri 4.0 terimi ilk olarak Alman Hükümeti tarafından desteklenen 2011 yılında Hannover Fair 2011'de açıklandı.

Bulut tabanlı tasarım ve imalat; işletmelerin Endüstri 4.0 döneminde ortaya çıkan teknolojilere daha kolay uyum sağlayabilmeleri için Endüstri 4.0 çağının bir parçası olarak, bir alt yapı sunmak üzere ortaya çıkmıştır. Bulut tabanlı tasarım ve imalat, günümüzde var olan gelişmiş imalat üretim modellerinden ve Endüstri 4.0 çağının teknolojilerinden yola çıkılarak oluşturulan yeni hizmet odaklı bir üretim modeli olarak tanımlanabilir [3].

Klasik modeller; eski hiyerarşik iş modelleri, kitlesel iş birlikleri çeviklik derecesini, yaratıcılığı, firmaların ayakta kalması için gerekli olan bağlantıları sağlayamazlar [4]. Bulut tabanlı tasarım ve imalat ile işletmeler internetin olduğu her yerde ve her zaman kendi taleplerine bağlı olarak hizmet alabilir, kaynaklarını ve yeteneklerini paylaşabilir; aynı zamanda müşteri olabilir. Böylelikle, işletmeler üretmek istedikleri değerlerin üretilebilmesi için gereken kaynak ve yetenekleri bulut imalat sayesinde farklı işletmelerden sağlayabilir; yatırım engeline takılmadan sadece odaklanması gereken noktalara odaklanarak, daha hızlı bir şekilde pazara girebilir, daha verimli bir şekilde ürün ve servisleri üretebilir.

Bulut tabanlı tasarım ve imalat küreselleşmeyi sağlamıştır. Küreselleşme sayesinde tasarım ve imalat aktiviteleri, küçük ve orta büyüklükteki işletmeler (KOBİ) ve büyük ölçekli şirketlerin bulunduğu merkezi bir yapıda yönetilir ve dağıtık bir yapıda icra edilebilir. İnternetin var olduğu her yerden sürekli bir iletişim mevcuttur. Böylelikle; zamandan ve maliyetten tasarruf sağlanır.

Endüstri ve akademik çevrenin farklı bakış açılarına, geçmiş deneyimlerine ve farklı ihtiyaçlara sulunan çözümlere göre farklı birçok bulut imalat mimarileri/modelleri/çatıları ve uygulamaları ortaya çıkmıştır. Temel olarak bulut tasarım ve imalat modelleri bulut sağlayıcıları, bulut kullanıcıları ve imalat kaynaklarının bulunduğu bir havuzdan oluşur. Bu temel yapının üzerine bulut kullanıcıları ve bulut sağlayıcıları arasında imalat kaynaklarının servis haline getirilmesi için gerekli olan iş süreçleri, farkı katmanlarla ve bu katmanların farklı içerikleri ile tanımlanır.

Bu tez çalışması kapsamında öncelikle bulut imalat sisteminin kavramsal yapısı, kullanım için gerekli altyapı ve teknoloji ile birlikte sağlayacağı avantajlar incelenmiştir. İşletmeler için yol haritası olarak kullanılabilecek bir bulut imalat modeli tasarlanmıştır. Geliştirilen bulut imalat modelinin özellikleri anlatılmış,

geleneksel imalat modellerinden farklılıkları ortaya konulmuş ve bulut imalat modellerinin yetersiz kaldığı noktalar anlatılmıştır.

Yüksek performanslı hesaplama, esneklik, ölçeklenebilirlik, internet ile birbirlerine bağlı bir ortam, kullandığın kadarını öde, büyük veri analiz desteği, her yerden erişim, kaynak havuzu, sanallaştırma, çok kullanıcılı yapı, altyapı hizmeti, bir servis olarak platform, bir servis olarak donanım, bir servis olarak yazılım yetenekleri sayesinde bulut tabanlı tasarım ve imalat paradigması imalat sektörünün birçok problemine çözüm getirmiştir. Fakat merkezi bir yapıya sahip olması ve üçüncü bir aracıya ihtiyaç duyması bazı problemler oluşturmaktadır. Güvenirlik, güvenlik, sürdürülebilirlik, ölçeklenebilirlik, veri kitlenmesi, tek nokta başarısızlığı, veri değiştirilmesi; merkezi bir yapıya sahip olması ve üçüncü bir aracıya ihtiyaç duymasından kaynaklanan temel problemlerden bazılarıdır.

Blokzinciri (Blockchain-BC), merkezi olmayan ve dağıtık bir teknolojidir. BC teknolojisi depoladığı verilerin güvenirliliğini garanti ederek ön plana çıkan bir teknolojidir; bu nedenle üzerinde tutulan veri değiştirilemez, silinemez. BC, değişmez ve dağıtılmış defter olarak düşünülebilir. BC, doğası gereği sahip olduğu özellikler sayesinde sürdürülebilirlik, değişmezlik, süreç tutarlılığı, şeffaflık gibi önemli teknolojik avantajlar sağlar. BC teknolojisinin gelişimi ve eşsiz özellikleriyle, BC teknolojisinden tedarik zinciri, sağlık, oyun, eğitim gibi birçok alanda faydalanılmaktadır [5].

Uygulamalar, amaçlarına göre farklı güvenlik seviyelerine, ağ erişim iznine, hız seviyesine, ağ yapısına ihtiyaç duyabilirler. Bazı uygulamalar tamamen merkezi olmayan bir ağ yapısına ihtiyaç duyarken, bazıları kısmi bir merkezileşmeyi kabul edebilir. Bu ihtiyaçlar BC'nin açık, özel ve konsorsiyum türleri ile karşılanabilir. Halka açık (public) BC ağı tamamen merkezi olmayan, ağ üzerindeki herkesin ağda tutulan tüm verileri görebildiği ve uzlaşma işlemine katılabildiği bir ağ türüdür. Özel BC ağ türünde merkezileşmiş bir yapı vardır. Ağı yöneten şirket/kişiler tarafından

onaylanan düğümler ağa katılabilir, uzlaşma işlemi ise sadece ağın yöneticileri tarafından yapılabilir. Konsorsiyum BC ağ türünde ise uzlaşma işlemine sadece seçilen düğümler katılabilir. Tüm bu türlerin birbirlerine göre üstünlükleri ve zayıf yönleri vardır.

Ethereum, BC ağı üzerinde değer transferini gerçekleştirebileceğimiz public BC uygulamasıdır. Dünya'da BC ağının büyüklüğü bakımından Ethereum BC ağı 2.sırada yer almaktadır [6]. Ethereum ağının büyüklüğü güvenirliliğini arttırır. Ayrıca Ethereum kendi programlama dili Solidity ile programlanabilir akıllı sözleşmeleri destekler. Böylelikle BC ağı üzerinde uygulamalar geliştirilebilir. Ethereum ağı, hem BC ağının hem de akıllı sözleşmelerin avantajlarını içerisinde barındırır. Ayrıca; SC desteğinin olması, merkezi olmayan ve tamamen dağıtık bir mimari yapıya sahip olması BC destekli merkezi olmayan uygulamalarının (DApp) geliştirilmesinde Ethereum BC ağının tercih edilmesini sağlar.

Bulut tabanlı tasarım ve imalat platformlarının merkeziyetçi bir yapıya sahip olması ve güvenilir bir aracıya ihtiyaç duymasından kaynaklanan temel problemlerin BC teknolojisi ile giderilebileceği düşüncesiyle, bu tez çalışması kapsamında bulut tabanlı tasarım ve imalat platformlarının temel döngüsü akıllı sözleşme destekli DApp uygulaması ile gerçekleştirilmiştir. Ethereum BC ağı kullanılarak, kaynak sağlayıcı ve müşteri arasındaki iletişim akıllı sözleşmeler ile sağlanmıştır. Geliştirilen merkezi olmayan bulut imalat uygulaması (DCMApp) hibrit yapıya sahiptir, tamamı açık (public) bir BC ağında işletilmemektedir. Bu özelliği ile diğer BC destekli bulut imalat platformlarından kendini ayırır. DCMApp'ın hibrit yapısı çok daha fazla şeffaf, ekonomik ve güvenli bir imalat anlaşmalarının yapılmasına imkân sağlar. Herhangi bir altyapı kurulum masrafı olmadan düşük maliyetlerle BC ağında anlaşmalar saklanabilir. Ethereum ağından faydalanılması anlaşmaların manipüle edilmesini neredeyse imkânsız hale getirir.

Bulut kullanıcılar ile bulut sağlayıcıları ortak bir havuzda birleştiren DCMApp uygulamasında bulut sağlayıcıları kaynaklarını sisteme tanımlar ve kullanıcılar tüm kaynakların bulunduğu havuzdan ihtiyacı olduğu kaynağı ve ihtiyacı kadarını hizmet olarak alır. Kullanıcıların, ihtiyaç duyduğu kaynağı ortak havuzdan bulması ve ihtiyacı kadarını kullanabilmesi servis arama algoritmaları ile gerçekleştirilmektedir. Gelen istekler analiz edilir, istek analizi ile talep için hangi tip servislerin gerekli olduğu bilgisi elde edilir. İlgili olabilecek servisler arasından en uygun olabilecek servisler belirlenir. Bu işlem birden fazla alternatif çözümü üretir. Alternatif çözümlerin hesaplanması kullanıcıya farklı önem derecelerine göre seçim yapmaya imkân sağlar.

Servis seçimi ve planlama işlemleri bulut tabanlı tasarım ve uygulamalarında temel problemlerdir. Bulut imalat ve tasarımda, müşteri gereksinimlerini karşılayacak şekilde servis seçimi ve planlaması yapılır. Maliyet, ürünün teslim zamanı gibi hedeflerin yanında kalite, risk ve öncelik gibi birçok optimizasyon hedefi dikkate alınarak servis seçimi ve planlaması yapılmalıdır. Bulut imalatın dinamik ve esnek bir yapıya sahip olması bulut imalat ortamında servis seçimi ve planlama işlemlerinin geniş ölçekli ve dinamik bir şekilde yapılmasını gerekli hale getirir.

Klasik imalat sistemlerinde yapılan servis seçimi ve üretim planlama işlemleri genel olarak bulut imalat sistemleri için pratik değildir. Çünkü bulanık ve nicel olmayan kriterler dikkate alınmaz [7]. Son zamanlarda geliştirilmiş imalat sistemlerindeki yeni optimizasyon kriterlerini araştıran birçok çalışma yapılmıştır. Fakat bu çalışmalarda kriterler (Hizmet kalitesi-Quality of Service(QoS))-[8], [9], [10] parça parça veya ayrı ayrı enerji [11], güvenilirlik [12], dürüstlük [13] olarak ele alınmıştır. Bu durum, kriterlerin birbirlerine göre ilişkileri ve farklı senaryolardaki varyasyonları dikkate alınmadan bir gerçekçi olmayan optimizasyon işleminin yapılmasına neden olur [7]. Bu problemlerin üstesinden gelebilmek ve bu eksiklikleri giderebilmek için planlama ve servis seçimini gerçekleştiren genel bir optimizasyon modelinin oluşturulmasına ihtiyaç vardır.

Bu tez çalışması kapsamında, literatürde yer alan bu eksikliklerden yola çıkılarak geliştirilen DCMApp üzerinde servis seçme ve planlama işlemleri üzerine bir optimizasyon modeli geliştirilmiştir. Geliştirilen optimizasyon modeli, geleneksel bulut imalat modeli servis seçme ve planlama optimizasyon modellerinden daha farklıdır. DCMApp dağıtık olan ve merkezi olmayan bir yapıya sahiptir. Bu nedenle, geliştirilen optimizasyon probleminin Ethereum ağı üzerinde yapılan anlaşmalardan elde edilen veriler ile yapılması gerekmektedir. Akıllı sözleşmeler ile yapılan birçok anlaşmanın arasından en uygun kaynağın farklı parametreler ile kıyaslanması ve seçilmesi gerekmektedir. Geliştirilen karar destek yazılımı ile kaynak-iş eşleştirilmesi merkezi olmayan bir yapıda müşteri önceliklerini dikkate alacak şekilde yapılabilmektedir.

Servis seçme işleminde filtreleme işlemi kullanılarak arama yapılacak servis miktarı azaltılarak çözüm uzayı daraltılmıştır. Elde edilen sonuçlar sadece koşulları sağlayan sonuçlardır. Talep edilenden fazla özelliğe sahip olan bir makine kullanılması, kapasitenin âtıl durumda kalmasına neden olacaktır. Tam olarak istenilen özelliklere sahip bir servisin seçilmesi öncelikli olmalıdır. Geliştirilen model zaman (T), maliyet (C), kalite (Q), risk (R) parametrelerini (parametreler çeşitlendirilebilir) ele almaktadır. Bu parametreler, farklı işletmeler için farklı önceliklere sahip olabilir. Geliştirilen model ile işletmenin o anda ihtiyaç duyduğu parametreye daha fazla ağırlık verilerek işletmenin önceliklerine en uygun olan çözüm sunulur.

Binlerce kaynağın bulunduğu bir kaynak havuzu ile binlerce iş ve alt iş parçalarının müşteriye göre eşleştirilmesi matematiksel olarak çözülemeyecek kadar büyük bir problemdir. Bu nedenle alt işlere sahip olan bir işin kaynak planlamasının yapılabilmesi için bir sezgisel algoritmaya ihtiyaç vardır. Bu çalışma kapsamında çok parçalı bir işin planlamasında AHP ve genetik algoritmadan faydalanılmıştır. Sezgisel algoritmalardan genetik algoritma, çözüm uzayında sezgisel olarak tarama yaparak daha hızlı çözüme ulaşmamızı sağlamaktadır. Her bir alt iş parçası için seçilen kaynağın bu iş için uygunluk derecesi AHP (Analitik Hiyerarşi Prosesi) ile hesaplanır. AHP ve genetik algoritmadan faydalanılarak müşteriye özgü kriterler temel alınarak

bir planlama yapılabilmekte ve müşteriye alternatif çözümler sunulabilmektedir. Sunulan bu model farklı kriterlerin birbirlerine göre ilişkilerini dikkate alacak şekilde birçok farklı kriterler için uygulanabilir. Burada amaç DCMApp içinde ihtiyaç duyulan optimizasyon ihtiyacını gidermektir. Bu tez kapsamında optimizasyon problemini çözebilmek için örnek olarak genetik algoritma ile birlikte AHP seçilmiştir. Bunun dışında birçok optimizasyon veya sezgisel yöntemlerden faydalanılabilir.

Literatürde görülen eksiklikleri aşağıdaki gibi listelenmiştir;

- BC tabanlı bulut imalat çalışmaları özel ağlar ağırlıklıdır.
- Halka açık BC ağları ile bulut imalat alanında eksiklik vardır.
- Halka açık bir bulut imalat uygulamasında verinin nasıl saklanacağı hakkında eksiklik vardır.
- Halka açık bir bulut imalat yapısında akıllı sözleşmelerin nasıl kullanılacağı ve kaynak – iş tanımlarının nasıl yapılacağı hakkında eksiklik görülmüştür.

Bu tez çalışmasının literatüre sunduğu katma değerleri aşağıdaki gibi listeleyebiliriz.

- Endüstri 4.0 dönemine ait geliştirilen bulut tabanlı tasarım ve imalat modelleri incelenmiştir. Bu tez çalışması kapsamında örnek bir bulut tabanlı tasarım ve imalat modeli geliştirilmiştir. Geliştirilen bulut tabanlı tasarım ve imalat modeli genel yapıya vurgu yapmaktadır.
- Bulut tabanlı tasarım ve imalat modelinin kısıtları ve imalat ortamında oluşabilecek problemler incelenmiştir.
- Bulut imalatta merkezileşmeden kaynaklanan problemler ortaya konulmuştur. Bu problemlere çözüm olarak blokzinciri (BC) destekli merkezi olmayan

DCMApp adında bulut imalat uygulaması geliştirilmiştir. DCMApp'ın hibrit bir yapıya sahip olması DCMApp'ı özgün kılmaktadır.

- Geliştirilen merkezi olmayan bulut imalat uygulamasına bir kaynak seçim algoritması bileşeni entegre edilerek DCMApp'in nasıl geliştirilebileceği ifade edilmiştir. Geliştirilen kaynak seçim algoritması hibrit bir yapıya sahiptir. QoS parametrelerini dikkate almaktadır.

Geleneksel bulut imalat platformlarının merkezi bir yapıya sahip olması ve 3. bir aracıya ihtiyaç duymasından kaynaklanan problemlerin BC teknolojisi ile giderilebileceği düşüncesiyle, bulut tabanlı tasarım ve imalat platformlarının temel döngüsü akıllı sözleşme destekli DApp uygulaması ile gerçekleştirilmiştir. Literatürde gerçekleştirilen diğer DApp uygulamalarından farklı olarak DApp uygulaması hibrit bir yapıya sahiptir. Bu hibrit yapı, özel BC ağlarının sebep olduğu kısmi merkezileşmeden kurtulmamızı ve tamamen halka açık olmaması da maliyet açısından daha tasarruflu ve veri güvenliği açısından daha güvenilir hale gelmesini sağlamıştır. Aynı zamanda halka açık BC ağının avantajlarından faydalanması DApp'a güvenirlik ve güvenlik sağlamıştır.

Geliştirilen DApp uygulaması üzerine bulanık ve nicel olmayan kriterleri (TCQR (zaman, maliyet, kalite, risk), ED (erken teslim), LD (geç teslim), şirket önceliği) ve bu kriterlerin birbirlerine göre ilişkileri (korelasyon) ve farklı senaryo varyasyonlarını dikkate alacak şekilde bir kaynak planlama ve servis seçimini gerçekleştiren genel bir optimizasyon modeli geliştirilmiştir. Geliştirilen model, müşterilerin önceliklerini dikkate alır. Literatürde yapılan çalışmalardan farklı olarak kriterleri sadece tek tek değil birlikte de değerlendirmesi ve geliştirilen modelin dağıtık olan ve merkezi olmayan bir yapı üzerinde otomatik süreçler halinde uygulanmasıdır.

Bu tez çalışması 5 bölümden oluşmaktadır. Bölüm 1'de literatür incelemesi sonucunda görülen yetersizlikler, bu yetersizler için sunulan çözümler ve geliştirilen uygulamaların literatüre sağladığı katma değerler kısaca bahsedilmektedir. Bu bölümde tezin amacı açıkça belirtilmektedir. Bölüm 2'de Endüstri 4.0 ve teknolojileri tanıtılmıştır. Endüstri 4.0 teknolojilerinden biri olan bulut tabanlı tasarım ve imalat (CBDM) modeli incelenmiştir. Literatürde geliştirilen farklı CBDM modelleri incelenmiş ve CBDM'nin yetersiz kaldığı noktalardan bahsedilmiştir. CBDM platformlarında kaynak seçimi ve planlama modellerinin literatür incelemeleri gerçekleştirilmiştir. Bölüm 3'de CBDM platformlarının yetersizliklerine bir çözüm olarak sunulan DApp uygulamaları tanıtılmıştır. BC destekli DApp uygulamaları ile geleneksel DApp uygulamalarının farkı belirginleştirilmiştir. Uygulamanın daha iyi anlaşılabilmesi adına BC destekli DApp uygulamalarının kullandığı teknolojiler ayrıntılanmıştır. Bulut imalat ortamında BC destekli DApp kullanımının sağladığı avantajlar belirtilmiştir. Bölüm 4'te tez kapsamında geliştirilen uygulama anlatılmaktadır. Literatürde geliştirilen CBDM modellerinin incelenmesi sonucunda yeni bir CBDM modeli geliştirilmiştir. CBDM modeli temel alınarak bulut imalat platformu için bir DApp yazılımı (DCMApp) geliştirilmiştir. DCMApp üzerinde geliştirilen servis planlama modeli de bu bölüm altında ayrıntılı olarak bahsedilmektedir. Tezin amacı, tezin literatüre kattığı değerler, geliştirilen DCMApp uygulamasının değerlendirme özeti ve DCMApp uygulaması üzerinde geliştirilen servis planlama modelinin literatüre sunduğu katma değerler Bölüm 5'te özetlenmektedir.

BÖLÜM 2. BULUT TABANLI TASARIM VE İMALAT

Bulut tabanlı imalatın tanımı, temel özellikleri, bulut imalat modelleri, programlama modelleri, dosya sistemleri, operasyon süreçleri, bilgi ve iletişim modelleri, kısıtları, eksiklikleri gibi birçok açıdan bulut imalat konusu üzerine araştırmalar yapılmaktadır.

Bu bölümde bulut tabanlı tasarım ve imalat (CBDM) kavramına ait tanımlamalar, CBDM kavramının faydalandığı Endüstri 4.0 çağına ait araç ve teknolojiler, geliştirilen CBDM modelleri, CBDM yaklaşımının kısıtları, doğası gereği oluşan problemler ve bu problemlere sunulan çözümler anlatılmaktadır.

2.1. Tanım

CBDM için literatürde yapılan tanımlamalar CBDM'nin daha iyi anlaşılması açısından yardımcı olabilir. Endüstri ve akademik çevrenin farklı bakış açılarına, geçmiş deneyimlerine ve farklı ihtiyaçlara sunulan çözümlere göre bulut imalatın birçok tanımı yapılmıştır.

Literatürde yapılan çalışmalarda, bulut tabanlı tasarım ve imalat kavramının farklı noktalarına değinen tanımlamalardan bazıları aşağıda verilmektedir.

- Bulut imalat, kullanıcıların öngörülemeyen ihtiyaçları ya da ani artan iş yükleri durumlarında esnek ve adaptif bir form içinde bir dizi standardize edilmiş servislere, iş ihtiyaçlarını karşılayabilecek altyapı ve teknolojiye kullanıcıların erişebilmesini sağlayan yeni bir modeldir [14].

- Bulut imalat, talep üzerine imalat hizmeti sağlayan bütünleşmiş siber fiziksel sistemdir [15].
- Bulut imalat; ağa bağlı imalat ve servis yönelimli imalat, zeki imalat anlayışını uygulamak için gerekli olan teknik araçlardan ve bulut bilişim, büyük veri, nesnelerin interneti gibi Endüstri 4.0 araçlarından faydalanan gelişmiş bir imalat modelidir [16].
- Bulut imalat, hizmet tedarikçileri ve müşteriler arasında minimum maliyetle hızlı ürün geliştirme ve innovasyonlara imkân veren ürün gerçekleştirme modelidir. Ayrıca; tasarım ve imalat kaynaklarını birbirine bağlayan bir dizi fiziksel ve sanallaştırılmış hizmet havuzundan oluşan paralel ve dağıtık sistemlerin bir türüdür [17].
- Bulut imalat, üretim kaynaklarını ve yeteneklerini üretim bulutu aracılığıyla erişilebilen isteğe bağlı hizmetlere sanallaştıran bir servis yönelimli imalat modelidir [18]. Bulut imalat; ağ geliştiricisi, üretici, tüketici ağı üzerinden etkili iş birliği, veri madenciliği ve iletişim yoluyla ürün ve süreç yaşam döngüsü boyunca sürdürülebilir çözümler sağlayabilen çok kiracılı, akıllı, bilgi temelli bir imalat platformudur [18].

Yapılan tanımlamalar bakış açısına ve ihtiyaca göre bulut imalatın farklı noktalarına vurgu yapsa da bir bulut imalat kavramı; kaynak paylaşımı, talebe bağlı hizmet, geniş ağ erişimi, her yerden erişim, kaynak havuzu, kullandığın kadarını öde, hızlı esneklik ve ölçülebilir hizmet, bir servis olarak herşey özelliklerine sahiptir. Bulut imalatın henüz kabul edilmiş ortak bir tanımı yoktur [19]. Tanımlara bakıldığında bulut imalatın Endüstri 4.0 çağına ait teknolojilerden faydalandığı ve ileri imalat anlayışlarını desteklediği görülmektedir.

2.2. Endüstri 4.0 Araçları ve Teknolojileri

Akıllı fabrikalar, taşımacılık, üretim ve depolama tesisleri, makineler, üretim araç ve gereçleri; birbirleriyle gerçek zamanlı etkileşim kurabilen, bilgi alışverişinde bulunabilen cihazlar; daha şeffaf, güvenilir, güvenli bir tasarım, imalat, pazarlama ve

sevkiyat süreci Endüstri 4.0 ile 4. Sanayi devrimi ile gelen değişim ve gelişimlerdir. Bulut bilişim, sanallaştırma teknolojileri, siber fiziksel sistemler, nesnelerin interneti, servis yönelimli mimari (SOA), yapay zekâ, büyük veri endüstri 4.0 araçlarından bazılarıdır. Bulut tabanlı tasarım ve imalat modelinin (CBDM) aktif olarak faydalandığı teknolojiler bu bölüm altında anlatılmaktadır.

2.2.1. Bulut bilişim

1980'li yıllarda kisişel bilgisayar üretilmeye başlandığında insanlara kendi bilgi işlem ortamlarını kontrol etme, ihtiyaçlarına uygun yazılım seçme ve sistemleri kendi zevklerine göre özelleştirme özgürlüğünü yani "özgürleştirme"'yi vadetmişlerdi [20]. Fakat kisişel bilgisayarlar (PC) için büyük bir eksiklik vardı. Kişisel bilgisayarların iş birliği içerisinde çalışabilmesi ve paylaşım yapabilmesi için "sneakernet" birinci araçtı. Elektronik bilgiler; manyetik bant, disket, optik disk, USB flash sürücü veya harici sabit sürücü gibi fiziksel cihazlarla paylaşılıyordu.

1980'li yılların ortalarında sunucu-istemci modeli ile kullanıcılar yerel olarak programlarını yürütebilirken, veri paylaşımını bir ağ yardımıyla uzakta bulunan veri merkezleri ile yapabilir hale geldi. İnternet teknolojilerinin kullanılması ile beraber iş birliği içinde olan işletmeler birbirleri ile daha kolay iletişime geçebilir hale geldi. Ağların artması ile iş birliği içinde bulunulan işletmelerin sayısı artmış oldu ve daha fazla kaynak, yetenek, tesis ve diğer temel imkânlar paylaşılabilir hale geldi. Bu gelişim süreci dünya çapında bir bilgi paylaşımına olanak sağladı. Bilgiye ulaşmak daha kolay hale geldi, mesafelerin önemi ortadan kalktı. Daha fazla veriye sahip olmak, bilgi işlem kaynaklarına sahip olmak üzerine bir yarış vardı. Kullanıcılar tüm bilgi işlem süreçlerini kendileri yönetiyor, donanımsal-yazılımsal altyapıyı kuruyordu. Bu süreçleri üçüncü bir aracıya teslim etmek düşüncesinden oldukça uzak anlayış ve işleyiş mevcuttu.

Günümüzün koşullarını yakalamak isteyen kurumlar/kişiler bilgi ve süreç kontrolünü yapabilmek için yazılımlar kurmak, donanımsal altyapıyı hazır etmek ve yenilikleri takip etmek zorundadır. Veri boyutunun büyümesi, bilgi işleme araçlarının ve

donanımsal teknolojilerin hızla gelişmesi, yazılımların güncellenmesi bilgi işlem kontrolünün sağlanmasını zorlaştırdı, maliyetini arttırdı. Kurumlar/kişiler yapması gereken şeylere odaklanmak yerine zamandan ve maliyetten ödün vererek tüm bu işlerle uğraşmak zorunda kaldılar. Nihai olarak; bilgi işlem kaynaklarına sahip olmak için yarışan kurumlar veri ve bilgi işleme kontrolünü üçüncü taraf hizmet sağlayıcılarına teslim etmeye hazır oldular. İnternet ile iletişim kuracağımız bir dış kaynak bilişim hizmeti ile bu endişelerin ortadan kaldırılabileceği fikri yavaş yavaş benimsenmeye başladı [21]. Bu benimseme bulut bilişimin temellerini oluşturmuştur.

Bulut bilişim kavramı Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından endüstri ve akademik çevre tarafından yapılan tanımlamaların anahtar kelimelerini içerecek şekilde aşağıdaki gibi tanımlamıştır [22]:

"Bulut bilişim, en az yönetim çabası veya hizmet sağlayıcı etkileşimi ile hızlı bir şekilde sağlanabilen ve bırakılabilen yapılandırılabilir bilgi işlem kaynaklarının (ağ, sunucu, depolama birimleri, uygulamalar ve hizmetler) paylaşılan bir havuzuna her yerden pratik bir şekilde isteğe bağlı self-servis ağ erişimi sağlayan bir modeldir [22]."

Bulut bilişim teknolojisi isteğe bağlı self servis, her yerden erişim, çoklu kiracılık, sanallaştırma, esneklik, kontrol edilebilir ve ölçülebilir hizmet, kullanım başına ücretlendirme, servis tabanlı mimari yapısı özellikleri ile ön plana çıkmıştır [23]. Günümüzde oldukça yaygınlaşmış bir teknolojidir. Her türden, boyuttan ve sektörden kurum veri yedekleme, e-posta, sanal masaüstü, olağanüstü durum kurtarma, yazılım geliştirme, test etme, büyük veri analizi, müşterilere dönük web uygulamaları gibi çok çeşitli alanlarda bulut bilişim teknolojisini kullanmaktadır [24].

Bulut servisler, kullanıcıların en az çaba ile kaynaklara erişebilmesini sağlamaktadır. Bu servisler aynı anda birçok kullanıcıya hitap edebilen, ölçeklenebilir ve erişimi kolay servislerdir. Servislerin kullanıma hazır olması, ihtiyaç doğrultusunda kiralanabilmesi, herhangi bir teknolojik altyapı gerektirmemesi, bakım maliyetlerinin olmaması kullanıcıların bu servisleri kullanmaya iten önemli etkenlerdir. Bu servisler altyapı, platform ve yazılım olarak sınıflandırılabilir. Altyapının servis olarak

sunulduğu yapıda kullanıcıların fiziksel kaynakları barındırmadığı ve dışarıdan servis olarak satın aldığı kabul edilir. Platformun servis olarak sunulması durumunda fiziksel kaynaklarla birlikte ara yazılımlar da beraberinde sunulmaktadır. Yazılımın servis olarak sunulduğu modelde ise kullanıcılar sadece yazılıma erişirler ve fiziksel kaynaklar üzerinde işleyen hiçbir mekanizma ile alakalı müdahalede bulunmazlar. Bu sınıflandırmalar içerisinde yazılımın servis olarak sunulması modeli servis sağlayıcısına en fazla sorumluluğun yüklendiği modeldir.

Bulut bilişimin sağladığı bu avantajların yanında güvenlik en önemli problem olarak görülmektedir. Bu sorunu aşabilmek adına özel bulut sistemleri geliştirilmiştir. Bu yapıda işletmeler belirli bir kullanıcı kitlesi için özel olarak bir bulut sistemi sağlarlar. Bu gereksinim ile beraber bulut sistemler özel, genel ve hibrit olmak üzere farklı biçimlerde oluşturulabilirler.

Bulut bilişim teknolojisi bilgisayar sistemlerinde bir devri kapatıp yeni bir devir açmıştır. Bulut bilişim ile yaşanan değişiklik sıradan kullanıcıdan yazılım geliştiriciye, bilgi teknolojileri (BT) yöneticisine hatta donanım üreticisine kadar tüm bilişim ekosistemini etkilemektedir.

2.2.2. Sanallaştırma teknolojisi

Sanallaştırma, bulut üretimde kaynak paylaşımı ve dinamik kaynak ayrıştırması için oldukça önemli bir adımdır. Bulut imalat yaklaşımında, fiziksel olmayan kaynaklar (yazılım kaynakları, veri depolama, hesaplama yeteneği) ve fiziksel yetenekler (makineler, monitörler gibi) bulut ile bütünleştirilmelidir [25]. Bulut ile bütünleşme ancak sanallaştırma ile sağlanabilir.

Bulut imalat sanallaştırması, bulut bilişim sanallaştırmasından daha zor bir görevdir. Bilişim kaynaklarının sanallaştırılması için sanallaştırma yazılımları yeterlidir; fakat bulut bilişim ortamları için yeterli değildir. Bir imalat ortamı herhangi bir şekilde veri alınamayan birçok fiziksel makineden ve imalat yeteneklerinden oluşur. İmalat kaynakları (malzeme, insan gibi) karmaşık özelliklere sahiptir. Üretim kaynaklarını ve

yeteneklerini temsil edecek şekilde bilgi modeli oluşturmak zordur. Bulut imalat ortamı dinamiktir. Web hizmetleri ve semantik web ile ilişkili şuanda bulunan standartlar (XML/SOAP/WSDL/UDDI/OWL) üretim kaynaklarına doğrudan uygulanamaz [26]. Çünkü üretim hizmetlerinin anlamı ve kullanımı geleneksel web servislerinden farklıdır. Üretim alanı hakkında birçok kritik bilgi mevcut hizmet tanımlama yöntemleriyle temsil edilemez.

Sanallaştırma sayesinde imalat yetenekleri ve kaynakları sanal bir hizmet olarak kullanıcılara çoklu erişim ile sunulabilmektedir. Daha etkin bir şekilde kaynakların ve yeteneklerin kullanımı sağlanır. Bulut ortamına ait yetenekler, farklı kullanıcılardan gelen talepler doğrultusunda düzenlenir ve tek bir fiziksel imalat kaynağı, ihtiyaçlar doğrultusunda istenilen sayıda mantıksal parçaya bölünerek aynı anda birden çok kullanıcının hizmetine sunulur. Bulut kullanıcıları, kaynaklara yatırım yapmadan sadece kaynağa ihtiyaç duyduklarında hizmet alabilir. Kısa süreli yoğunluğun yaşanması durumunda bulut kullanıcısı buluttan kaynak miktarını arttırabilir, kısa süre için yüksek maliyetlere katlanmak zorunda kalmaz. Sanallaştırma, iş gücü ve maliyeti azaltmanın yanında daha verimli, esnek, kullanılabilir bir sanal imalat ortamının oluşmasını sağlar.

2.2.3. Nesnelerin interneti

Bugün, iletişimin büyük çoğunluğu bilgisayar, telefon gibi insanların doğrudan kullandığı cihazlar üzerinedir ve ana iletişim şekli insan-insan'dır. Teknolojinin gelişmesiyle, ihtiyaçların farklılaşmasıyla bu iletişim şekli insan-insan formundan insan-insan, insan-nesne, nesne-nesne formuna dönüşmektedir [27]. Bu dönüşüm şekli iletişimin her zaman, her yerde ve her şeyde olmasını sağlayacak, bilgi ve iletişim teknolojileri dünyasına yeni bir boyut getirecek, insanların hayatını son derece değiştirecektir.

Nesnelerin interneti (IoT), geleceğin internetinin bir parçası olarak görülmektedir. İletişim formunu insan-insan formundan nesne-nesne formuna dönüşmesini sağlar. Nesnelerin interneti iletişim içerisinde olan aktif bir role sahip milyarlarca akıllı

nesneden oluşur. IoT ile nesnelerin insanlarla, insanların insanlarla konuşmasına ihtiyaç duyulmadan nesnelerin bilgiye erişebilmesi, birbirleri ile iletişim kurabilmesi ve süreçlerin yürütebilmesi mümkün kılınır.

'Nesnelerin interneti' (IoT) kavramını Kevin Ashton ilk olarak 1999 yılında önerdi ve IoT'yi radyo frekansı ile tanımlama (RFID) teknolojisine sahip benzersiz bir şekilde tanımlanabilir, birbirlerine bağlı nesneler olarak tanımlamıştır [28]. IoT kavramının temelini birçok alanda kullanımı yaygınlaşan RFID teknolojisi oluşturur. Kablosuz sensör teknolojilerinin gelişmesi IoT kavramının genişlemesini sağlamıştır. Cihazların algılama, bilgi yakalama, işleme, yorumlama gibi yetenekleri bilgi ve iletişim teknolojilerin gelişmesiyle artmıştır. Böylece; IoT kavramı ortamdan sadece bilgi almak kavramından, zeki ortam ve otonom kontrol kavramlarına genişlemiştir. Kablosuz sensör ağları, barkodlar, RFID, NFC (Ağ dosya sistemi), düşük enerjili kablosuz iletişimler, bulut bilişim gibi birçok teknoloji IoT içerisinde geliştirilmiştir. Bu teknolojilerle beraber IoT'nin kavramı genişleyip değişse de IoT içerisindeki nesnelerin sanal ortamda tanımlanmalarını sağlayacak eşsiz bir kimliğe sahip olmaları, birbirleriyle veri paylaşımı yapabilmeleri, iletişim içerisinde olmaları ve veri işleyebilmeleri ortaktır.

IoT doğası gereği çok farklı nesneleri, iletişim teknolojilerini, ağın çeşitli türlerini barındırır. Çok karmaşık heterojen bir ağ yapısına sahiptir. Bu nedenle IoT ağının yönetilmesi oldukça zordur. IoT içerisinde standartların geliştirilmesi bu tip problemlere çözüm olarak görülmektedir ve üzerinde çalışmalar yapılmaktadır. Birçok başarılı çalışma yapılmıştır [29], [30], [31] fakat hızlı gelişim ve dinamik yapısı standartların geliştirilmesini zorlaştırır.

IoT, endüstriyel alanda, sağlık, eğitim gibi gerçek hayatta birçok alanda kendine yer bulmuştur. Bu anlamda IoT teknolojilerinde güvenlik ve güvenirlik dikkat edilmesi gereken noktalardan biri olmaktadır. IoT teknolojilerinin güvenliği ve güvenirliği için protokoller, standartlar, şifreleme algoritmaları geliştirilmiştir.

2.2.4. Siber fiziksel sistemler

Fiziksel dünyaya ait olan nesneler bilişim ve iletişim yetenekleri ile donatılmaktadır. Bu yeteneklere sahip olan nesneler, fiziksel dünyayı bilgisayar ve iletişimin siber dünyasına adapte eder. Böylece, insanların fiziksel dünya ile olan etkileşimi ve fiziksel dünyaya ait gücü artar.

Fiziksel dünyaya ait işlemlerin bilişim ve iletişim çekirdekleri tarafından izlendiği, kontrol edildiği, koordine edildiği fiziksel ve mühendislik sistemlerine siber fiziksel sistemler (CPS) adı verilir [32]. Gömülü sistemler, gerçek zamanlı sistemler, sensör sistemleri, dağıtık sistemler, network sistemleri gibi birçok sistemi içerisinde barındırır. Dinamik yapıda olan bu sistemleri bir bütün olarak izler, kontrol eder, sistemler arasındaki senkronizasyonu sağlar. Endüstri 4.0 için çekirdek bir teknoloji olan siber fiziksel sistemler, bulut imalat için çekirdek bir teknoloji olmamasına rağmen; bulut imalatın amacına tam olarak ulaşılabilmesi için kritiktir.

Sensörlerin maliyetinin düşmesi ve kapasitelerinin artması, bilişim cihazlarının maliyetinin düşmesi, kapasitelerinin artması, düşük güce ihtiyaç duyması ile birlikte kullanımın artması, kablosuz teknolojilerin gelişmesi, internetin yaygınlaşması ve bant genişliklerinin artması siber fiziksel sistemlerin kullanım alanlarını ve sıklığını arttırmıştır. Havacılık, sağlık sektörü, fabrika otomasyonu, süreç kontrolü gibi birçok alanda siber fiziksel sistemlerden faydanılmaktadır.

Siber fiziksel sistemler, gerçek dünya ile entegredir. Gerçek dünyanın dinamikliğine, esnekliğine, belirsizliğine uyum sağlamalıdır. Gerçek zamanlı olarak çalışmalıdır ve güvenirlikten, güvenlikten ödün vermemelidir. Sistemlerde meydana gelen hataları telafi edebilmeli veya beklenmedik kazalar ve arızalar, sağlam bir sistem tasarımı ile azaltılmalıdır.

Siber fiziksel sistemlerde üstesinden gelinmesi gereken problemler ve CPS teknolojileri yeni araştırma konularını ve zorluklarını beraberinde getirir. Birbirleriyle haberleşen birçok heterojen sistemi içerisinde bulundurması yeni uygulamaların ve

taleplerin oluşmasına sebep olur. CPS'nin tasarlanması ve geliştirilmesi; kontrol mühendisleri, çevre mühendisliği, yazılım mühendisliği, ağ bilimi, gömülü sistemler, sensör teknolojileri, sinyal işleme, biyoloji, sosyoloji, bilgisayar gibi birçok alanı bir araya getiren multidisipliner bir çalışmayı gerektirir.

2.2.5. Servis yönelimli mimari (SOA)

SOA, heterojen ortamlarda hızlı ve daha düşük maliyetlerle (zaman ve nakit) dağıtık yazılım uygulamalarının gelişmesini sağlayan, servis olarak adlandırılan birbirlerinden farklı, bağımsız, kendine yeten, gevşek bağlı yapıdaki bileşenlerin bir araya gelerek uyumlu çalışmasını sağlayan bir bilişim yaklaşımıdır [33], [34].

SOA sayesinde tek yapının parçalara ayrılarak ölçeklenmesi sağlanabilirken ayrıca farklı yapıların da birbirleri ile etkileşim kurması sağlanmış olur. SOA, standartlar çerçevesinde bileşenlerin birbirleri ile haberleşmesini sağlar. Farklı hizmetlerin sadece ilgili olan taraflarca sağlanması uzmanlaşmayı arttırır ve işin kaliteli yapılmasını ve sunulmasını sağlar. Belirlenen standartlar ile sunulan hizmetler aynı biçimde taraflar arasında paylaşılır. Bir SOA yaklaşımının bileşenleri aşağıdaki gibidir [35]:

- Servisler: SOA'nın temel öğesidir. Servisler sözleşmeler ile vaadedilen fonksiyonların en az hepsini içeren kod parçalarıdır.
- Sözleşme: Servisler tarafından desteklenen mesajların tümünü içerir.
- Poliçe: Hizmet tüketicileri için belirtilen iyi yönetilmiş koşulları temsil eder. Güvenlik, denetim gibi dinamik özellikler burada bulunur.
- Uç nokta: Servisin bulunduğu adresi temsil eder.
- Mesaj: Servisler için bir iletişim ortamıdır. Mesajlar SOAP, http, SMTP formunda olabilir. Mesajlar, başlık ve gövdeden oluşur. Başlık, mesaj içeriğini temsil eden çok genel bir ifadeyi barındırır. Mesajların daha kolay ve güvenli bir şekilde yönlendirilmesinde önemli bir rol oynarlar.
- Hizmet alan: Hizmet alan; yazılımın bir parçası olabilir veya servislerle etkileşimde bulunan kodun bir parçası olabilir.

SOA tasarımı, iş mantığı problemlerinin çözümünde yardımcı olan bir tasarımdır. Yazılımların, planlanan iş mantıklarının uygulanabilmesi ve geliştirilebilmesi adına SOA mimarisinin yazılımda kullanılması gerekir. Bu nedenle SOA, yazılım alanına daha yakın bir tasarıma sahiptir. Yazılım açısından bakıldığında, SOA mimarisi dağıtık bir yapının kurulmasını kolaylaştırır.

Büyük ve karmaşık problemlerin çözümlenmesi, çoklu işlemci kullanma ihtiyacı, ölçeklenebilirliği arttırma, sadece belirli donanımlar üzerinde çalışabilen yazılımların varlığı, güvenlik gibi sebeplerden ötürü dağıtık yazılımlara ihtiyaç duyulur. Dağıtık yazılımlarda tüm işler tek bir kod ile veya tek bir bilgisayar üzerinde yapılmaz. Problem daha küçük parçalar halinde fonksiyonlar şeklinde çözümlenir.

Farklı uygulamaların birbirleriyle haberleşebilmesi için ortak bir standarda ihtiyaç vardır. Aksi takdirde ortak bir dilde konuşulamayacağı için geliştirilmesi oldukça zor olacaktır. Fonksiyonların farklı dillerde farklı noktalarda sunulabilmesi için bir servis haline getirilmesi gerekir. Servis tabanlı mimari ile fonksiyonların bir servis olarak sunulması platform bağımlılığını ortadan kaldırır, yeniden kullanılabilirliği ve uyumluluğu arttırır.

2.3. Bulut Tabanlı Tasarım ve İmalat Yaklaşımının Kısıtları

Endüstri sektörü ve araştırmacılar tarafından bulut tabanlı tasarım ve imalat modeli farklı bakış açıları ile birçok açıdan yorumlanmıştır. Buna rağmen mimari yapısı, temel özellikleri, hizmet modu gibi kavramlarının daha doğru bir şekilde anlaşılabilmesi için CBDM'nin daha fazla tartışılmasına ve anlatılmasına ihtiyaç vardır. CBDM'nin ayrıntıları, CBDM uygulamasının hayata geçirilmesi aşamaları ve bu aşamada yaşanabilecek problemlerden, CBDM'nin sahip olduğu kısıtlardan bahsedilmelidir. Ayrıntıların eksikliği bulut imalatın geleceğini sekteye uğratır [36].

CBDM platformu en genel anlamda merkezi ve uzaktan erişilebilen bir sistem aracılığıyla herhangi bir yatırıma ihtiyaç duyulmadan kullandığın kadarını öde prensibiyle şirketlerin/kişilerin ihtiyaçlarını giderebileceği bir yazılım aracıdır.

Kullanıcılar, CBDM platformunun avantajlarından yararlanabilmek için bir yazılım arayüzü kullanırlar. CBDM kullanıcılarının 2 farklı türünden biri olan servis sağlayıcıları, bulut platformuna avantajlı kaynaklarını ve rekabet edebilecekleri yeteneklerini tanımlar ve burada pazarlar. Müşteri kullanıcı türü ise, özelleşmiş taleplerini platformda yayınlar ve sistemdeki kaynakları veya fonksiyonları keşfeder ve hizmet olarak kiralar. Müşteri ve kaynak sağlayıcı hizmet aldığı CBDM platformuna tek bir adresten ulaşabilir. Müşteri ve kaynak sağlayıcının yanında bu sürecin işleyişinden sorumlu platformun sahibi / sahipleri vardır. Platformun sahipleri CBDM platformunun erişiminden, yönetiminden, güvenliğinden, güvenirliğinden sorumludur.

Müşteriler ve kaynak sağlayıcılarının birbirleriyle iletişimi platform aracılığıyla (üçüncü parti bir aracı) gerçekleştirir. CBDM platformu güvenirlik ve güven temelleri üzerine kurulmalıdır. Güven ve güvenirlik sadece müşteri ve kaynak sağlayıcıları arasında değil platform kurucu / kurucularına ve platforma karşı da olmalıdır. CBDM platformunda bulunan veriler, üçüncü parti aracının elindedir.

Hizmetin, iletişimin sürekliliği sağlanmalıdır. CBDM platformlarının bir veya birkaç sunucu içerisinde barındırılması ve tek bir url adres üzerinden erişilebiliyor olması sürekliliğin ve iletişimin devamı için aslında bir tehdit oluşturmaktadır. İletişimin olmaması hizmetin olmaması anlamına gelir. Anlık bir hizmet kesintisi büyük maddi kayıplara neden olabilir. Sunucuların hizmet veremez olması, CBDM platformunu oluşturan şirketin/şirketlerin iflas etmesi, ortakların anlaşamaması gibi sebeplerden ötürü sadece anlık değil iletişim tamamen kesilebilir. CBDM platformu tamamen piyasadan çekilebilir. Böyle bir durumda CBDM platformunu kullanan kullanıcılar verilerini dışarıya çıkaramazlar. Kullanıcı verileri sistemde kilitli kalır.

CBDM platformları genel olarak yukarıda bahsedilen bir hareketliliğe sahiptir. CBDM uygulamaları incelendiğinde CBDM platformları ile yaşanabilecek problemlerin temelini aslında CBDM platformlarının sahip olduğu özelliklerin neden olduğu görülmüştür. CBDM platformlarının problemli olan noktalarından bazıları aşağıda özetlenmektedir [37]:

- CBDM platformları merkezi bir yapıya sahiptir ve merkezi sistemler güvenilir bir aracıya ihtiyaç duyarlar.
- CBDM platformları çoklu kullanıma izin verir. Kaynaklar birden fazla kişiye kiralanabilir. Ancak bu durum, farklı kullanıcılardan gelen verilerin merkezi sistemlerde tek bir noktada depolanmasına sebep olur.
- Kullanıcı verileri, uygulama sahibi tarafından tasarlanan biçimde saklanır ve yönetilir.
- Kullanıcılar bulut uygulaması tarafından izin verilen ölçüde kendi verilerine erişebilir.
- Bulut platformları arasında standart bir konuşma dilinin bulunmaması, verilerin uygulamada farklı formlarda tutulması veri erişimini zorlaştırır ve veri kilitleme sorununun temelini oluşturur.
- Sorumluluklar bulut üretim platformlarına dağıtılır. Sorumluluk alan şirketlere güvenmelisiniz. Sisteme bağımlılık yaratır [38].
- Kullanıcılar bulut uygulaması üzerinde sınırlı kontrole sahiptir [39].
- CBDM platformlarının gücü, platformu oluşturan kurumun gücü kadardır.

Bulut üretiminin bu özellikleri, ölçeklenebilirlik, birlikte çalışabilirlik, güvenilirlik, güvenlik, veri gizliliği, verimlilik, süreklilik ve esneklik açısından verimsiz olmasını sağlar [40], [41], [42].

2.4. Literatürde Geliştirilen Bulut Tabanlı Tasarım ve İmalat Modelleri

CBDM platformu, CBDM modelinin uygulamaya dökülmüş halidir. CBDM modeline ait katmanlı yapı CBDM platformları ile uygulamaya alınır. CBDM platformlarının geliştirilmesinde birçok sistem mimari yapısı sunulmuştur. Li ve arkadaşları [43] 5 katmanlı bir sistem mimari yapısı önermişlerdir. Bu katmanlar kaynak katmanı (source layer), sanal kaynak katmanı, çekirdek fonksiyon (core function) katmanı, uygulama arayüz katmanı, uygulama katmanı. Zhang ve arkadaşları [44] kaynak katmanı, algılama (perception) katmanı, servis katmanı, ara katman katmanı ve uygulama katmanını içeren 5 katmanlı bir CBDM mimari yapısı önermiştir. Ren ise fiziksel kaynaklar, bulut imalat platformu, servis uygulama katmanından oluşan 3 katmanlı bir

mimari yapısı önermiştir [45]. Xu; imalat kaynakları, sanal servis, global servis ve uygulama katmanından oluşan 4 katmanlı bir mimari yapı önermiştir [46]. Wu ve arkadaşları, insan-bilgisayar etkileşimi, ürün konfigürasyonu, hizmet kapsülleme, kaynak tahsisi katmanlarını içeren 5 katmanlı bir mimari model önermiştir [47]. Guo, Li ve arkadaşlarının önerdiği 5 katmanlı mimari model yapısını temel alarak bulut platformunu 2 temel alt katmana ayırmıştır: Bulut terminal alt sistemi ve bulut platform alt sistemi [48]. 2 temel alt katman 6 katman ile ayrıntılanmıştır. Bulut terminal alt sistemi fiziksel kaynak katmanı, sanal kaynak katmanı, kaynak statik yönetim katmanı ile bulut platform alt sistemi kaynak dinamik yönetim katmanı, uygulama arayüzü katmanı ve uygulama katmanı ile ayrıntılanmıştır. Thames ve Schaefer ise bulut platformunu donanım katmanı, kontrol katmanı ve sanal katman olarak basit bir mimari yapı önermişlerdir [49].

CBDM modelleri farklı ayrıntı seviyelerine sahip olabilir ama nihayetinde isimleri farklı fakat hedefleri aynı olan 4 temel katmana tüm modeller sahiptir [36]: sanal servis katmanı, global servis katmanı, uygulama katmanı ve kullanıcı arayüz katmanı. Farklı katmanların uygulamaya dökülmesi farklı teknolojileri gerekli hale getirir [50]. Üretim kaynaklarını algılamak ve fiziksel kaynakları sanal kaynaklara dönüştürebilmek için sanallaştırma teknolojilerine ve IoT teknolojilerine ihtiyaç duyulur. Bulut bilişim, servis ile ilgili teknolojiler ve semantik web teknolojileri global servis katmanının çekirdek teknolojileridir. Arayüz katmanı insan bilgisayar etkileşimin olduğu katmandır ve bu katmanda büyük veri analizleri, gelişmiş imalat modelleri, yüksek performanslı hesaplama teknolojileri büyük önem taşımaktadır [51].

Önerilen mimari yaklaşımlar, bulut tabanlı tasarım ve imalat platformu aracılığıyla merkezi bir şekilde çalışır. Tüm bulut ağı arabulucu olarak da adlandırılan tek bir veya küçük bir grup şirketler tarafından denetlenir ve yönetilir. Merkezi yapı esneklik, şeffaflık ve ölçeklenebilirlik konusunda kısıtlıdır. Bu problemleri çözebilmek adına merkezi olmayan bulut imalat mimarileri üzerine çalışmalar yapılmaya başlanmıştır. Skulj ve arkadaşları, bulut üretimi için merkezi olmayan bir ağ mimarisi önermektedirler [40]. Önerilen mimari, özerk çalışma sistemini (AWS: özerk çalışma sistemi) [52] temel almaktadır. Bu mimari, ağda özel bir şirketin bulunmasını

gerektirmez, ağda özerk çalışma sistemi entegrasyonunu gerektirir. Merkezi olmayan bulut imalat modeli imalat servis sağlayıcıları ve imalat servis son-kullanıcıları olmak üzere 2 katmanlı bir yapıda tasarlanmıştır.

Bulut tabanlı tasarım ve imalat modelinin merkezi olmayan bir mimari yapısı için son yıllarda blokzinciri teknolojisinden faydalanılmaktadır. Blokzinciri teknolojisi merkezi olmayan dağıtık bir mimari yapıya sahiptir. Yu ve arkadaşları blokzinciri destekli bir bulut imalat mimarisi önermektedir [53]. Önerilen blokzinciri destekli bulut imalat mimarisi üretim kaynaklarını bulur ve bu kaynakları hizmet olarak sunar. Blokzinciri ağı işlem sonuçlarını kaydetmek ve hizmet yapısına aracılık etmek amacıyla kullanılmaktadır. Li ve arkadaşları CBDM'nin güvenliğini ve ölçeklenebilirliğini iyileştirebilmek için dağıtık noktadan noktaya bir ağ platformu tasarlamışlardır [54]. BCmfg isimli platformun tasarlanmasında blokzinciri teknolojisinden faydalanılmıştır. Bu çalışmada kullanılan BC ağı, başka amaçlar için kullanılan genel bir BC ağı değildir, yalnızca bu model için geliştirilmiştir. Bu BC ağındaki veriler şifrelenir ve tüm veriler aynı anda hem veritabanına hem de ağa kaydedilir. BCmfg kaynak katmanı, algılama (perception) katmanı, imalat katmanı, altyapı ve uygulama katmanlarından oluşmaktadır. Bir başka blokzinciri destekli bulut imalat modeli Ali ve arkadaşları tarafından önerilmiştir [41]. Birçok bulut imalat sağlayıcıları için noktadan noktaya merkezi olmayan bir ağ yapısı ortaya koymaktadırlar. Lee ve arkadaşları siber fiziksel üretim sistemlerinde BC teknolojilerinin kullanımı ve karşılaşılan problemler konusunda bir çalışma yapmışlardır [55].

Yapılan bu çalışmalarda farklı katman sayısı ile farklı mimari yapıları ortaya konulmuştur. CBDM'nin merkezi bir yapıya sahip olmasından kaynaklanan kısıtlamalar blokzinciri teknolojisinden faydalanılarak giderilmeye çalışılmıştır. Blokzinciri teknolojisinin farklı türleri (özel, genel, konsorsiyum) ile farklı BC destekli CBDM mimarileri geliştirilmeye çalışılmıştır. Kaynak ve arkadaşları BC ağının genel türünü kullanarak farklı bir BC destekli bulut imalat modeli önermişlerdir [37]. Genel Ethereum BC ağı kullanılmaktadır. Geliştirilen BC destekli bulut imalat

uygulaması hibrit bir yapıdadır. Veriler yalnızca BC ağında tutulmaz, verilerin bazıları kullanıcı bilgisayarın yerel veritabanlarında tutulur.

2.5. CBDM'nin Uygulanmasında Karşılaşılan Problemler

CBDM'nin tanımı, sınırları, yetenekleri yeterince belirgin değildir ve uygulamaya geçilebilmesi için ayrıntıların bilinmesi gerekir. CBDM uygulanması CBDM'nin tüm yaşam döngüsünün planmasını ve ayrıntılı bir şekilde tasarlanmasını gerekli kılar. CBDM ile nelerin yapılabileceği, yetenekleri, ne zaman tercih edilmesi gerektiği, avantajları ve dezavantajları daha somut hale getirilmelidir.

Bulut üretimin başarısı, işletmelerin yaygın bir şekilde CBDM'yi benimsemesine ve katılımına bağlıdır [36]. İşletmeler bulut üretimin ne olduğunu, neler yapabileceğini/yapamayacağını, avantajlarını, dezavantajlarını, diğer imalat modellerden farklılıklarını, CBDM'nin adım adım uygulama ayrıntılarını, CBDM uygulamaları sırasında yaşanabilecek riskleri, uygulama maliyetlerini bilmeleri gerekir. İşletmelerin CBDM'yi uygulayabilmeleri için birçok açıdan hazır olmaları gerekir. Öyle ki bazı şirketler gerçekten CBDM'ye ihtiyaçları olup olmadığından bile emin değildirler.

CBDM platformlarında yer alan kaynaklar çok çeşitlidir ve onları tanımlamak oldukça zordur. Kaynakların bilişim dünyasına taşınabilmesi için sanallaştırma teknolojisine ihtiyaç duyulur. Kaynakların çok çeşitli ve karmaşık olması bu katmanın uygulamaya dökülmesini oldukça zorlaştırır. Sisteme tanımlanan kaynaklar hizmete sunulur.

CBDM'yi uygulayabilmek için sadece sanallaştırma değil, çok sayıda teknoloji desteğine ihtiyaç vardır. Li ve arkadaşları bulut üretim için kilit teknolojileri 8 kategoriye ayırdı [56]: bulut bilişim, semantik web, servis odaklı teknolojiler, sanallaştırma, gelişmiş yüksek performans, bilgi işlem teknolojileri ve gelişmiş üretim modelleri. Bu teknolojiler CBDM'nin uygulanmasında çekirdek yapıda teknolojilerdir. Bu teknolojilerin yanında yapay zekâ, mobil internet gibi

teknolojilerde CBDM'nin uygulanmasında faydalanılan teknolojilerdir. Fakat uygulama esnasında hepsine ihtiyaç duyulmaz.

CBDM teknolojilerinin ne olduğu, uygulamanın hangi aşamalarında hangi teknolojiye ihtiyaç duyulacağı, teknolojilerin etkili bir şekilde nasıl entegre edileceği CBDM'nin uygulanabilmesi için gerekli olan bilgilerdir. Bu bilgilere sahip olmayan şirketler CBDM uygulamaların ve teknolojilerin karmaşıklığından korkabilirler. Ayrıca; bu teknolojilerinde olgunlaşmasına ihtiyaç vardır. Bu teknolojiler tam olarak oturmadan CBDM tam olarak uygulanamaz [36].

Müşteriler, ihtiyaç duyduğu hizmeti kullandığın kadar öde prensibiyle alır ve sisteme geri bildirimde bulunur. Müşterilerin hizmet aldığı servis sağlayıcısı ile ilgili değerlendirmesi ve oylaması rekabetin artmasını sağlar. Adil bir değerlendirme sistemi CBDM platformları için şarttır. Bu nedenle sadece öznel bir değerlendirme değil, geçmiş verilerden faydalanacak şekilde objektif bir değerlendirmeyi de içermelidir. Bu durum müşterilerin güvenilir hizmet sağlayıcıları seçmeleri için önemlidir.

CBDM platformları uygulama arayüzleri kullanıcı-makine etkileşimin olduğu katmandır. CBDM platformları bir veya birden fazla amaca göre tasarlanır. Kullanıcı arayüzleri CBDM platformlarının amacını karşılamalıdır. Böyle kullanıcı dostu bir arayüz hazırlamak oldukça zordur. Hele ki genel bir amaca sahip olan CBDM platformunun hazırlanması ve kullanıcı arayüzünün hazırlanması oldukça emek ister. Kaynakların genel bir şekilde tanımlanması, kaynak sağlayıcı ve müşterilerin birbirlerine eşleştirilmesi genel CBDM platformları için daha zor bir hal almaktadır.

2.6. CBDM Platformlarında Kaynak Seçimi

Servis seçimi, bulut imalat için hayati öneme sahip imalat görevlerini uygulamak için kritik öneme sahiptir. CBDM platformları servis seçimi ve planlamasında farklı birkaç özelliğe sahiptir [7]:

- CBDM platformları hizmet yönelimlidir ve müşteri odaklıdır.
- Çoğu bulanık ve nicel olmayan özelliklere sahip olan CBDM platformlarında daha fazla optimizasyon hedefinin (ürün kalitesi, hizmet değerlendirmesi, sistem sürdürülebilirliği, enerji, fayda, güven) dikkate alınması gerekir.
- CBDM platformlarında üretim kaynakları daha geniş bir alanda ve daha esnek bir şekilde paylaşılmaktadır. CBDM platformlarına kaynaklar dinamik olarak girebilir ve çıkabilir.

CBDM platformlarında optimum ve kişisel amaca uygun kaynak seçimini oluşturabilmek için bir matematiksel modele ve algoritmaya ihtiyaç vardır. Bu konu ile ilgili yapılan çalışmalar daha başlangıç aşamasındadır [7].

Cheng ve arkadaşları sanal kaynaklar arasında bulunan korelasyon ilişkisini ve özelliklerini dikkate alan kaynak seçimi ve optimum planlama için genetik algoritma temelli bir formül geliştirmişlerdir [57]. Wang ve arkadaşları ise CBDM platformlarında imalat makinelerinin optimum seçimi için parçacık sürü optimizasyon algoritmasını temel alan yeni bir model geliştirmişlerdir [58]. Tian ve arkadaşları CBDM'de optimum kaynak seçimi için melez arı algoritması geliştirmişlerdir [59]. Çalışmalarında QoS kısıtlamaları dikkate alınır. Cao ve arkadaşları TCQS (zaman, maliyet, kalite, servis) kriterlerini dikkate alarak servis seçimi ve planlama modeli geliştirmişlerdir [7]. Bu çalışmada TCQS kriterlerini göreceli üstünlük derecelerine dönüştürebilmek için bulanık karar verme teorisi benimsenmiştir. TCQS kriterlerinin ağırlık katsayıları analitik hiyerarşi süreci (AHP) ile hesaplanır ve doğrusal bir amaç fonksiyonuna dönüştürülür. Amaç fonksiyonu ile karınca kolonisi optimizasyon algoritması servis seçimi ve planlama modeli için kullanılır. Wang ve arkadaşları tarafından CBDM platformlarında dağıtık imalat kaynaklarının seçimi için dağıtık genetik algoritmayı (DGA) temel alan bir model geliştirilmiştir [60]. Geliştirilen model, yanıtlama hızı, optimum sonuca yakınsama oranı ve global optimum sonucu arama yeteneği bakımından Cao2016 tarafından geliştirilen model ile karşılaştırılmış, DGA modelinin daha iyi etkiye sahip olduğu savunulmuştur. Zhou ve Yao, QoS özelliklerini dikkate alacak şekilde optimum servis seçimi için hibrit yapay arı kolonisi (HABC) ismi verdiği yeni bir yaklaşım önermiştir [61]. HABC algoritması genetik

algoritma, basit yapay arı koloni algoritması, parçacık sürü optimizasyonu (PCO) algoritmalarından daha iyi kaynak seçim çözümleri üretmektedir. Que ve arkadaşları da QoS özelliklerini dikkate alacak şekilde CBDM platformlarında kaynak seçimi için geliştirilmiş adaptif bağışıklık genetik algoritması geliştirmişlerdir [62]. Öncelikle kullanıcıların QoS gereksinimleri karşılayacak en iyi çözümü bulabilmek için kapsamlı bir matematiksel model geliştirmişlerdir. Her bir QoS özelliği kullanıcı isteğine göre bir ağırlığa sahip olur. Daha sonra QoS özellikleri dikkate alınarak bir kaynak seçimi için IEIGA algoritması geliştirilmiştir. IEIGA algoritması adaptif genetik algoritma ile bağışıklık optimizasyon algoritmasının entegre edilmesiyle geliştirilmiştir. Li ve arkadaşları QoS kısıtlarını da dikkate alacak şekilde PSO temelli bir model geliştirmişlerdir. QoS kısıtları olarak zaman, maliyet, kullanılabilirlik, güvenilirlik özellikleri alınmıştır [63]. Aynı amaçla Zhou ve arkadaşları [64] temel ABC algoritmasını temel alarak MS-DABC isimli bir algoritma, Zhou ve Yao [61] ABC algoritmasını temel alarak DE-caABC isimli bir algoritma geliştirmişlerdir.

Son yıllarda yapılan çalışmalarda kaynak seçiminde QoS özelliklerinin dikkate alındığı, QoS parametrelerinin, daha birçok nicel ve bulanık özelliklerin kullanıcıların isteklerine göre ağırlıklandırıldığı görülmüştür. Tek bir iş parçacığı için kaynak seçimini gerçekleştiren çalışmalar olduğu gibi alt iş parçacıklarından oluşan bir iş için kaynak seçimi gerçekleştiren çalışmalar da mevcuttur. Kullanıcı isteğine göre ağırlıklandırılan özellikler temel alınarak yapay zekâ tekniklerinden, matematiksel modellerden, istatistiksel yöntemlerden faydalanarak yeni modeller geliştirilmiştir.

Literatürde yapılan tüm çalışmalar kıymetlidir. Bu çalışmaları birbirlerinden farklı kılan özellikler şunlardır:

- Farklı bulanık ve nicel olmayan özellikler dikkate alınır.
- QoS özelliklerinin, diğer nicel ve bulanık özelliklerin korelasyon ilişkileri ve özelliklerin kaynak seçimine olan etkileri incelenir.
- Kaynak seçimi için farklı yapay zekâ teknikleri, istatistiksel teknikler bir arada kullanılarak yeni hibrit modeller geliştirilmiştir.

Tezin, kaynak seçimi ile ilgili kısmında, birçok alt iş parçacığından oluşan bir işin kaynak seçimi için AHP ve genetik algoritmayı temel alan yeni bir model geliştirilmiştir. Geliştirilen yeni model zaman (T), maliyet (C), kalite (Q), risk (R)-TCQR özelliklerinin yanında önceden seçilmiş şirketler (preselected company-P), erken teslim ve geç teslim özellikleri de dikkate alınmıştır. Kullanıcılar, önceden belirlediği ve tercih ettiği şirketleri öncelikli olarak kullanmak isteyebilirler. P özelliği önceliklendirilmiş olan bu kaynakların diğer kaynaklara göre daha ön sıralarda çözüm listesinde gösterilmesini sağlar. QoS parametreleri dikkate alınarak genetik algoritma ve AHP yöntemiyle kullanıcı kriterlerine en uygun sonucu veren optimum kaynak seçimi gerçekleştiren yeni bir model geliştirilmiştir. Geliştirilen model farklı QoS özelliklerini dikkate alması ve yeni hibrit modeliyle diğer çalışmalardan farklılıklar oluşturmaktadır.

BÖLÜM 3. MERKEZİ OLMAYAN UYGULAMALAR (DApp)

Merkezi olmayan uygulamalar (DApp), merkezileşmenin sebep olduğu problemlerden kaçınmak için tercih edilen, kullanıcıların bilgilerini işlemek ve yönetmek için bir aracıya ihtiyaç duymayan, kullanıcılar ile sağlayıcıları doğrudan birbirlerine bağlayan uygulamalardır. DApp'lar genellikle blokzinciri kullanılarak inşa edilirler. Fakat farklı platformlarda da DApp'lar geliştirilebilir. Blokzinciri teknolojisinin gelişimi ile birlikte DApp'ların gelişimi ve kullanımı artmıştır.

Bu bölümde klasik DApp uygulamaları ve blokzinciri (BC) destekli DApp uygulamalarının farklılıkları, blokzinciri teknolojisi, DApp uygulamalarının farklı amaçlara hizmet vermesini yaygınlaştıran akıllı sözleşmeler, BC destekli DApp uygulamalarının özellikleri ve bulut imalat sektöründe BC destekli DApp'ın yeri ve önemi bahsedilmektedir.

3.1. Tanım

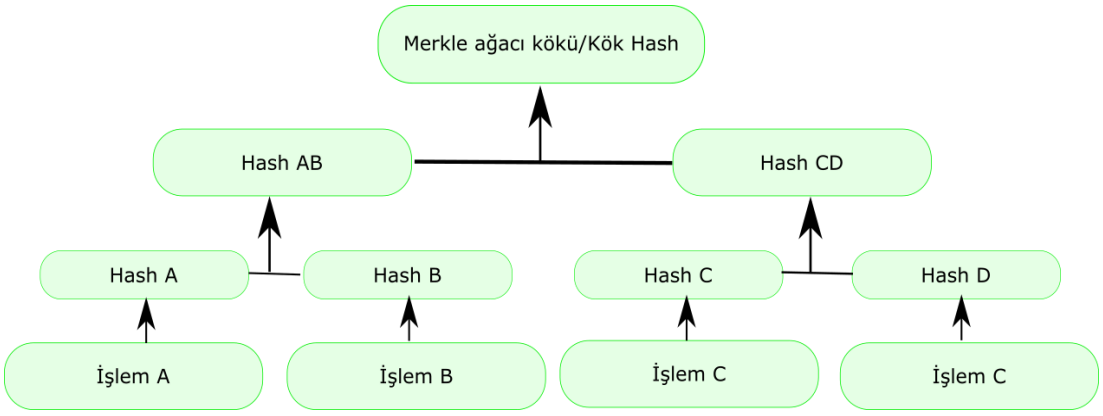
Merkezi olmayan bir mimari yapıya sahip uygulamalar, P2P (eşten eşe bağlantı) ağında çalışan genellikle açık kaynak kodlu ve iş birliği ile karar verme mekanizmalarına sahip dağıtılmış uygulamalardır. Uygulama verileri tek bir sunucuda tutulmaz, ağdaki birden fazla düğümde uygulama verilerinin kopyası bulunur.

Torrent uygulamaları, P2P ağlarında çalışan blokzinciri kullanılmadan geliştirilen geleneksel DApp'lara örnektir. Büyük boyutlu dosyalar torrent uygulamasıyla paylaşılır. Torrent uygulamasına ait dosyaya ve arama motorlarına erişim sağlayan birçok web sitesi vardır. Böylece bir düğüm kaybı ve düğüme erişim kaybı sisteme erişimi engellemez. Dosya paylaşan kişi sayısı arttıkça, dosya daha hızlı indirilir. Paylaşım durduğunda, ağ iletişimi durur, önceki adımlar ve paylaşılan dosya kaybolur.

3.2. Blokzinciri Teknolojisi

Blokzinciri teknolojisi, veri kayıtlarının bir listesini tutan dağıtık bir veritabanı olarak veya katılımcılar arasında tüm süreçlerin çalıştırılması ve paylaşılması için genel bir muhasebe defteri olarak görülebilir [65]. Klasik veritabanı ve muhasabe defterinin aksine merkezi olmayan ve dağıtık olan bir yapıya sahiptir.

Blokzinciri teknolojisi Merkle/Hash ağacı olarak bilinen bir veri yapısının üzerine geliştirilmiştir. Merkle ağacı büyük verilerin verimli ve güvenli bir şekilde doğrulanmasında kullanılan ikili bir ağaç yapısıdır; kök, düğüm ve yapraklardan oluşur. Merkle kök, merkle ağacının kök düğümüdür. Merkle kök kendisine bağlı olan tüm düğümlerin kriptografik özet hash bilgilerini barındırır. Yaprak olmayan düğümler kendi çocuk düğümlerinin kriptografik hash değerleri ile etiketlenir. Her yaprak düğüm ise veri bloğunun şifrelenmiş hash değeri ile etiketlenir (Şekil 3.1.). Kök düğüme kadar yani tek bir hash değeri elde edilene kadar bu şekilde devam edilir.



Şekil 3.1. Merkle/Hash ağacı yapısı.

Ağacın herhangi bir düğümünde bir değişiklik meydana geldiğinde o değişikliğe ait yeni bir hash değeri üretilir. Bu değişiklik merkle kökünün hash değerinin değişmesine neden olur. Orijinal dosyanın merkle ağacı ile kökten yapraklara doğru hash değerleri karşılaştırılarak değişiklik yapılan veri bulunabilir. Bu şekilde bir oluşum büyük miktardaki verilerin etkili ve güvenli bir şekilde doğrulanmasını sağlar. Verinin değiştirilmemesi garanti altına alınır. Günümüzde genellikle merkle ağaçları bir ağ

üzerinden noktadan noktaya iletilen veri bloklarının orijinal haliyle hatasız alınmasını sağlamak amacıyla kullanılır.

Merkle ağacının sunduğu BC ağına klasik BC adı verilmektedir. Günümüzde kullanılan BC teknolojisi klasik BC ağından farklıdır. Sadece merkle ağacının sunduğu avantajları barındırmaz. İçerisinde ağ üzerinde demokrasiyi sağlayan uzlaşma algoritmalarını ve dağıtık mimarilerin müzdarip olduğu çift harcama problemini çözebilmek için PoW (iş ispatı) algoritmalarını da barındırır. Uzlaşma algoritmaları sayesinde merkezi olmayan bir sistem elde edilir. BC ağında bulunan her düğüm uzlaşma algoritmaları sayesinde eşit haklara sahip olur ve ağın çıkarları bireysel çıkarların üstündedir.

Satoshi Nakamoto tarafından 2008 yılında Merkle ağacı temel alınarak bilgisayarlar arası ağ ile noktadan noktaya güvenli bir veri alışverişi sağlayacak, veri geçmişini içerecek, merkezi bir otorite olmadan yönetilebilecek bir blokzinciri uygulaması geliştirildi [66]. Uygulama, herhangi bir aracı olmadan bir noktadan diğerine çevrimiçi olarak ödemelere izin vermektedir [67]. Geliştirilen blokzinciri uygulamasına bitcoin ismi verildi. Bitcoin kriptopara biriminin temel özellikleri kullanılarak benzer uygulamalar geliştirildi ve uygulamalar kriptopara terimi ile etiketlendi.

BC teknolojisi, bilgisayar biliminde kriptografi ve veri yapılarında karşımıza çıkan eski bir terim olmasına rağmen, bitcoin uygulamasının başarısıyla popüler oldu. Günümüzde kullanılan BC ağı ile daha güvenilir farklı uygulamalar geliştirildi. Sağlık, endüstri, tedarik zinciri, eğitim, sayısal kimlik, oy kullanma, telif kayıt sistemleri, tapu kayıt sistemleri gibi birçok alanda BC uygulamalarına rastlanmaktadır [68].

Bir blokzinciri uygulaması aşağıdaki temel özellikleri içerisinde barındırır [69], [70]:

- BC dağıtık bir mimari yapıya sahiptir, merkezi olmayan bir ağ üzerinde çalışır.
- İşlemler P2P ağda tüm düğümlere yayınlanır.
- İşlemler birden fazla düğüm tarafından onaylanır ve onaylanan düğümler zincire dâhil edilirler.

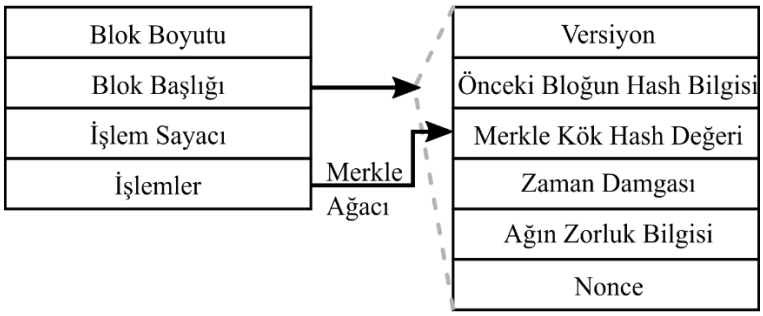
- Şeffaflık: İnsanların takibi ve kontrolü yerine ağın kurallarına göre ağın yönetimi gerçekleştirilir.
- Ağda kararlar uzlaşma protokolleri kullanılarak ağın lehine olacak şekilde verilir.
- Ağa eklenen veriler değiştirilemez.
- Bloklar birbirlerine hash bilgisiyle bağlıdır. İşlemler, başlangıç noktasından son duruma kadar izlenebilir.
- Sistem tek bir varlığa ait değildir. Merkezi otoriteye ihtiyaç duymaz.
- Yapılan işlemler tekrarlanamaz.
- Veri bütünlüğünü destekler.

3.2.1. Mimari yapısı

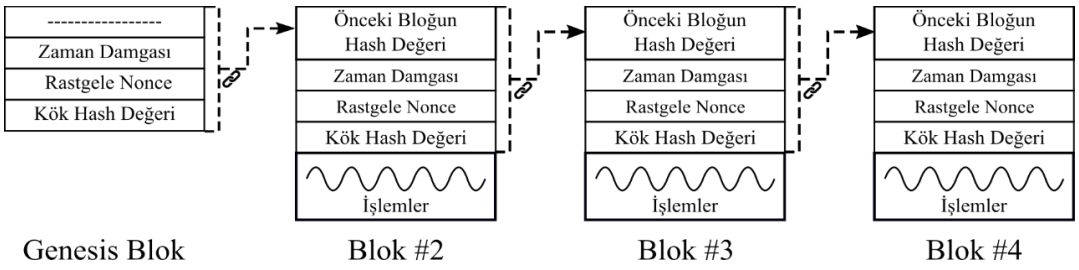
Bitcoin uygulaması klasik blokzinciri teknolojisine dayanmaktadır. Klasik BC teknolojisi veri senkronizasyonu, dağıtılmış sistemlerde bulunan çift harcama problemlerinden müzdariptir. Uzlaşma algoritmaları uygulanarak veri senkronizasyonu problemi çözülmeye çalışılmıştır. Dağıtık sistemlerde yaşanabilecek çift harcama problemi PoW uzlaşma algoritmasını temel alan merkezi olmayan bir ödeme sistemi ile çözülmüştür [71], [72]. Bu çözüm ilk önce Satoshi Nakamoto tarafından bitcoin makalesinde uygulanmıştır [73]. Yapılan iyileştirmeler ile birlikte, şu anda kullanılan BC teknolojisi geleneksel BC teknolojisinden farklıdır. Bu tez çalışması kapsamında geliştirilen DApp uygulamasının faydalandığı BC teknolojisi, uzlaşma ve PoW modellerine dayanan P2P iletişimine sahip merkezi olmayan ve tamamen dağıtılmış bir sistemdir.

Çalışmada kullanılan BC teknolojisinin genel mimarisi Şekil 3.2.'de gösterilmektedir. Şekil 3.2.a'da görüldüğü gibi her bir blok; blok boyutu, blok başlığı, işlem sayacı ve işlem alanlarına sahiptir. Blok başlığı alanının kendi içerisinde 6 alanı vardır: versiyon, bir önceki bloğun hash bilgisi, merkle kök alanı, zaman damgası, ağın zorluk bilgisi ve nonce (yalnızca bir kez kullanılan sayı). Versiyon alanı yazılım protokolünü izlememizi sağlar. Bir önceki bloğun hash bilgisi veri güvenliğini sağlayan alandır. İşlem alanında herhangi bir değişiklik meydana gelirse bir önceki bloğun hash bilgisi

değişir. Merkle kök alanı, blokzincirinde bulunan merkle ağacının hash değerini tutar. Ağa bildirilen işlemler uzlaşma algoritmalarıyla onaya sunulur. Onaylanan işlemler merkle ağacına eklenir. İşlem alanında herhangi bir değişiklik olduğunda, işlemle ilişkili yerel ağacın kök hash değeri değişir ve değiştirilen bloktan sonra oluşturulan tüm bloklar kaybolur. Kazma güçlüğü alanı PoW algoritmalarının zorluk derecesini belirtir. Nonce alanı ise PoW algoritması için kullanılan rastgele bir değerdir. Nonce, "Number Only Used Once" ifadesinin kısaltmasıdır. Zorluk derecesi arttıkça nonce değerini bulmak zorlaşacaktır.



a. Blokzinciri ağında bulunan bir bloğun yapısı.



b. Blokzincirinin yapısı.

Şekil 3.2. Blokzinciri ağ yapısı.

Şekil 3.2.b'de görüldüğü gibi blokzinciri ağı genesis blok ile başlar. Zincirin başlangıç bloğudur. Herhangi bir bloğa bağlı değildir. Genesis blokzinciri başlatan kişi tarafından oluşturulur. Genesis blok, ağın temel kurallarını ve yapılarını içerir ve diğer bloklardan farklı alanlara sahiptir.

3.2.2. Çalışma yapısı

Kullanıcılar ağ üzerinde işlem yapabilmek için ağa kaydolmalıdır. Kayıtlı kullanıcının ortak ve özel anahtarları vardır. Özel anahtarı olan herkes bu hesap için tüm işlemleri yapmaya yetkilidir. Ağda işlem yapanın kimlik veya diğer kişisel tanımlayıcı detayları bilinmez. Genel anahtar cüzdan adresini temsil eder. Genel anahtarı olan ağda bulunan tüm düğümler Ethereum ağında doğrulama gerçekleştirebilir.

BC destekli merkezi olmayan ve dağıtık olan uygulamalarda:

- Bir işlem (transaction) oluşturulur.
- Düğüm olarak oluşturulan işlem, bilgisayarların P2P ağına iletilir.
- Ağa iletilen işlem BC türüne göre bir grup bilgisayar tarafından, tek bir bilgisayar tarafından veya tüm ağ tarafından bir uzlaşma algoritması ile doğrulanır.
- Doğrulanan işlem dijital imzalı olarak paketlenir ve bir sonraki bloğa bağlanır. Böylece BC ağına yeni bir veri bloğu eklenmiş olur.
- Onaylanan işlem artık BC ağında kalıcı olarak saklanır. Bloklar blokzinciri ağında bulunan herkes tarafından görülür.

Bloklar bir dizi yüksek işlem gücü gerektiren kriptogtafik süreçlerden geçerek oluşturulurlar. Hazırlanan bloğa ait hash değeri belirli bir formata (tanımlanmış bir değer aralığı içerisinde olma, belirli bir karakter dizisi ile başlama gibi) sahip olmalıdır. Bu format kazma güçlüğü alanında bulunan değerle ayarlanır. Bu değer her 10 dk.'da bir yeni bir blok üretilebilecek şekilde ayarlıdır. Kazma güçlüğü arttıkça verilen formatta hash değerini bulmak daha da zorlaşır.

Kazma güçlüğü hash değerinin başlangıç kaç bitinin 0 değerine sahip olduğuyla ilgilidir. Hash fonksiyonlarından biri olan SHA256 fonksiyonuna bir girdi verdiğimizi düşünelim. Bu fonksiyonun çıktısının (hash değeri) ilk biti 1 veya 0'dır. İlk bitin 0 bulunması en fazla 2 denemeyle gerçekleştirilebilir. Eğer ki format ilk 2 bitinin 0 değerine sahip olması ise, bu formata uygun bir değer üretilebilme ihtimali %25'e

düşer. Bu şekilde en başta istenilen sıfır sayısı arttıkça bu formata uygun değerin üretilebilmesi için deneme sayısıda artar. Bu da işlem gücü ve zaman ihtiyacı gerektirir.

Daha önceden belirlenen formata uygun hash değerlerin bulunması işlemini madenciler gerçekleştirir. Madenciler, olabilecek her türlü girdiyi deneme yanılma yöntemiyle bulmaya çalışır. İstenilen formatı bulan ilk madenci bloğu blokzincirine eklemeye hak kazanır ve yaptığı iş karşılığında ödüllendirilir. Bütün madencilerin ödülü alabilme şansı vardır. İşlem gücü ne kadar yüksek ise o kadar çok girdi test edilebilir. Bu da şansı arttırır. Bloğun üretilmesi ortalama 10 dk. olarak belirlenmiştir. Zorluk derecesi maksimum kullanıcı sayısı, akım gücü ve ağın genel yüküne bağlı olarak bu süreye göre ayarlanır. Madencilik sürecini şu şekilde özetleyebiliriz:

- Yeni blok içerisinde yer alacak işlemler seçilir. İşlemler katılımcılara iletilir.
- İşlemler kullanılarak merkle ağacı yapısı ve merkle ağacının kök yapısı oluşturulur.
- Merkle kök değeri, bir önceki bloğun hash değeri, zaman damgası, kazma güçlüğü ve nonce değerleri girilir, blok başlığı oluşturulur.
- Blok başlığına ait hash değeri üretilir. Hash değerinin istenilen formata uygun olup olmadığı kontrol edilir. (PoW algoritması çalıştırılır.)
- Eğer uygun hash değeri oluştuysa yeni blok başarılı bir şekilde oluşmuş olur ve tüm ağa blok bildirilir.
- Eğer işlemler başarılı ve daha önceden harcanmamışsa katılımcılar bloğu kabul eder. Kabul edilen bloğun hash değeri bir sonraki blok için girdi olur.
- Eğer uygun bir hash değeri üretilemediyse nonce değeri arttırılarak uygun hash değeri bulunmaya çalışılır. Nonce değeri sınıra geldiyse ve hala blok üretilemediyse blok başlığını oluşturan diğer değerler tekrar ele alınır, güncellenir ve akış tekrar baştan alınır.

PoW algoritması işe başlamak için blok üretebilmek için kullanılan bir algoritmadır. Blok üretiminde yapılan işlerin matematiksel olarak ispatı PoW (çalışma kanıtı) algoritması ile gerçekleştirir. PoW algoritmasında madenciler ağdaki işlemleri

tamamlamak ve ödüllendirilebilmek için birbirleriyle yarışırlar. Kurallara uygun bir bloğun oluşturulabilmesi için ciddi bir hesaplama gücüne ihtiyaç duyulur. En yaygın kullanılan kriptoparalardan olan Bitcoin ve Ethereum PoW algoritmasını kullanır. PoW algoritmasına ait ayrıntılar "3.2.3. Uzlaşı algoritmaları" başlığı altında verilmektedir.

3.2.3. Uzlaşı algoritmaları

Merkezi yapılarda kararlar tek bir lider tarafından alınır. Merkezi olmayan yapılarda kritik kararlar uzlaşma algoritmalarıyla alınmaktadır. Uzlaşma algoritmalarında ağda bulunan herkes fikir beyan edebilir. Tüm anlaşmaları, fikirleri önemser ve onları toplar. Her fikir eşit düzeyde ayrıcalıklıdır. Nihai karar ağın çoğunluğunun temsil ettiği fikirden yanadır (Şekil 3.3.). Uzlaşma algoritmaları ile bireysel çıkarlar değil ağın çıkarları düşünülür ve herzaman ağ için bir kazanç elde edilir.

Şekil 3.3. Uzlaşma algoritmalarında nihai karar.

Birçok blokzinciri uzlaşma algoritması geliştirilmiştir. Ethereum ve Bitcoin BC uygulamaları PoW uzlaşma algoritmasını kullanır. PoW uzlaşma algoritmasından sonra birçok uzlaşma algoritması geliştirilmiştir. Bu uzlaşma modelleri üç farklı uzlaşma protokolü ve bu protokollerin çeşitlendirilmiş hali olarak görebiliriz [74].

- Emek kanıtı (Proof of Work) uzlaşma protokolü
- Hisse kanıtı (Proof of Stake) uzlaşma protokolü
- Bizans hata toleransı (BHT) modeli

3.2.3.1. Emek kanıtı (PoW) uzlaşma protokolü

Emek kanıtı (Proof Of Work - PoW), Bitcoin ağında kullanılan bir uzlaşma protokolüdür. PoW protokolünde madenciler, daha önceden belirlenmiş hash değerini bulabilmek için karmaşık hesaplamalar yapmaktadır [75]. Belirlenen hash değerine ulaşan ilk madenci zincire bloğu eklemeye hak kazanır. Oluşturulan yeni blok, elde edilen hash değeri ile birlikte ağa sunulur ve ağda bulunan kullanıcıların onayına sunulur. Onaylanan blok ağa eklenir ve tüm madenciler yeni bloğu zincirlerine ekler. İlk uygun hash değerini bulan madenci ödüllendirilir.

PoW protokolünde hesaplamaya dayalı zor problemler sunulur fakat çözümlerinin doğrulanması işlemleri kolaydır. Diğer madenciler tarafından problemin çözümü kolaylıkla doğrulanabilir. Bu protokolün bir diğer önemli özelliği problemin çözümü için yapılan geçmişteki işler, gelecekteki bulmacaları çözme olasılığını etkilemez [74]. Bulmacalar birbirlerinden bağımsızdır.

Dağıtık ağ ortamlarında aynı anda hash değerine ulaşan birden fazla madenci olabilir. Bu problem çift harcama problemi olarak anılır. Bu durumda BC ağında çatallar oluşur. Çatalı oluşturan madencilerin bir sonraki bloğun hash değerini aynı anda bulma ihtimalleri yoktur [75]. PoW protokolünde böyle bir durumda çatalın uzun kolu seçilir. Kısa çatalda bulunan madenciler, uzun çatalla geçerek bir sonraki bloğun üretilmesi için çalışırlar.

Madenciler bulmacaları çözerek, yapılan işlemleri kontrol ederek sistem üzerinde beklenmedik değişikliklerin yapılmasını engeller. Madenci sayısının artması, ağda bulunan kişi sayısının artması, yazılım ve donanımsal gelişmeler bulmacaların çözülmesini kolaylaştırmaktadır. Bulmacaların kolay bir şekilde çözülmesi kötü niyetli kişilerin ağı ele geçirmesine neden olabilir. Kötü niyetli kişilerden ağı koruyabilmek için bulmacaların zorluk derecesi zamanla arttırılmaktadır.

Günümüzde herhangi bir bilgisayar ile bulmacaları çözmek oldukça zorlaşmıştır. Zorluk derecesinin arttırılması, nonce değerinin bulunması için çok daha fazla deneme

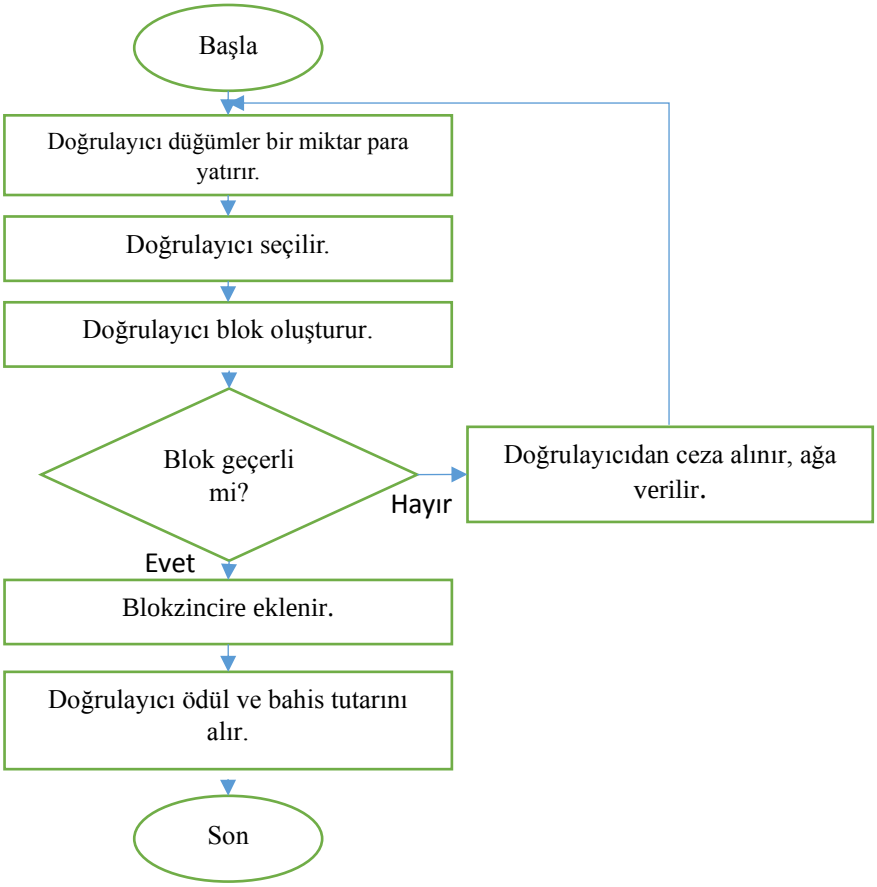
yapılmasını gerektirecektir. Bulmacaları çözebilmek için birden fazla bilgisayardan faydalanılmaktadır. Böylece, bilgisayarlar arasında iş yükü dağıtılabilmekte ve bulmaca daha hızlı bir şekilde çözümlenebilmektedir. Bulmacanın zorlaşması ile beraber kullanılan zaman, enerji ve kaynaklar artmış, israfa neden olmuştur. PoW protokolü en büyük eleştiriyi israfa sebep olmasından alır.

3.2.3.2. Hisse kanıtı (PoS) uzlaşma protokolü

Hisse kanıtı (Proof of Stake - PoS) uzlaşma protokolü fikri, PoW protokolünün yüksek enerji tüketimi sorununu ortadan kaldırmak amacıyla 2011 yılında bitcointalk.org forumunda ortaya çıkmıştır. PoW protokolü melez bir tasarıma sahiptir. İlk kriptopara basımı için PoW protokolü, sonrasında ağ güvenliğinin sağlanmasında PoS protokolü kullanılır [76]. Günümüzde kullanılan hisse kanıtı uzlaşma protokolünde cüzdanında kriptopara bulunduran her düğüm blok oluşturabilmekte ve doğrulayıcı olabilmektedir. Düğümler rastgele seçilmektedir fakat seçilme olasılıkları her düğüm için eşit değildir. Bir düğümün cüzdanında ne kadar çok kriptoparası varsa ve kriptopara yaşı (düğümün elinde bulunan kriptoparayı ne kadar süre harcamadığı) ne kadar yüksek ise bloğu oluşturma olasılığı o kadar artmaktadır. Hisse kanıtı uzlaşma protokolünün temel işleyişini gösteren akış diyagramı Şekil 3.4.'de verilmektedir.

Hisse seneti algoritmasında kullanıcılar kriptoparalarının bir kısmını işlem yapmak için ağa yatırmak zorundadır. Bu para bahis parası olarak adlandırılır ve doğrulama işleminin teminatı olarak tutulur. Cüzdanında bulunan para ve cüzdanında harcamadan tutma süresi bir sonraki bloğu doğrulama olasılığını etkilemektedir. Bu parametreler dikkate alınarak rastgele bir doğrulayıcı düğüm seçilir. Doğrulayıcı, bir önceki bloğa ait hash değeri ile kendi hash değerininden oluşan yeni bir hash değeri üretir. Üretilen hash değeri ile birlikte işlem blok haline dönüştürülür. Üretilen blok ağa sunulur ve ağ kontrol eder. Oluşturulan blok doğru ise doğrulayıcı bahis ve ödül ücretini alır. Eğer bir hileli durum söz konusu ise blok üreticisi cezalandırılır ve bir sonraki ağ doğrulanma hakkını kaybeder.

Hissesi çok olan bir düğüm, ağ için daha az tehlike arz eder. Şöyle ki bir doğrulayıcı eğer ki kötü niyetli biriyse ve ağa saldırmak niyetindeyse cüzdanında büyük bir kriptopara miktarını biriktirmek zorundadır. Bu durum kendi saldırısının en çok kendisine zarar vermesi demektir.



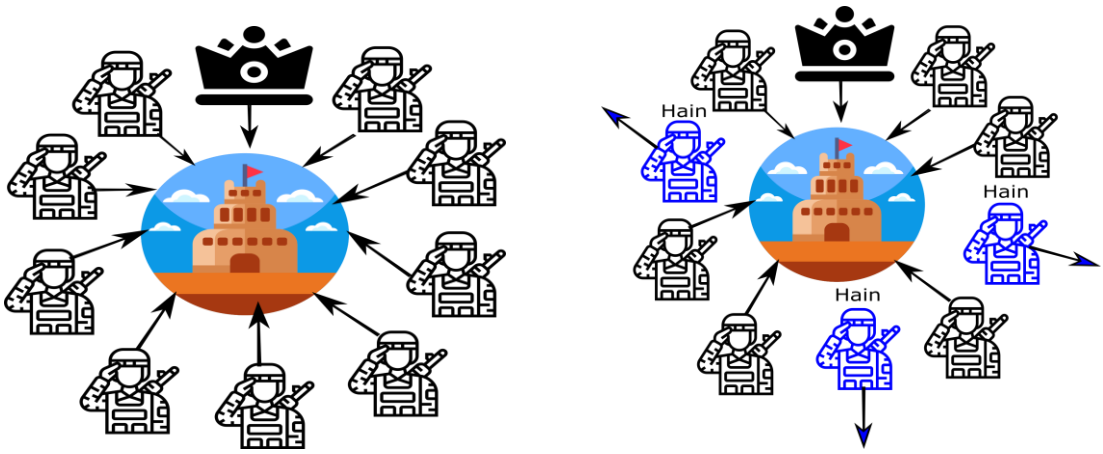
Şekil 3.4. PoS protokolünün çalışma işleyişi.

PoS protokolünde, işlemlerin ucuz ve doğrulama hızının yüksek olması çatallaşma probleminin yaşanmasını kolaylaştırmaktadır. Hangi çatalın kazandığına bakmadan her iki çataldaki blokların onaylanması, seçilen düğümlerin aynı anda birden fazla blok oluşturması ve bloklardan farklı işlem ücreti talep etmesi ve bunlara bağlı olarak çift harcama problemlerinin yaşanması PoS protokollerinin en önemli sorunu olarak adlandırılmaktadır [74]. Bu probleme karşı farklı çözümler üretilmiştir. Örneğin; Casper PoS protokolünde çatallaşma durumunda her iki çatalı onaylayan düğümler ceza almaktadır.

Farklı amaçlar için hazırlanan birçok BC uygulaması vardır. Bu uygulamaların kullandığı farklı uzlaşma algoritmaları geliştirilmiştir. PoS protokolünü temel alan ve eksiklerini gidermek amacıyla geliştirilen uzlaşma protokolleride (Temsili hisse kanıtı-DPoS, kiralık hisse kanıtı-LPoS vb.) vardır. PoS protokolü PoW uzlaşma protokolüne göre daha az kaynak harcar, yüksek doğrulama hızına sahiptir, daha düşük komisyon ücretine ve başlangıç yatırımına ihtiyaç duyar. Bu özellikleriyle tercih edilen bir uzlaşma protokolüdür.

3.2.3.3. Bizans hata toleransı (BHT) modeli

Dağıtık sistemlerde güvenli bir şekilde haberleşmek, ortak bir karara varmak kolay değildir. Düğümlerin başarısız olması ya da dürüst davranmaması durumunda düğümlerin ortak bir karara varması, sistemin koordineli bir şekilde çalışması mümkün değildir. Bu problem bizans generalleri problem kavramının temelini oluşturmaktadır (Şekil 3.5.).



Zafere götüren koordineli saldırı. Yenilgiye yol açan koordinasyonsuz saldırı.

Şekil 3.5. Bizans generalleri problemi.

Bizans generalleri problemi, bir grup Bizans generalinin bir sonraki adımlarının ne olacağı ile ilgili fikir birliğine varmaya çalışırken karşılaştıkları sorunları gösterir. Bizans generalleri farklı bölükler halinde düşman kentinin dışında kenti çevreleyecek şekilde konuşlanmaktadır. Her generalin kendi bölüğü vardır ve bölükler kendi generallerinin emirlerini uygular. Generaller birbirleriyle haberciler aracılığıyla

haberleşebilir. Generaller, düşmanı gözlemledikten sonra ortak bir karara varmaları gerekir. Verilen karar değiştirilemezdir. Burada önemli olan şey çoğunluğun fikir birliğine ulaşmasıdır.

Bizans generalleri birbirleriyle iletişim kurarken problem yaşayabilirler. İletişim, haberciler aracılığıyla bir mesaj yoluyla gerçekleşmektedir. Ulaştırılan mesaj gecikebilir, zarar görebilir, kayıp olabilir, değiştirilebilir. Generallerden bazıları kötü niyetli olabilir, sistemi yanıltıcı mesajlar gönderebilir.

Dağıtık bir yapıya sahip olan blokzinciri ağ yapısının iletişimi sırasında bizans generallerinin yaşadıkları problemler yaşanabilir. Bizans generalleri problemini blokzinciri ağ yapısına uyarlayacak olursak generaller düğümleri, haberciler ağdan gönderilen mesajları, sadık generaller sadık düğümleri, hain generaller blokzincirine yanlış bilgi veren düğümleri temsil eder. Blokzinciri ağının başarısızlığının önüne geçebilmek için dağıtılmış ağ içerisinde generallerin çoğunluğunun fikir birliğine varması gerekir.

Bizans hata toleransı (BHT) modeli bizans generalleri probleminin başarısızlık durumlarını belirli bir düzeye kadar tolere eden modelin ismidir. Blokzinciri ağlarında bizans hatalarına karşı %100 tolerans gösterecek bir çözüm yoktur. Katılımcı sayısı BHT'nin yüzdesini arttırmaktadır. Katılımcı sayısı ne kadar çok artarsa ağın kandırılma olasılığı o derece düşmektedir. PoW protokolünde güçlük derecesinin artması da BHT'yı arttırmaktadır.

Bizans generalleri problemi halka açık BC uygulamalarında uzlaşma algoritmalarıyla çözümlenmeye çalışılmıştır. Özel BC uygulamalarında bizans generalleri problemi ile karşılaşma ihtimali halka açık BC uygulamalarına göre daha azdır. Özel BC ağına katılım onay bekler ve herkes katılamaz. Hyperledger Fabric özel BC ağı uygulamalarına örnek verilebilir. Bu uygulamada PoW protokolüne benzer işlevler gerçekleştirilir. Blok oluşturulduktan sonra doğrulama işlemi için blok tüm ağa gönderilir. Farklı olarak bloğun zincire eklenebilmesi için doğrulama yapanların sayısının en az 2/3'ünde aynı hash değerinin görülmesi gerekir. Hain generaller

azınlıkta kaldığı sürece problem yaşanmaz. Ağın çoğunluğu ele geçirildiyse %51 saldırısı meydana gelir ve bu çözüm fayda vermez.

Bizans generalleri problemi için sunulan bir başka çözüm, defter tutucuların kullanılmasıdır. Bu modelle temsili BHT uzlaşma modeli denir. Bu modelde düğümlerin 2 farklı görevi vardır. Defter tutucular ve sıradan düğümler. Defter tutucular, tüm ağın muhasebesini tutar. Defter tutucular oy çokluğu ile seçilir ve ağda bulunan tüm düğümler tarafından bilinirler. Defter tutucular ağı yanıltmaya çalışan kişiler için bir engel oluşturmaktadır.

Farklı BHT modelleri farklı tekniklerle BHT modelinin uygulanmasını sağlamıştır. Amaç bizans generali probleminin oluşumunu dağıtık yapıdaki blokzinciri uygulamalarında engellemektir. Problemin çözümünü kesin olarak sağlayan bir model yoktur fakat büyük oranda engellenebildiği söylenebilir.

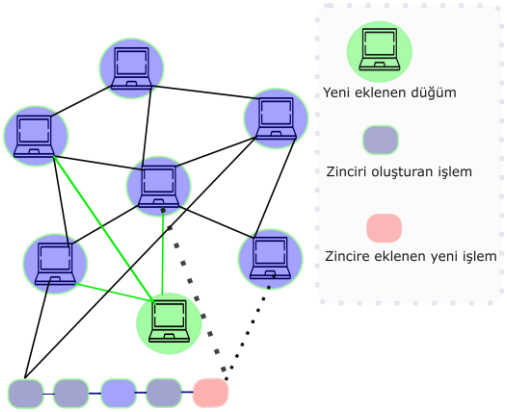
3.2.4. Blokzinciri ağ türleri

Uygulamalar amaçlarına göre farklı güvenlik seviyesine, ağ erişim iznine, hız seviyesine ve ağ yapısına ihtiyaç duyabilirler. Bazı uygulamalar tamamen merkezi olmayan bir ağ yapısına ihtiyaç duyarken, bazı uygulamalarda kısmi merkezileşmeyi kabul edebilir. Bu ihtiyaçları karşılamak için farklı BC teknolojisi türleri mevcuttur:

- Halka açık (public) BC
- Özel BC ağı
- Konsorsiyum BC ağı

Halka açık (public) BC ağı: Halka açık BC ağı tamamen merkezi olmayan bir yapıya sahiptir. Ağda bulunan herkes tüm kayıtlara erişebilir ve uzlaşma sürecine katılabilir (Şekil 3.6.). Buradaki amaç mümkün olduğunca çok kişinin ağa katılması ve kişi sayısının artmasıyla %51 atak saldırıları, bizans generalleri problemi gibi problemlerden ağı korumak ve ağın güvenliğini sağlamaktır.

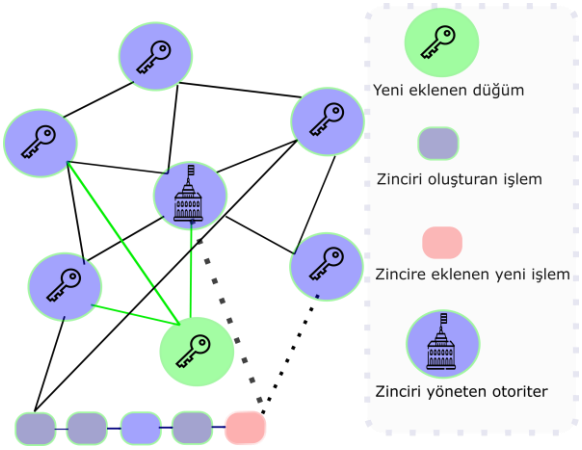
Verilerin herkes tarafından okunabilir olması halka açık BC ağlarınn güvensiz olduğu anlamına gelmez [54]. Verilerin güvenliği kriptografi ile sağlanır. Bitcoin uygulaması halka açık BC ağına örnek olarak verilebilir. Bitcoin bütünüyle izin gerektirmeyen halka açık bir BC uygulamasıdır. Bitcoin ağına dâhil olmak için bitcoin uygulamasının indirilmesi yeterlidir. Uygulamayı kullanan herkes ağda bulunan verileri okuyabilir ve uzlaşma sürecine katılabilir.



Şekil 3.6. Halka açık blokzinciri ağ yapısı.

Özel BC ağı: Özel BC ağında blokzinciri ağına katılabilmek için izin gerekir. Ağı yöneten şirket/kişiler tarafından onaylanan düğümler ağa katılabilir. Uzlaşma sürecine ağda bulunan herkes katılamaz. Ağda bulunan verileri herkes okuyamaz. Ağın kuralları ağı yöneten grup tarafından belirlenir. Özel BC ağ yapısı Şekil 3.7.'de temsil edilmektedir.

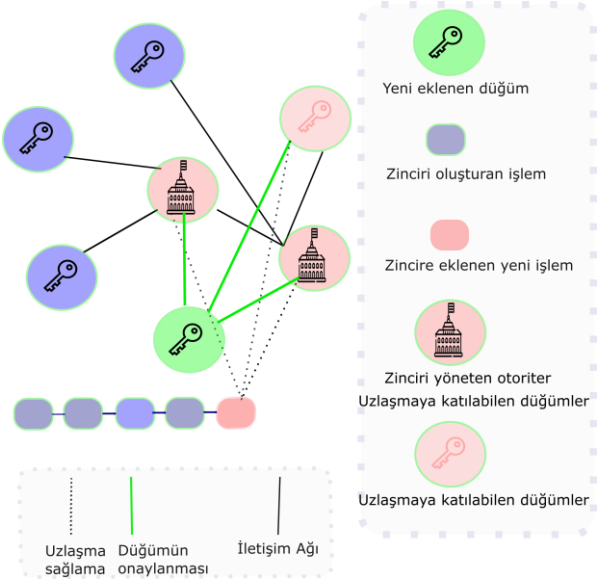
Özel BC ağ yapısı daha çok şirketler tarafından tercih edilmektedir. Halka açık BC ağlarında verilerini bulundurmak istemeyen şirketler kendi BC ağlarını oluşturabilirler. Bu özel BC ağı sadece şirket çalışanlarından oluşabilir ve dışarıya kapalıdır. Özel BC ağında ağın kontrolü ağın yöneticilerinin (şirket yöneticilerinin) sorumluluğundadır. Blokları hangi düğümlerin oluşturacağı, hangi düğümlerin blokları doğrulayacağı ve kimlerin hangi verileri okuyabileceği merkezi otorite tarafından belirlenir. Özel BC ağ yapısında kısmi bir merkezileşme mevcuttur. Tamamen bağımsız bir yapıya sahip değildir. Otoriteye güven söz konusudur.



Şekil 3.7. Özel blokzinciri ağ yapısı.

Konsorsiyum BC ağı (Şekil 3.8.): Konsorsiyum BC ağında birkaç şirket tarafından bir ağ yönetimi söz konusudur, kısmi bir merkezileşme mevcuttur. Ağdaki verilere erişim özel BC ağ yapısında olduğu gibi kısıtlanabilir. Konsorsiyum BC ağında düğümlerin büyük bir çoğunluğu dürüst davranıyorsa, bu ağ türünden faydalanan sistem herhangi bir sorunla karşılaşmaz.

Konsorsiyum BC ağı, aynı sektörde çalışan farklı şirketlerin faaliyet gösterdiği bir ortamda yararlı olacaktır. Ortak kararların alınması, ortak işlemlerin yürütülmesi ve bilgi aktarılmasında konsorsiyum BC ağı ortak bir zemin olacaktır. Sektörle ilgili bilgiler farklı şirketlerin çalışanları ile paylaşılabilecektir.



Şekil 3.8. Konsorsiyum blokzinciri ağ yapısı.

BC türlerinin farklı kıstaslara göre karşılaştırılması Tablo 3.1.'de verilmektedir.

Tablo 3.1. Genel, Özel ve Konsorsiyum blokzincirlerinin karşılaştırılması

	Genel BC	**Konsorsiyum BC**	**Özel BC**
Uzlaşma sağlayıcılar	Bütün madenciler	Seçilmiş düğümler	Bir şirket/kişi
Okuma izinleri	Açık	İzinli/açık	İzinli/açık
Değişmezlik	Neredeyse imkânsızdır	Değiştirilebilir	Değiştirilebilir
Merkeziyetçilik	Yok	Kısmi	Evet
Ağa katılım	Açık, herkes katılabilir	Seçilen kişiler	Seçilen kişiler
Uzlaşma işlemlerine katılım	İzinsiz	İzinli	İzinli
Ağ güvenliği	Evet	Kısmen	Hayır

3.2.5. BC teknolojisinin avantajları ve dezavantajları

Avantajları [75], [77]:

- Blokzincirinde, ağda bulunan verilerin bir kopyası tüm paydaşlar tarafından kopyalanır. Bu bir güvenlik açığı olarak görülebilir. Ancak, bu güvenilirliği sağlamak için bir gerekliliktir. Verilerin bu şekilde saklanması herhangi bir düğüm kaybında veri kaybının yaşanmamasını sağlar.
- Dijital imza ve doğrulamalar sayesinde aracılara ihtiyaç duymaz, ağa ve paydaşların birbirlerine güvenmesi sağlanır. 3. parti aracı olmadan her düğüm birbirlerine bağlanır. Daha şeffaf, güvenilir ve güvenlidir. Aracı ihtiyacının ortadan kalkması aracı ücretinin kalkması demektir.
- BC teknolojisinde değişmezlik garanti edilir ve istenmeyen kişiler tarafından veri değiştirilemez ve silinemez.
- BC teknolojisinde şeffaflık özelliği garanti edilir. Sistemde bulunan kişiler ağda bulunan işlemlerin önceki ve anlık durumları hakkında bilgi edinebilir.
- Bloklar birbirlerine hash değeri ile bağlıdır. Bu özelliği oluşturulma aşamasından son aşamasına kadar işlemlerin izlenebilirliğini mümkün kılar. Tedarik zinciri uygulamalarında BC teknolojisinin bu özelliğinden sıkça faydalanılmaktadır.

- Merkezi bir otorite olmadan çalışabilir. Ağ, ağın kurallarına göre yönetilir. Bu özelliği sayesinde ağ kontrol edilemez, iptal edilemez veya kapatılamaz.
- Akıllı sözleşmeler ile belirli faaliyetler otomatikleştirilebilir.

Dezavantajları:

- PoW gibi uzlaşma protokollerini kullanan BC uygulamaları çok fazla enerji tüketimi gerçekleştirir.
- Madencilik işlemleri pahalı donanımlar gerektirir. Bilgi işlem uygulamalarının büyük bir çoğunluğu hash değerini bulabilmek için yapılan deneme yanılma ile boşa harcanır.
- Blokzincirine yeni bir blok eklemek normal veritabanlarına nazaran yavaştır.
- Blokzincirinde ağdaki her düğümün tüm verilerin bir kopyasını saklaması ve içeriğini okuyabiliyor olması kullanıcı mahremiyetine zarar verebilir. Bunu önlemek adına kriptografiden ve farklı BC ağ türlerinden faydalanılır.
- Akıllı sözleşmeler bir kez oluşturulduktan sonra, veriler ağa kaydedildikten sonra bir daha değiştirilemezler. Bu durum kötü niyetli kişilere karşı ağı savunmasız bırakabilir. İstenmeyen kişiler tarafından yapılan değişikliklerin giderilebilmesi için yeni sözleşmelerin oluşturulması gerekir.

3.3. Akıllı Sözleşmeler

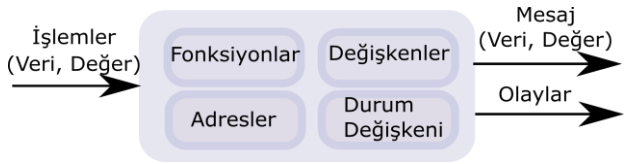
Akıllı sözleşmeler (SC) kavramına ait ilk düşünceler aslında daha eskilere dayanmaktadır. İlk akıllı sözleşme ifadesi 1993 yılında Nick Szabo tarafından yapılmıştır [78]. Nick Szabo, transfer sistemi ile bir digital sözleşmenin kurallarına uyarak nasıl veri/değer girilebileceğini ve kazanılabileceğini anlatmıştır. Bu düşünce blokzinciri teknolojiyle uygulamaya konulabilmiştir. Blokzinciri akıllı sözleşmelerin, akıllı sözleşme de blokzinciri teknolojisine ait uygulamaların gelişmesini sağlamıştır.

Akıllı sözleşmeler merkeziyetçilikten uzak dijital sözleşmeler ile üçüncü tarafa ihtiyaç duymadan işlemlerin gerçekleşmesini sağlayan program parçalarıdır. Akıllı sözleşmeler ile belirtilen kurallar otomatik olarak icra edilirler. Akıllı sözleşmeyi

kullanmak isteyen kullanıcılar sözleşmenin kurallarını kabul etmiş olur. Sözleşme ağa sunulduktan sonra geri alınamaz, silinemez, değiştirilemez. İşlemler bu sayede hızlı ve kolay bir şekilde yürütülür. İnsan müdahalesi olmadan yapılması ağın daha güvenilir olmasını sağlar. Akıllı sözleşmelerin temel özelliklerinden bazıları şunlardır:

- Akıllı sözleşmeler blokzinciri ağı kullanarak uygulanabilmiştir.
- Akıllı sözleşmeler makine tarafından okunabilen yazılım parçalarıdır.
- İş süreçlerini otomatik gerçekleştirir. Bu süreci hızlandırır. Olası hataları minimize eder.
- Merkezi otoriteye ihtiyaç duymaz. Dağıtık bir yapıda icra edilirler.
- Akıllı sözleşmeler az miktarda da olsa ücretlidir.
- Akıllı sözleşmeler aracılığıyla aracı kurumlar ortadan kaldırıldığından giderlerin en aza inmesi sağlanır.

Akıllı sözleşmelerin yapısı Şekil 3.9.'da gösterilmektedir. Şekil 3.9.'da görüldüğü gibi SC'lar değişkenlerden ve çalıştırılabilir fonksiyonlardan meydana gelir. Her SC, BC ağı üzerinde eşsiz bir adresle tanımlanır. İşlemler ihtiyaç duyduğunda SC işlevlerini tetikler ve SC fonksiyonuna ait komutlar icra edilir. SC fonksiyonunun ihtiyaç duyduğu girdi parametreleri işlemler (transaction) tarafından sağlanır [79]. Fonksiyon çalıştırıldıktan sonra SC içerisindeki durum değişkenleri güncellenir.



Şekil 3.9. Akıllı sözleşmelerin (SC) yapısı.

Akıllı sözleşmeler yüksek seviyeli programlama dilleriyle (örn; Solidity, Python gibi.) yazılabilir. Derlenen SC kodları BC ağına eşsiz bir adresle yüklenir. BC ağına yüklenen SC'lar, doğrulama işleminin bir parçası olarak ağda bulunan tüm düğümlerde çalıştırılırlar [79]. BC ağında oluşturulan SC'lar sistemde kalıcı bir uygulama örneği olarak kaydedilirler. Aynı SC kodundan oluşturulan her örnek, yeni bir uygulama örneğidir.

BC teknolojisine olan ilginin artmasını sağlayan Bitcoin uygulaması, klasik BC teknolojisini temel alır ve akıllı sözleşmeleri desteklememektedir. Bitcoin sadece değer transferi için geliştirilen bir uygulamadır. Akıllı sözleşme desteği olmaması sebebiyle farklı alanlarda uygulamaların geliştirilebilmesi için uygun değildir.

Akıllı sözleşmelerin geliştirilmesindeki en büyük katkıyı Ethereum uygulaması sağlamıştır. Ethereum akıllı sözleşmeleri destekleyen halka açık BC ağı uygulamasıdır. Ethereum ağı hem halka açık BC ağının sunduğu avantajlardan hem de akıllı sözleşmelerin sağladığı kolaylıklardan ve imkânlardan faydalanır. Ethereum ağı, Bitcoin ağından sonra ikinci en geniş BC ağına sahiptir. Büyük bir ağa sahip olması ağın kötü niyetli kişiler tarafından ele geçirilmesini engeller. Çok sayıda katılımcı ile tüm dünyada kabul gören Ethereum ağının farklı alanlarda kullanılması akıllı sözleşmelerle mümkün hale gelmiştir.

3.4. Blokzinciri Teknolojisi ile Güçlendirilmiş DApp (BC-DApp)

Son zamanlarda popüler olan uygulamaların (Whatsapp, Twitter vb.) verileri, tek bir kuruluştan / kişiden sorumlu sunucularda depolanır. Bu tür uygulamalar merkezi uygulamalardır. Merkezi uygulamaların güvenlik, güvenilir üçüncü taraf ihtiyacı, daha az şeffaflık, tek nokta hatası gibi kısıtları vardır. Bu kısıtlamalar nedeniyle, uygulamaların tamamında veya bazılarında merkezi bir yapıdan kaçınmak gerekebilir.

Merkezi olmayan uygulamalar (DApp), eşler arası (P2P) ağda çalışan dağıtılmış uygulamalardır. Uygulama verileri tek bir sunucuda tutulmaz ve ağdaki birden çok düğümde verilerin kopyaları bulunur. Torrent uygulamaları, P2P ağlarında çalışan DApp'lara örnektir. Büyük dosyalar torrent uygulamasıyla paylaşılır. Bu dosyalara ve arama motorlarına erişim sağlayan birçok web sitesi vardır. Böylece, bir düğüm kaybı ve düğüme erişim kaybı sisteme erişimi engellemez. Dosya paylaşan kişi sayısı arttıkça, dosya daha hızlı indirilir. Paylaşım durduğunda ağ kesilir, önceki adımlar ve paylaşılan dosya kaybolur.

BC teknolojisinin temel mimarisi üzerine kurulan DApp'lar, geleneksel DApp'lardan farklı bazı özelliklere sahiptir:

- BC teknolojisini temel alan DApp'lar bir kripto parayı kullanmalıdır.
- BC teknolojisini temel alan DApp'lar açık kaynak olmalıdır.
- DApp uygulama verileri, merkezi olmayan bir BC'de saklanmalıdır.
- Konsensus algoritmaları (PoW tabanlı algoritma, PoS mutabakat algoritması-Proof of Stake, yetki verilen teminat kanıtı-delegated Proof of Stake) kullanılmalıdır.

Bu özellikler sayesinde BC destekli DApp'lar daha güvenilirdir. BC destekli DApp'larda veriler değiştirilemez veya silinemezler. Torrent gibi DApp'larda geçmiş tutulmaz, veri kaybı oluşur, olay akışı depolanamaz.

Blokzinciri destekli DApp'lar, herhangi bir aracı olmadan sistemde yer alan kurallar çerçevesinde yürütülür. Kurallar sistemi oluşturanlar tarafından belirlenmiştir ve değiştirilemez. DApp ağına katılan kişiler ağın kurallarını kabul eder. Kurallar temel olacak şekilde ağ üzerinde yapılacak her işlem için ağa katılan kişilerin uzlaşmasına ihtiyaç vardır. Uzlaşılan karar ağa kaydedilir ve ağdan silinemez. Bu özellikleri BC destekli DApp uygulamalarını daha adil, güvenilir ve güvenli yapar.

Blokzinciri destekli DApp'lar 3 farklı açıdan hayata kazandırılmıştır: Sadece kriptopara transferi yapan kişiselleştirilmiş blokzincirine sahip DApp'lar (Bitcoin), kriptoparayı dışarıdan bilgilerle harmanlayan DApp'lar (Omni protokolü) ve akıllı sözleşmelerle farklı amaçlara hizmet edebilen DApp'lar (Programlanabilen daha geniş ölçekli uygulamalar-Ethereum DApp). Tüm bu DApp uygulamaları, blokzincirinin sunduğu avantajlara sahiptir.

Akıllı sözleşme ile DApp uygulamasına ait kurallar ve fonksiyonlar daha yetenekli hale gelmiştir. Böylece, blokzinciri üzerinde farklı DApp uygulamalarının geliştirilebilmesine olanak sağlanmıştır. Sağlık, oyun (Blokchain Cuties), eğitim, savunma sanayi, tedarik zinciri gibi birçok alanda BC destekli DApp'lar

geliştirilmektedir. Bu uygulamaları güvenli bir şekilde kullanabilmek için güvenli bir kriptopara cüzdanına sahip olmanız yeterlidir. DApp uygulamalarının gelişimini destekleyen blokzinciri platformlarına Ethereum, GoChain, EOS, IOST, STEEM örnek verilebilir.

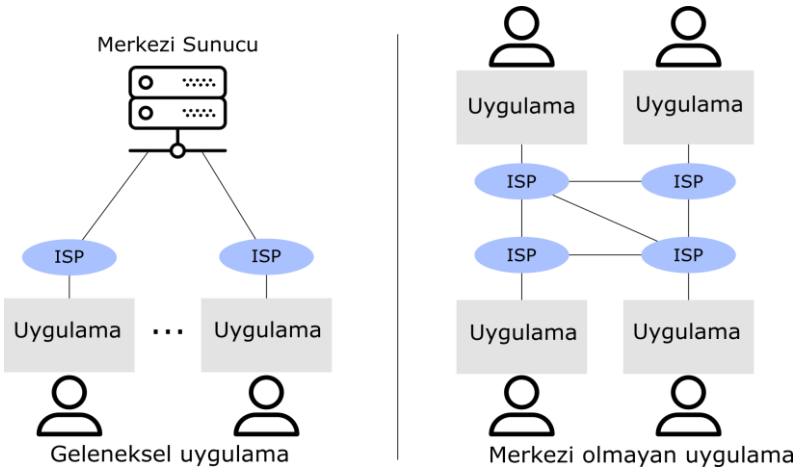
Ethereum platformlarında DApp'lar akıllı sözleşmeler ile icra edilir. Akıllı sözleşmeler ile oluşturulan DApp'larda, kayıt herhangi bir kişisel bilgiye ihtiyaç duyulmadan özel bir adresle, cüzdanla gerçekleştirilir. BC destekli bir DApp geliştirebilmek için BC ağı ile iletişim kurabilen herhangi bir yazılım dili kullanmak mümkündür. Bu çalışmada DApp uygulamasına ait kurallar ve uygulamanın çekirdek yapısı akıllı sözleşmelerin desteklediği solidity programlama dili kullanılarak yazılmıştır. DApp uygulamasının BC ağı dışındaki modülleri dotnet core çatısı üzerinde C# dili kullanılarak geliştirilmiştir.

3.5. Bulut İmalatta BC-DApp

Bulut imalat uygulamaları ve mimari yapıları endüstri ve akademik çevrenin farklı bakış açılarına, geçmiş deneyimlerine ve farklı ihtiyaçlara yönelik çözümlere göre şekillenir. Bulut üretim uygulamalarındaki farklılıklara rağmen, uygulamaların ortak özellikleri vardır. Bir bulut üretim uygulaması merkezi bir yapıya sahiptir. Bulut imalat platformlarında merkezi yapıya sahip olmasından kaynaklanan birçok problem vardır.

Bulut üretim platformlarını uygulamaya dökebilmek için öncelikle sistemin tasarlanması ve kurulması gerekir. Bu tür sistemlerin kurulumu ve bakımı maliyetlidir. Bu maliyetin kullanıcılar tarafından bir şekilde karşılanması gerekir. Kullanıcılar uygulamadan hizmet alabilmek için bu maliyeti ödemek zorunda bırakılır. Bulut imalat platformları internet üzerinde çalışan uygulamalar olduğundan, uygulamayı bir kez bilgisayara indirerek uygulamanın işlevlerinden yararlanmak mümkün değildir. DApp uygulamalarında ise kullanıcılar bir merkeze bağlı değildir. Uygulama tarafından sunulan fonksiyonlar herhangi bir noktadan kullanıcılar tarafından çalıştırılabilirler. Bulut üretim sistemleri tarafından sunulan işlevler BC destekli bir

DApp uygulaması ile kullanıcılara sunulursa, kullanıcılar merkezi bir noktadan bağımsız bir şekilde bu işlevleri özgürce kullanabilirler. BC destekli DApp'lar ile merkezi sunucu ihtiyacı ortadan kalkacak (Şekil 3.10.) ve kullanıcılar merkezi uygulamanın işletimi ve bakımı için gerekli olan ücretten sorumlu tutulmayacaktır.



Şekil 3.10. Geleneksel ve merkezi olmayan uygulama.

Merkezi uygulamada, uygulamaya erişim tek bir fiziksel noktadan gerçekleşir. Fiziksel sunucunun zarar görmesi, uygulamanın son bulması veya uygulamaya yapılan saldırılar gibi sebepler nedeniyle uygulamaya erişilemeyebilir. Uygulamanın bir süreliğine veya tamamen hizmet verememesi ciddi kayıplara neden olur. Bulut üretim sistemlerindeki başarısızlık hem hizmet sağlayıcı hem de alıcı taraf için maddi kayıplara neden olur. Kesintisiz bir hizmet için, platformun yüksek kaliteli bir ağ bağlantısına sahip olması ve uygulamanın çalıştığı fiziksel makinelerin sorunsuz ve güvenli olması gerekir [80]. BC destekli merkezi olmayan bulut imalat uygulamalarında hizmetin sürekliliği tek bir noktaya bağlı olmayacaktır. BC ağının herhangi bir düğümünde bir sorun olması durumunda, başka bir düğüme bağlanılarak hizmetin sürekliliği sağlanabilecektir. BC destekli merkezi olmayan bulut imalat uygulamaları herhangi bir noktadan sürekli olarak çalıştırılabilir ve taraflar arasında ihtiyaç duyulan işlevsellik sağlanabilir.

Bulut imalat sistemlerinde aynı uygulama birçok kullanıcıya hizmet verir. Uygulamadan faydalanabilmek için uygulamaya üye olunması gerekir. Kullanıcının uygulamaya oturum açmasına izin vermek için bir kullanıcı kimlik doğrulama sistemi

kullanılır. Bu sistemler genellikle kullanıcı adı ve şifre bilgilerine dayanır. Uygulamada kullanıcı verileri ortak bir veritabanında saklanır. Bu güvenlik zaafiyeti oluşturur. Uygulamaya giriş yapan kullanıcı, kendisine verilen yetki ile uygulamada kendisiyle ilgili ekranlarda işlemler yapar. Oturum açma sisteminin güvenliğini artırmak için 2FA yöntemi (iki faktörlü kimlik doğrulama) kullanan uygulamalar da vardır. Ancak, bu güvenlik önlemleri kullanıcıları yalnızca dış tehditlerden korumak için tasarlanmıştır. Uygulamayı yöneten veya geliştirenlerin tehditlerine karşı herhangi bir güvenlik önlemi sunmaz. Uygulamanın veritabanına erişimi olan herkes sistem için potansiyel bir tehdittir. BC destekli bulut imalat uygulamasında kullanıcı uygulamayı kendi bilgisayarında çalıştırarak, kullanıcı adı ve şifre gibi bilgilerini uzak bir sunucu ile paylaşmadan cüzdanının gizli anahtarını kullanarak tüm işlemlerini gerçekleştirebilir. Kullanıcı BC ağındaki geniş bir kitle tarafından doğrulanan verileri depolar.

Geleneksel bulut imalat uygulamalarında uygulamanın hizmet verememesi durumunda kullanıcı verileri uygulama içerisinde kilitli kalır. Kullanıcı kendi özel verilerine dahi erişemez duruma gelir. Verilerini başka bir platformda kullanmak üzere merkezi uygulamadan dışarıya çıkaramaz. Daha da kötüsü uygulama bu verileri belirli bir süre sonra tamamen silebilir veya bu verileri satabilir. Örneğin; Linkup isimli çevrimiçi depolama şirketi, müşteri verilerinin %45'inden fazlasına erişimini kaybettikten sonra 2008 yılında kapatılmıştır [81]. Linkup şirketi müşteri verilerini depolamak için Nirvanix şirketine güveniyordu. 20.000 Linkup kullanıcı hizmet alamadı. 20.000 Linkup kullanıcısının verileri kayboldu. Kullanıcı verilerini platformda kilitli tutulması uygulama sahipleri için bir kullanıcı garantisidir. Verilerin kilitli kalması kullanıcıları artan fiyatlar, güvenilirlik ve erişilebilirlik sorunlarına karşı savunmasız bırakır.

Uygulamanın kullanıcı sayısını artırmak uygulama sahibi için büyük önem taşır. Uygulamanın kullanıcı sayısının artması uygulama sahibi için büyük önem taşır. Bulut imalat uygulamalarında kullanıcı sayısı arttıkça ürün çeşitliliği ve alışverişi de artar. Alışveriş iki şekilde gerçekleşir. Hizmet veya ürün alıcı, ücreti uygulamaya veya doğrudan ürünün sahibine gönderir. Alışveriş sürecinde, uygulama bir aracı olarak

görev alırsa; ücret, aracı olarak görev alan uygulamanın banka hesabında korunur. Ücret, ürünü alan kişinin onayıyla ürünü satan kişiye gönderilir. Aracı uygulama bütün bu işlemler için komisyon ücreti alır.

Bulut imalat uygulamalarında kullanıcıların güvenilir bir aracıya ihtiyacı vardır ve sürecin işleyişi sistem tarafından değil kurum veya kişiler tarafından kontrol edilir. Aracı kurum/kişiler, taraflar arasındaki olası bir anlaşmazlıkta herhangi bir tarafı seçebilir ve tarafsız olmayabilir. BC ağında, akıllı sözleşmeler ile taraflar arasındaki anlaşmalar garanti altına alınır. Taraflar akıllı sözleşmelerde yer alan kurallara tam olarak uymak zorundadır. Böylece anlaşmaların güvenliği garanti altına alınmış olur. BC destekli DApp ile yapılan anlaşmalar tüm ağ tarafından doğrulanır. Doğrulanan anlaşmalar, BC destekli DApp uygulamalarının sürecine dâhil edilir. Hiçbir aracıya ihtiyaç yoktur. Örneğin; ilgili koşulların karşılanması durumunda ücretin ödenmesi isteniyorsa sözleşmede yer alan koşullar yerine getirilene kadar sistemdeki ücret aktarılmaz. Ücret transferi için bir bankaya veya aracı şirkete gerek yoktur. Paranın güvenliği ücretsiz olarak sağlanmaktadır.

BÖLÜM 4. GELİŞTİRİLEN CBDM MODELİ, BULUT İMALAT PLATFORMUNDA DAPP UYGULAMASI (DCMAPP) VE DCMAPP SERVİS PLANLAMA MODELİ

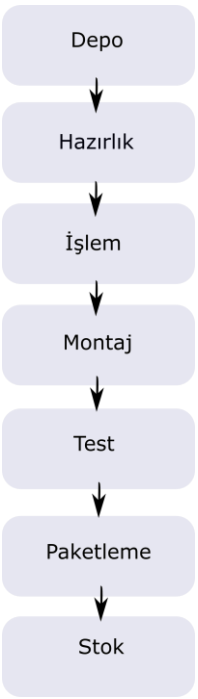
Bulut tabanlı tasarım ve imalat (CBDM) kavramına ait tanımlamalar, Endüstri 4.0 çağına ait araç ve teknolojiler, geliştirilen CBDM modelleri incelenerek bir bulut imalat modeli geliştirilmiş, Bölüm 4.1'de geliştirilen bulut imalat modeli ayrıntılı olarak anlatılmaktadır. CBDM modellerinin merkezi olmasından kaynaklanan problemlerine çözüm olarak bulut imalat ortamında gerçekleştirilen temel üretim döngüsü, merkezi olmayan ve dağıtık bir uygulama (DApp) üzerinde gerçekleştirilmiştir. Geliştirilen merkezi olmayan ve dağıtık uygulamaya DCMApp (Decentralized Cloud Manufacturing Application) ismi verilmiştir. DCMApp, Bölüm 4.2'de ayrıntılı olarak anlatılmaktadır. Bulut imalat uygulamaları için hayati önem arz eden, imalat görevlerini uygulamak için kritik öneme sahip servis arama işlemi için DCMApp üzerinde AHP ve genetik algoritmayı temel alan yeni bir model geliştirilmiştir. Geliştirilen yeni hibrit modelin matematiksel yapısı "4.3. DCMApp üzerinde geliştirilen kaynak planlama modeli" başlığı altında ayrıntılanmıştır.

4.1. Geliştirilen Bulut İmalat Modeli

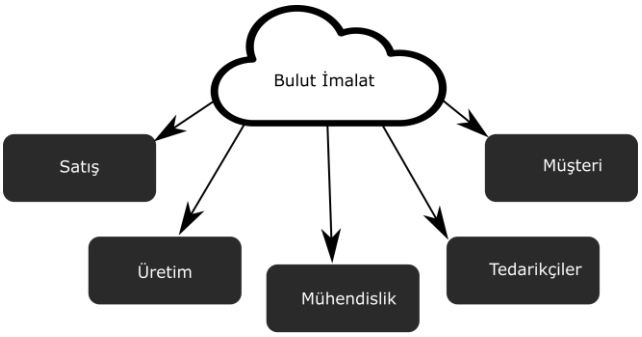
Geleneksel imalat sistemlerinde sistemin limitleri, sistemi oluşturan parçalar tarafından belirlenir. Kurumsal bir şirketin seçenekleri yalnızca ağında bulunan iş ortakları ile sınırlıdır. Sistemin çevresinde bulunan iş ortakları ve şirketin kaynakları sistemin gelişiminde en büyük engel olarak karşımıza çıkmaktadır. Özellikle KOBİ'lerin gelişimi için, bu durum maliyetleri arttıracağından gelişim bütçe ile sınırlı kalacaktır.

Şekil 4.1.'de bir ürünün tipik imalat aşamaları gösterilmektedir. Bu dizilimin her bir aşamasında çeşitli kaynaklar kullanılmaktadır. Geleneksel imalat modellerinde bu iş

ortaklarının veya kaynakların tedarik aşamaları, bakımı, işletilmesi ve diğer tüm aşamaları işletmeye aittir. Geleneksel imalat yöntemlerinin oluşturduğu bu problemleri aşabilmek için sorumlulukları dağıtarak bir bulut ağ yapısı ile her bir parçayı yönetmenin verimliliği arttıracağı düşünülmektedir. Bu yapı Şekil 4.2.'de müşteri, tedarikçi, üretim, mühendislik, satış gibi parçalardan oluşan bir bulut imalat modeli olarak gösterilmektedir. Burada imalat kaynakları paylaşılan bir havuz olarak yönetilmektedir.



Şekil 4.1. Geleneksel imalat modelinin aşamaları.

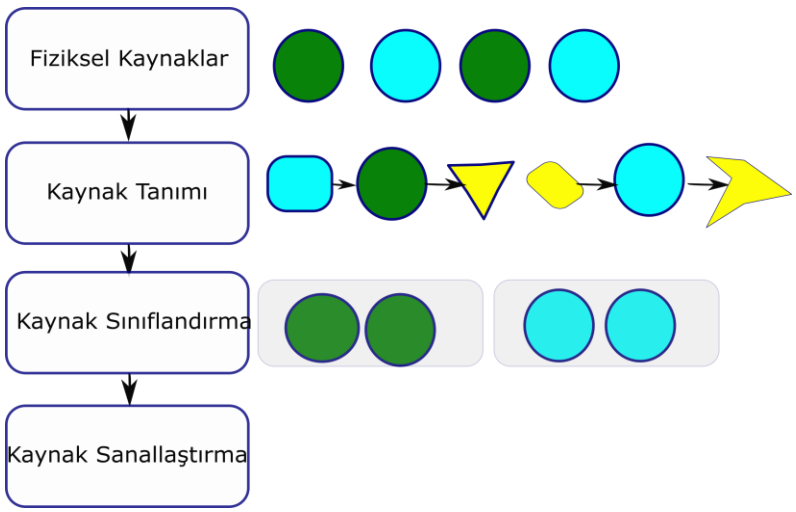


Şekil 4.2. Bulut imalat modelinin aşamaları.

Burada ürünün servis olarak verildiği bir model mevcuttur. Bulut imalatta da kaynakların ve fonksiyonların bir servis olarak verilebilmesi üzerine

odaklanılmaktadır. Ancak bulut imalat iki farklı çeşide ayrılmaktadır. Birincisi bulut bilişimin imalata uygulanması, ikincisi ise imalat aşamalarının bir bulut imalat haline getirilmesidir. Her iki durumda da bulut bilişim kullanılacaktır. Bulut imalatta farklı olarak fonksiyonların sanallaştırılarak bir servis olarak verilmesi en önemli kıstastır. XaaS (servis olarak X) kavramı bu noktada ortaya çıkmaktadır. Bu modelde herhangi bir yatırım yapmadan bir kaynağın veya fonksiyonun kullanılabilmesi esastır.

Bulut imalat modelinde kaynakların sanallaştırılması en önemli adımdır. Sanallaştırma işlemleri bulut bilişimde olduğu gibi hızlı bir şekilde gerçekleşmesi oldukça zordur. Bilişim kaynaklarının sanallaştırılması yazılımlar ile oldukça kolay iken fiziksel makinelerde sanallaştırma oldukça zordur. Öncelikle bu makinelerin veya diğer imalat kaynaklarının bilgi alışverişine uygun hale getirilmesi gerekir. İlgili kaynaklar sistemde tanımlanarak fonksiyonlarına göre ayrılmalıdır. Ancak bu adımdan sonra kaynakların fonksiyonları taklit edilerek veya gerçek zamanlı izlenerek sanallaştırılabilir. Şekil 4.3.'de bir kaynağın sanallaştırılabilmesi için izlenecek adımlar gösterilmiştir.



Şekil 4.3. Kaynak sanallaştırma adımları.

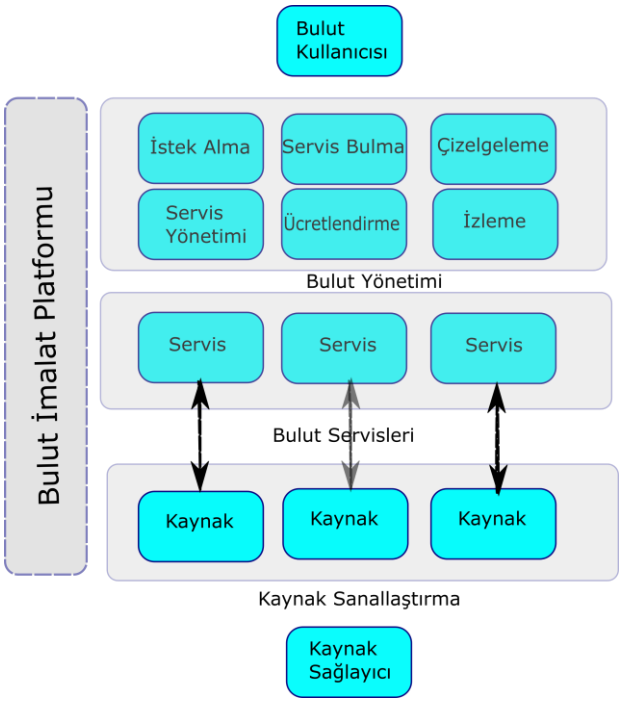
Kaynak sanallaştırması yapıldıktan sonra kaynakların daha etkin bir şekilde kullanılabilmesi için gereken alt yapı hazırlanmış olacaktır. İşletmeler imalat kaynaklarını bulut imalat platformu aracılığı ile paylaşabilir ve kaynaklarının verimliliğini arttırabilir. Bu sayede birer kaynak sağlayıcı konumunda bulut

kullanıcılarına hizmet verebilir. Bulut kullanıcılar kaynaklara yatırım yapmadan sadece kaynağa ihtiyaç duyduklarında hizmet alır. Kısa süreli yoğunluğun yaşanması durumunda bulut kullanıcısı buluttan kaynak miktarını arttırabilir. Kısa süre için yüksek maliyetlere katlanmak zorunda kalmaz. Ancak bulut imalat platformunun hizmet verebilmesi için sadece kaynakların sanallaştırılması yeterli değildir. Sanallaştırılan imalat kaynaklarının bir noktadan yönetilmesi gerekmektedir. Bulut yönetim ile bulut imalat platformu içinde bulunan tüm servisler koordine edilir ve kullanıcıların isteklerine göre en uygun çözümler bulunur.

Bulut yönetim içinde bulunan fonksiyonlar aşağıda açıklanmıştır.

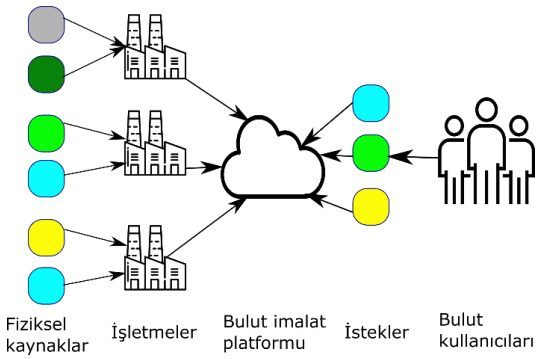
- İstek alma: Bulut kullanıcısından istekleri belirli bir formatta alır ve gerekli dönüşüm işlemlerini gerçekleştirir.
- Servis bulma: Gelen istekler analiz edilir ve isteklerin özellikleri ve ihtiyaç duydukları fonksiyonlar belirlenir. Fonksiyon özelliklerine göre istekler ilgili servislerle eşleştirilir.
- Servis yönetimi: Servisleri bulut platforma ekleme ve silme işlemlerini gerçekleştirir. Dinamik olarak servislerin ölçeklenebilmesi bulut platformun en önemli özelliğidir.
- Çizelgeleme: Bulut kullanıcılara en uygun kaynağın bulunmasını, bulut sağlayıcılara ise üretim planının en uygun şekilde dengelenmesini sağlar.
- Ücretlendirme: Bulut sistemlerin en önemli özelliklerinden biri olan "kullandığın kadar öde" yaklaşımı ile kullanılan hizmetlerin ücretlendirmesi kullanıldığı miktar kadarı ile hesaplanır.

Şekil 4.4.'de gösterilen bulut imalat platformu modelinde bulut kullanıcıları ile bulut sağlayıcıları (kaynak sağlayıcılar) arasında bilgi transferini sağlayan bir yapı görülmektedir. Bulut sağlayıcılarının sisteme fiziksel kaynaklarını kaydetmesi ile süreç başlar, fiziksel kaynaklar kaynak sanallaştırma katmanında sistemde tanımlanır. Sanallaştırılan kaynaklar bulutta kullanıma açılmak üzere servis haline getirilir. Bulut yönetim katmanında ise bu servisler kullanılarak bulut kullanıcılarına hizmet verilir.



Şekil 4.4. Bulut imalat platformu modeli.

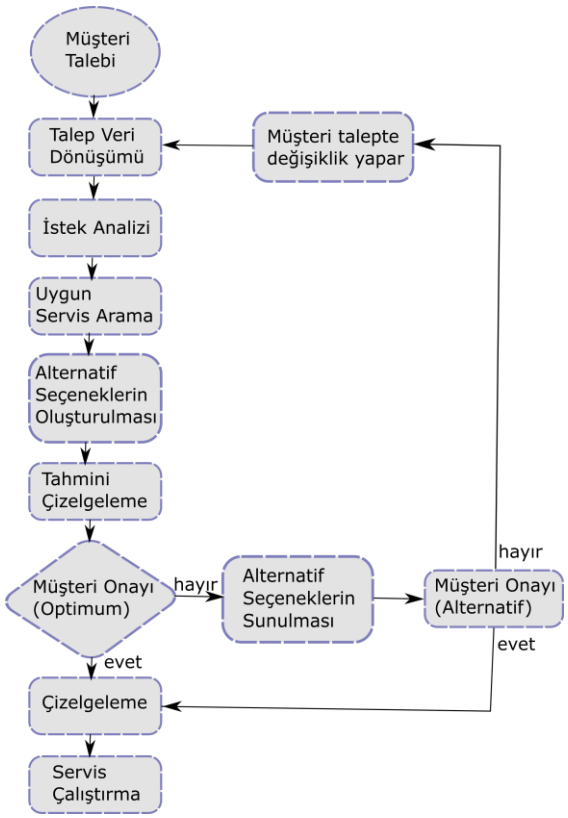
Şekil 4.5.'de gösterilen diyagramda bulut imalat yapısında birden fazla işletmenin tek bir sistem tarafından yönetildiğini göstermektedir. Hem kaynak sağlayıcıların fazla olması hem de kullanıcıların fazla olması planlama işlemini karmaşık hale getirir. Geleneksel üretim planlama ve kontrol sistemlerinden farklı olarak çok fazla sanal kaynağın çok fazla iş için gerçek zamanlı olarak planlanması gerekir. Ayrıca kaynaklar ve kullanıcılar arasındaki tüm süreçlerin bilgi senkronizasyonunun da anlık yapılabilmesi gerekir.



Şekil 4.5. Bulut imalat yapısı.

Şekil 4.6.'da müşteri açısından bir siparişin bulut imalat platformunda nasıl işleneceğine dair akış diyagramı verilmektedir. Bir web tabanlı kullanıcı arayüzü

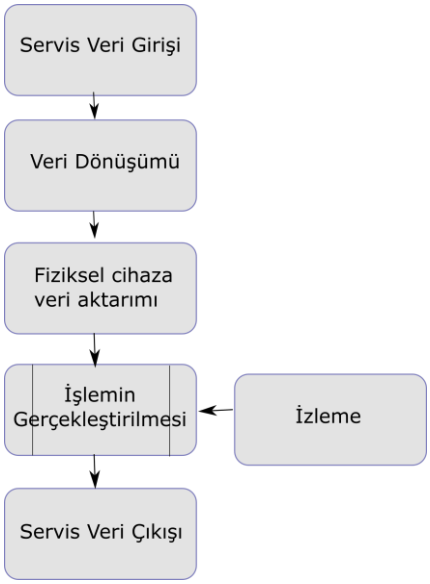
aracılığı ile müşteriden alınan talep, gerekli veri dönüşümlerinden sonra sisteme kaydedilir. Bu veri dönüşümü bir tasarımın istenen formata dönüştürülmesi, veri kontrollerinin yapılması gibi işlemleri içerir. İstek analizi ile talep için hangi tip servislerin gerekli olduğu bilgisi elde edilir. İlgili olabilecek servisler arasından en uygun olabilecek servisler belirlenir. Bu noktada zaman ve maliyet açısından bir optimizasyon işlemi yapılır. Bu işlem birden fazla alternatif çözümü üretir. Kullanılacak servislerin belirlenmesinden sonra bu işlemlerin bitiş zamanları hesaplanır. Tahmini bir teslim tarihi belirlenir ve müşteri onayına sunulur. Müşterinin sunulan teklifi reddetmesi durumunda diğer seçenekler sunulur ve müşterinin istediği seçenek işleme alınır. Müşterinin onay vermesi ile seçilen servisler için çizelgeleme işlemleri gerçekleştirilir. Servis üzerinde ilgili iş gerçekleştirilir ve süreç müşterinin ürünü alması ile tamamlanır.



Şekil 4.6. Bulut imalat platformu akış diyagramı.

Şekil 4.7.'de bulut imalat sisteminde bulunan servislerin veri akışını gösteren diyagram verilmiştir. Servisler fiziksel bir imalat kaynağının sanallaştırılması ile elde

edilen bulut imalat platformunun fiziksel makinelerle iletişim kurduğu arayüzlerdir. Servisler sayesinde sistemin fiziksel makineleri standart bir şekilde yönetebilmesi mümkün olur. Servislere gelen standartlaştırılmış veriler fiziksel makineye uygun olacak şekilde dönüştürülür ve fiziksel makineye aktarılarak işlemin gerçekleştirilmesi sağlanır. Sistem izleme arayüzleri sayesinde makineleri gerçek zamanlı olarak takip eder. İşlemlerin gerçekleştirilmesinden sonra işlem sonucu servisin veri çıkışı ile sisteme aktarılır. Bu çıkış verisi başarılı bir üretimin bilgilerini içerebilirken üretim esnasında karşılaşılan bir arıza veya hata bilgisi de olabilir.

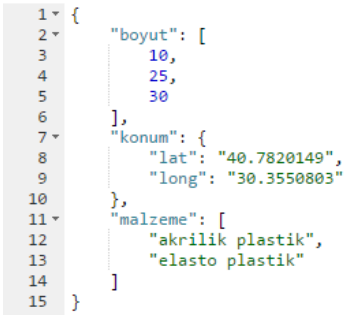


Şekil 4.7. Servis veri akış diyagramı

Servis yönetim modülü (Şekil 4.4.) bulut yönetimi katmanında bulunur. Amacı gelen istekleri uygun servislere yönlendirmektir. Bir servisin sisteme tanımlanabilmesi için sanallaştırma yardımı ile sunulan fonksiyonların detaylıca tanımlanabilmesi gerekir. Servisin hangi tür girdileri kabul ettiği, ne tür kriterlere sahip olduğu ve hangi çıktıları verdiği bilinmelidir. Sistemin bir istek için hangi servislerin uygun olabileceği bilgisini çıkarabilmesi için bu bilgiler zorunludur.

Servise ait birtakım kriterler malzeme tipi, desteklenen boyut, konum, birim maliyet gibi parametreler olabilir. Bu parametreler Şekil 4.8.'de gösterilen örnek veri yapısı ile temsil edilebilir. Her bir servisin kendine özgü kriterleri olabilir. Gelen isteğin

hangi kriterlere ihtiyaç duyduğu ve hangi servislerin bu ihtiyaca cevap verebileceği servis arama algoritması ile bulunur.



Şekil 4.8. Servis Kriterleri Örnek Veri Yapısı.

Uygun servislerin bulunması verimlilik açısından çok önemlidir. Talep edilenden fazla özelliğe sahip olan bir makine kullanıldığında, kullanılmayan özellikler kaynak israfıdır. Tam olarak istenilen özelliklere sahip bir servisin seçilmesi öncelikli olmalıdır. Elbette sadece özellikler açısından servisin seçimi optimum kaynağın bulunması için yeterli değildir. Servislerin maliyetlerinin de ele alınması gerekir.

4.2. Bulut İmalat Platformunda DApp Uygulaması (DCMApp)

Geliştirilen bulut imalat modeli, geleneksel bir imalat döngüsünün bulut platformlarında gerçekleştirilebilmesi için bir rehber niteliğindedir. Fakat bulut imalat platformlarının doğasında mevcut olan merkeziyetçilik problemleri geliştirilen bulut imalat platformunda da yaşanabilir ve yaşanmaktadır. Bu probleme çözüm olarak bu çalışmada, bulut imalat ortamında gerçekleştirilen temel üretim döngüsü, merkezi olmayan ve akıllı sözleşmelerin yardımıyla dağıtılmış olarak gerçekleştirilmektedir. Bu dağıtık uygulama halka açık BC ağı olarak Ethereum ağını kullanır. Bu başlık altında uygulamanın gereksinimleri, uygulamanın gerçekleştirilme adımları, uygulamada akıllı sözleşmenin (SC) kullanımı, uygulama ve uygulamanın değerlendirilmesi anlatılmaktadır.

4.2.1. DCMApp gereksinimleri

Uygulamanın karşılaması gereken birtakım gereksinimleri vardır. Bu gereksinimler şunlardır:

- İşletmeler veya bireyler sahip oldukları makineleri (kaynakları) kaydedebilmelidir.
- Müşteri işi sisteme kaydedebilmelidir.
- Müşteri, işe uygun olabilecek kaynakları görmelidir.
- Müşteri, işin yapılabilmesi için kaynak sağlayıcıya teklif gönderebilmelidir.
- Kaynak sağlayıcı iş tekliflerini görebilmeli ve değerlendirebilmelidir.
- Tüm işlemler bir SC'de gerçekleştirilmelidir.

Bu uygulamada, bulut imalatta olduğu gibi işi yaptırmak isteyen kitleye müşteri, işi yapan kitleye de kaynak sağlayıcı denir. Uygulamada, kaynak sağlayıcı imalatta kullanılacak kaynakların sahibidir. Kaynak sağlayıcı kaynaklarını sisteme yüklerken kaynaklarının tüm özelliklerini doğru bir şekilde tanımlamalıdır. Uygulamada müşteriler iş tanımlarını sisteme yükleyen ve sistemdeki ilgili kaynaklara ihtiyaç duyan ve yapılacak en uygun kaynağı arayan kişilerdir.

Kullanıcılar herhangi bir BC ağında çalışmak için bir BC cüzdanına ihtiyaç duyar. Uygulamada Ethereum ağı kullanıldığından, kullanıcıların bir Ethereum cüzdanı oluşturmaları gerekir. Çeşitli cüzdan uygulamaları ile cüzdan oluşturma işlemi ücretsiz olarak kolayca yapılabilir. Cüzdan, ağda işlem yapmak için özel anahtar ve bir adres ile şifre korumalı bir araçtır. Özel anahtarın çok iyi korunması gerekir. Kullanıcının uygulamayı kullanabilmesi için bir cüzdana sahip olması yeterlidir.

4.2.2. DCMApp akış diyagramı

Geliştirilen bulut imalat uygulamasında kaynak sağlayıcı kaynaklarını, dağıtılmış bulut imalat uygulamasına (DCMApp) kaydeder. Kayıt işlemi sırasında gerekli bilgiler arayüz üzerinden toplanır ve bir SC'ye yazılır. Kaynaktaki tüm işlemler için

kullanılacak, benzersiz bir SC adresi oluşturulur. İşlemi gerçekleştirebilmek için kaynak sahibinin kendi cüzdanını kullanması gerekir. Tüm kullanıcıların bir cüzdan ile işlem yapması zorunludur. Kaynakları yalnızca DCMApp'a kaydetmek yeterli değildir. Kullanıcılar kaydedilmiş kaynakları da görüntüleyebilmelidir. Bunun için kaynaklar için oluşturulan SC adresleri bir listede toplanmalıdır. Bu liste başka bir SC'de veya herhangi bir veritabanında saklanabilir. Liste bir SC'de depolanıyorsa, eklenen her yeni öğe için listenin bulunduğu SC'e bilgilerin eklenmesi gerekir. Bu maliyeti arttırır. Ancak liste dağıtılmış bir şekilde saklanabilecektir. Listenin dağıtık depolanması, listenin kontrolsüz değiştirilmemesini sağlar. Liste merkezi bir veritabanında depolanırsa fiyat önemli bir derece azalacak, fakat güvenilirlik problemleri ortaya çıkacaktır. Bu çalışmada kaynak adresleri listesinin değişimi ciddi bir güvenlik sorununa neden olmayacağından; kaynak adresleri listesinin veritabanında saklamanın daha uygun olacağı düşünülmektedir.

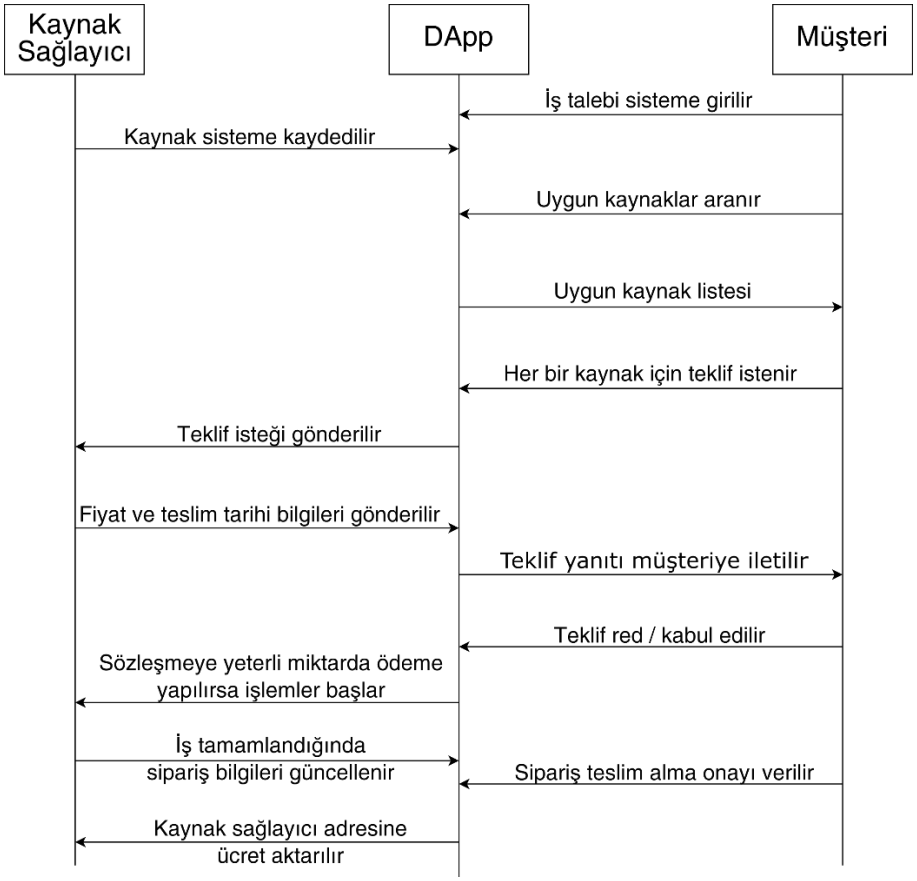
İlgili kaynaklar seçildikten sonra müşteri, işin ayrıntılarını ve son teslim tarihi bilgilerini sisteme girer. Uygulama BC ağında müşteriye özel bir SC oluşturur ve kalan tüm işlemler SC üzerinden devam eder. Müşteri, seçilen kaynaklara uygulama yoluyla teklif sunar. Teklif süreci, SC içindeki ilgili kaynaklar için bir iş akışı başlatır. SC'nin çalışması sırasında, SC'de belirtilen iş akışı tamamen izlenir. Bu işlem SC içinde yeniden işlenir ve tüm ağa yayılır. Dolayısıyla, SC'daki verilerin değiştirilmesi artık mümkün değildir.

Fiyat teklifi istenirken iş teklifi, iş için oluşturulan iş sözleşmesine (JC) yazılır. Teklif, kaynak sağlayıcısına bildirilmelidir. Bu bildirim, kaynağın SC'ına işlenebilir veya kaynağın SC'ında belirtilen iletişim adresine e-posta veya benzeri yöntemlerle gönderilebilir. Bildirimi e-posta veya SC yoluyla göndermek, işlemin işleyişinde herhangi bir değişiklik yapmaz. Daha hızlı bildirim ve daha az maliyet sağlaması sebebiyle e-posta ile göndermek mantıklı bir çözüm olarak görülür. JC içindeki bu bildirime yalnızca kaynak sahibi yanıt verebilir. Kaynak sahibi, teklifi inceledikten sonra iş hakkında bilgi edinir. Kaynak sahibi, söz konusu kaynak için bir iş planı ve teklif oluşturur. Kaynak sahibi hazırlıkları yaptıktan sonra JC 'ye ne zaman teslim

edeceğini ve fiyat teklifi bilgilerini girer. Bu şekilde kaynak sahibi, teklifini SC'lar aracılığıyla BC ağına sunar.

Müşteri, tüm kaynak sağlayıcılarının sunduğu teklifleri inceler. Müşteri tekliflerden birini kabul edebilir, tüm teklifleri reddedebilir veya yeni teklifler isteyebilir. Müşteri kabul ederse, JC tüm teklif sürecini tamamlar. Müşteri tarafından sunulan ücret SC'a kaydedilir. Böylece hem müşteri hem de kaynak sağlayıcı güvence altına alınmıştır. Müşteri, cüzdanındaki Ethereum ile ücreti öder. Müşteri iş onayı verdikten ve JC'a ödemeyi yaptıktan sonra, JC'nin durumu artık "JobAssigned" olarak değişir. Kaynak sahibi işi tamamladığında, siparişi müşteriye gönderir ve teslimat yöntemine göre elde edilen belge numarası JC'ye kaydedilir. İş durumu artık "Sent" olarak JC'ye kaydedilir ve sayaç başlatılır. Bu sayaç otomatik doğrulama mekanizması için kullanılacaktır.

Müşteri siparişi aldıktan sonra JC'daki onaylama metodu çağrılır. Bu metod, JC'daki parayı kaynak sahibinin cüzdanına aktarır ve JC'yi "Completed" olarak işaretler. Müşteri siparişin alındığını teyit etmezse, azami onay süresinin sonunda kaynak sahibi ücreti JC'den alma hakkına sahiptir. Maksimum onay süresi, JC oluşturulurken belirtilen bilgidir. Müşteri ve kaynak sahibi arasındaki tüm işlemler SC ile birlikte Ethereum ağına kaydedilir. SC verileri, Ethereum BC ağında kalıcı bir şekilde depolanır. Uygulamanın olay akış diyagramı Şekil 4.9.'da verilmektedir.



Şekil 4.9. DCMApp uygulamasının olay akış diyagramı.

4.2.3. DCMApp'da geliştirilen akıllı sözleşmeler

SC'lar BC ağına yerleştirildiğinde bir uygulama örneği olarak sistemde kalıcı olarak kaydedilir. Aynı SC kodundan oluşturulan her örnek için yeni bir uygulama örneği oluşturulur. SC kodu, değişkenleri ve fonksiyonları içerir. Bu uygulamada, SC'ları kullanırken tüm işleri tek bir SC'de depolamak veya her bir iş için aynı SC kodundan yeni bir SC örneği oluşturmak üzerine tartışılmıştır. Her iki seçenek de sistemin çalışmasını sağlayacaktır. Ancak; bu iki yöntemi karşılaştırırsak her iş için yeni bir SC örneği oluşturmanın neden daha uygun olacağı anlaşılabilir.

Tüm işlemlerin tek bir SC örneğinde yapılması durumunda;
Avantajlar:

- Yeni bir SC örneği için herhangi bir ücret alınmaz.

Dezavantajlar:

- Bütün işlemler, aynı SC içerisinde depolanır.
- Güncelleme işlemi oldukça zordur.
- Tek bir SC adresi üretildiğinden, aynı adresin kullanılması güvenlik problemlerine sebep olabilir.
- Uzun vadede sürdürülebilirliği zordur.
- SC oluşturan kişi SC'ye erişebilir.

Her iş sözleşmesi için yeni bir SC oluşturulması durumunda;
Avantajlar:

- SC'da yalnızca bir işin bilgisi saklanır.
- Yeni bir iş anlaşmasında güncel bir SC kodu kullanmak mümkündür.
- Her iş için ayrı ayrı SC adresleri oluşturmak, yalnızca ilgili kişilerin bu SC adresini bilmesini sağlayacaktır.
- İş yaptırmak isteyen kişi SC dağıtımını gerçekleştirdiğinden, bilinmeyen bir kişi SC'da herhangi bir bilgi saklayamaz.
- İş tamamlandığında, SC'ın bütün fonksiyonları kapatılır.

Dezavantajlar:

- Her yeni iş için SC tekrar oluşturmak ve dağıtmak maliyeti arttırır.

4.2.4. Geliştirilen DCMApp uygulaması

DCMApp'ın uygulanmasının geliştirilmesinde Solidity ve C# programlama dili kullanılmaktadır. Ethereum SC'lar Solidity programlama dili kullanılarak yazılırlar. Uygulama, dotnet core çatısı altında C# programlama dili kullanılarak geliştirilmiştir. dotnet core çatısı açık kaynak, cross platform özellikleri sayesinde seçilmiştir. Ek olarak, uygulama geliştiricisinin C# programlama dilindeki deneyimi dotnet core çatısının tercih edilme sebeplerinden biridir.

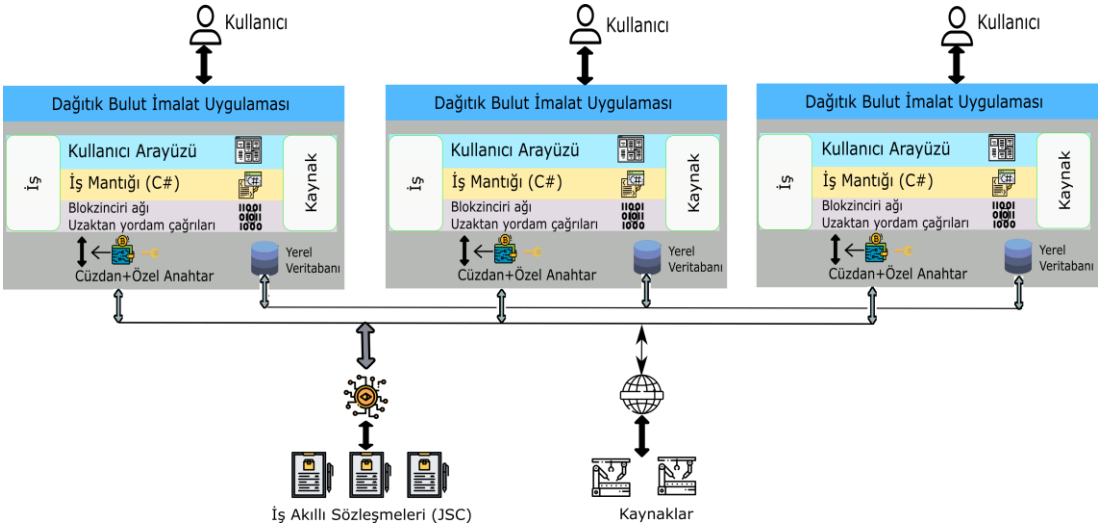
Uygulama, Ethereum ağına bağlanma yeteneği sayesinde dağıtılmış bir yapıya sahiptir. Uygulamanın merkezi bir yapıda olmaması için, uygulamanın tek bir noktaya

bağımlı olmaması gerekir. Merkezi olmayan uygulamalarda aynı problem birden fazla noktada çözülebilmelidir. Tamamen dağıtık ve tek bir noktadan tamamen bağımsız bir uygulama birçok zorluğu beraberinde getirir. Tamamen bağımsız bir uygulama tasarımı için "Veri nerede depolanacak?", "Uygulama kullanıcılara nasıl dağıtılacak?", "Versiyon yönetimi nasıl yapılacak?", "Ortak veriler nasıl saklanacak?" gibi soruların cevaplandırılması gerekir.

Geliştirilen merkezi olmayan uygulama dağıtılmış bir yapıda saklanması gereken verileri SC'larda saklar. Tamamen dağıtılmış bir yapı elde etmek için, tüm bilgilerin tek bir SC'de depolanması çok maliyetli bir çözüm olacaktır. Anlaşma koşullarını tehlikeye atmayacak büyük boyuttaki verileri dağıtılmamış bir çözümle saklamak maliyet probleminin çözümü olarak düşünülebilir. Anlaşmalardan ödün vermeyecek şekilde, uygulama içindeki verilerin yerel bir veritabanında depolanması uygun maliyetli bir çözüm olacaktır. Uygulamanın en önemli prensibi güvenlikten asla ödün vermemektir. Sürecin güvenliğini sağlayan SC'lar, sürecin temel bilgilerini saklar ve çalıştırır. Ancak; işlemin çalışması sırasında, bir dizi ayrıntılı bilgi farklı yöntemlerle yerel veritabanlarında saklanır. Bir iş tasarım dosyası, ağa kaydedilen kaynakların SC adresleri, kaynakların yetenekleri ayrıntılı bilgilere örnek olarak verilebilir. Bu bilgileri merkezi bir sunucuda saklamak bir çözüm olarak görülebilir.

Dağıtık bir uygulamada merkezi bir noktaya bağlı kalarak işlemlerin yürütülmesi istenmeyen bir durumdur. Uygulamanın merkeziyetçi bir yapıya sahip olması uzun vadede uygulamanın hem güvenlik hem de sürekliliği açısından ciddi sorunlara yol açar. Bu noktada çeşitli bulut depolama çözümleri yardımcı olabilir. Bu çözüm, özellikle taraflar arasındaki özel paylaşımlar için çok kullanışlı olacaktır. Alternatif bir çözüm olarak IPFS (gezegenler arası dosya sistemi) sistemleri de kullanılabilir. IPFS sistemleri dağıtık bir şekilde çalışabilen kullanıcıların verilerini depolamak için kullanılan bir sistemdir. IPFS sisteminde yer alan veriler dünya çapında dağıtılmış bir şekilde saklanmaktadır. Bu uygulamada ayrıntılı bilgiler merkezi bir depoda saklanır ve müşterinin yerel veritabanı ile senkronize edilir. Gelecekte bu verilerin IPFS'de depolanması veya diğer bulut depolama çözümlerinde saklanması düşünülmektedir.

Uygulama dotnet core çatısı kullanılarak hazırlandığından, paketlenebilir ve kullanıcıların bilgisayarlarında uygulamanın bir kopyası çalıştırılabilir. Kullanıcılar bu çözümü uygulamak istemezse, işlemleri basitleştirebilmek için uygulamanın bir kopyası merkezi veya birden çok sunucuda çalıştırılabilir. Böylece uygulamanın kullanılabilmesi için sadece web adresini bilmek yeterli olacaktır. Bu uygulamanın dağıtık bir yapıya sahip olmasını engellemez. Tüm işlemler Ethereum ağında da aynı şekilde yapılmaya devam edecektir. Uygulamanın yapısı Şekil 4.10.'da verilmektedir.



Şekil 4.10. DCMApp uygulama diyagramı.

Uygulama, ReST (Temsili Durum Transferi - Representational State Transfer) tabanlı web hizmetleri ile birlikte Ethereum ağı ile iletişim kurar. NEthereum kütüphanesi, Ethereum ağında C# işlemlerinin yapılabilmesini sağlar. Bu kütüphane ile ağa, solidity programlama dili kullanılarak yazılmış SC'lar dağıtılabilir ve SC'larda bulunan fonksiyonlar da çalıştırılabilir. Solidity programlama dili ile hazırlanmış SC'lar NEthereum kütüphanesi ile C# sınıflarına dönüştürülebilir. Bu sınıflar SC'ların bayt kodunu ve ABI (İkili Kod Uyumluluğu - Application Binary Interface)'lerini içerir. Uygulamada NEthereum kütüphanesinden faydalanılmaktadır. Proxy yöntemleri, SC'lerin sahip olduğu tüm işlevler için otomatik olarak oluşturulur.

Uygulama içerisinde iki temel SC vardır. İlk SC, kaynaklar için hazırlanmıştır. Buna kaynak akıllı sözleşme (Resource Smart Contract-RSC) ismi verilmiştir. İkinci SC, kaynaklarda yapılacak işler için hazırlanmıştır. Buna iş akıllı sözleşmesi (Job Smart

Contract-JSC) ismi verilmiştir. RSC değişkenleri Şekil 4.11.'de verilmektedir. RSC'ın kodları Ek A'da verilmiştir.

```
//Kaynak ve akıllı sözleşme sahibinin cüzdan
    adresi
address owner;
//Kaynak sahibi tarafından belirtilen kaynak ismi
string name;
//Kaynak özellikleri ve yetenekleri: boyut,renk,
    malzeme vb.
string properties;
//Kaynağın aktif olup olmadığı veya işin kabul
    edilip edilmediği bilgisi tutulur.
bool isActive;
//Alınan toplam iş sayısı: Bu sayı, kaynağın ne
    kadar tercih edildiği ve güvenilir olduğu ile
    ilgili müşterilere bir fikir verecektir.
uint256 jobCount;
//İş tamamlandıktan sonra bu kaynak için müşteri
    değerlendirmelerinin ortalaması
uint256 customerPoint;
```

Şekil 4.11. Kaynak akıllı sözleşme değişkenleri.

RSC içerisinde yalnızca kaynak tanımlaması yapılır. Bulut üretim ortamına hizmet edecek kaynakların her biri için bir RSC oluşturulur. Ancak tüm bu tanımların adresleri bir noktada toplanmalıdır. Bu problem tüm kaynakların SC adreslerini saklayacak başka bir SC oluşturarak çözülebilir. Bu çözüm sürdürülebilirliği zordur ve maliyetlidir. Bunun yerine, yalnızca adresleri merkezi bir sunucuda depolamak ve tüm kullanıcılara dağıtmak çok daha pratik ve ucuz olacaktır. Merkezi bir sunucuda barındırılan uzak kaynak deposu kullanılarak, kullanıcılar yerel veritabanlarındaki veriyi senkronize edebilecektir.

JSC, her iş için ağda tekrar oluşturulur. Her bir iş için, sisteme iş yükleyen müşteri uygulama aracılığıyla SC kodunu tüm ağa dağıtır. Enum, yapılar (struct), değişkenler ve fonksiyonlar JSC tasarımında mevcuttur. Enum yapısı işin ve teklifin durumunu tanımlar. "JobStatus" enum yapısı işin 6 farklı durumunu tanımlar: "Created", "OffersPending", JobAssigned", "Sent", "Confirmed" ve "Deleted". "OfferStatus" enum yapısı "OfferRequested" ve "OfferResponded" öğelerini tanımlar. Teklif verilmesi ve tekliflerin cevaplanması durumlarını temsil ederler. Bir teklifin sunulduğu "Offer" yapısı Şekil 4.12.'de verilen kod parçasıyla tanımlanmaktadır.

```
struct Offer{
  //Teklif sahibinin cüzdan adresi
  address offerer;
  //Date the offer request was created
  uint requestDate;
  //Teklif isteğinin oluşturulduğu tarih
  uint responseDate;
  //Teklifin durumu
  OfferStatus enumOfferStatus;
  //işin tamamlanmasının taahhüt edildiği tarih
  uint finishDate;
  //Bu iş için fiyat teklifi
  uint price;
  //Teklif için üretilen eşsiz değer
  uint id;
  //Teklifin kaynak adresi
  address resourceAddress;
  //Teklifin kabul edilip edilmediği durum
  bool accept;
}
```

Şekil 4.12. JSC teklif (offer) yapısı.

JSC'de tanımlanmış değişkenler Şekil 4.13.'de tanımlanmaktadır. Bu değişkenlerin bazıları JSC oluşturulurken kurucu (constructor) fonksiyonlar içerisinde kullanılır ve değişkenlerden bazıları diğer fonksiyonlarda kullanılmaktadır. JSC'de tanımlanan değişkenler, JSC'nin tarih bilgisini, iş durumunu, kaydedilen tekliflerin listesini, kabul durumu ve teklif kabul edildikten sonra işlemin tarih bilgisi ve teslimat sonrası bilgileri barındırır. JSC'yi oluşturan kullanıcı, oluşturma sırasında her zaman kurucu fonksiyonu çağırır. Kurucu fonksiyonun ayrıntıları Şekil 4.14.'de belirtilmektedir. Ayrıca kurucu fonksiyonda varsayılan olarak; kurucu yöntemi çağıran, sözleşmenin sahibi olarak atanır. Bununla birlikte o andaki tarih "createDate" değişkenine atanır ve "enumJobStatus" değişkeni "Created" olarak değiştirilir.

```
JobStatus enumJobStatus;
address contractOwner;
uint createDate;
uint dueDate;

uint resourceTypeID;
string propertiesJson;
string designUrl;

address payable acceptedOfferer;
uint acceptDate;
uint confimationMaxTime;
uint confimationExtendTime;
uint confirmDate;

string trackingCode;
uint sentDate;

bytes1 constant maxExtendCount = 0x03;
bytes1 extendCount;

mapping(address => Offer)  public offers;
uint offerCounter;
```

Şekil 4.13. JSC değişkenleri.

```
constructor(
  uint _dueDate,
  uint _resourceTypeID,
  string memory _propertiesJson,
  string memory _designUrl,
  uint _confimationMaxTime,
  uint _confimationExtendTime
  ) public
{
  dueDate = _dueDate;
  resourceTypeID = _resourceTypeID;
  propertiesJson = _propertiesJson;
  designUrl = _designUrl;
  confimationMaxTime = _confimationMaxTime;
  confimationExtendTime =
      _confimationExtendTime;
  contractOwner = msg.sender;
  createDate = now;
  offerCounter = 0;
  enumJobStatus = JobStatus.Created;
}
```

Şekil 4.14. JSC kurucu fonksiyonları.

JSC kurucu fonksiyonu çalıştırıldıktan sonra değişkenlere başangıç değerleri atanır. JSC'nin fonksiyonları artık BC ağında kullanılabilmek için hazırdır. Bu fonksiyonlar

Şekil 4.15.'de gösterilmektedir. Bu fonksiyonların her biri, işletilmeden önce ilgili koşulları sağlayıp sağlamadığı kontrol edilmelidir. Örneğin; bir iş teklifi yalnızca sözleşme sahibinin adresini kullanan kullanıcı tarafından yapılabilir veya teklife yanıt yalnızca cüzdan adreslerinin bulunduğu listeye kayıtlı olan kaynak sahibi tarafından yapılabilir. Bu doğrulama türü, tüm fonksiyonların Solidity programlama dilinin "require (boolean)" şekli ile icra edilir. Örnek; "responseOffer" fonksiyonu Şekil 4.16.'da verilmektedir. Kullanıcılar gerekli koşulları sağlamadıklarında bu fonksiyonları icra edemezler. JSC kodları Ek B'de verilmiştir.

```
function requestOffer(address offerer, address
    resourceAddress)
                public returns (uint)
function responseOffer(bool accept, uint price,
     uint finishDate)
                public returns (bool)
function acceptOffer(address payable offerer)
    payable
                public returns (bool)
function deleteJob() payable
                public returns (bool)
function sendFinishedJob(string memory
    _trackingCode) payable
                public returns (bool)
function confirm() public returns (bool)
function requestConfirmTimeExtend()
                public returns (bytes1, uint)
function offerCount()
                public view returns (uint)
function withdraw()
                public returns (bool)
```

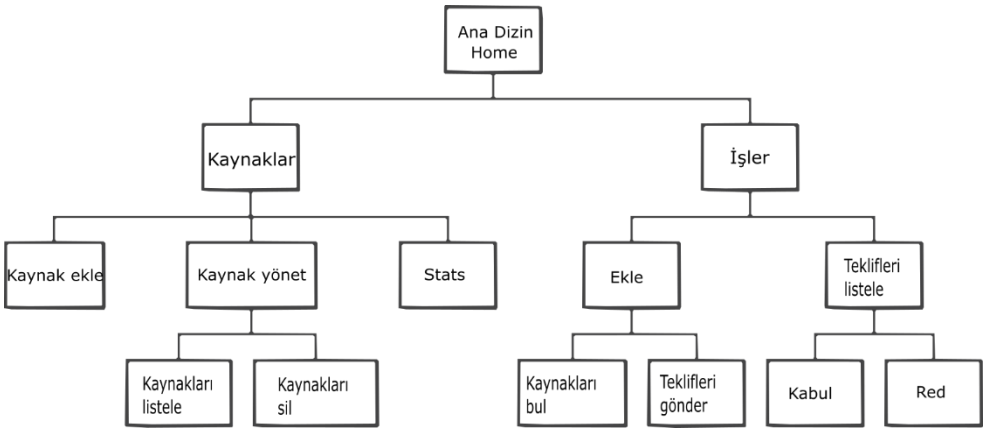
Şekil 4.15. JSC fonksiyonları.

SC'da tanımlanan her işlev Ethereum ağındaki bir bilgisayarda yürütülecektir. SC değişkenlerindeki değişiklikler bloklar halinde işlenecek ve tüm Ethereum ağına dağıtılacaktır. SC'da gerçekleştirilen işlemler güvenilirdir. Ethereum ağının bir parçası olan tüm istemciler, uzaktan SC fonksiyonlarını çağırabilen web servislerini çalıştırabilir. SC fonksiyonları Ethereum ağındaki herhangi bir bilgisayar tarafından uzaktan çağrı ile işletilebilir. Bu SC'ları diğer sistemlerle entegre etmeyi mümkün hale getirir.

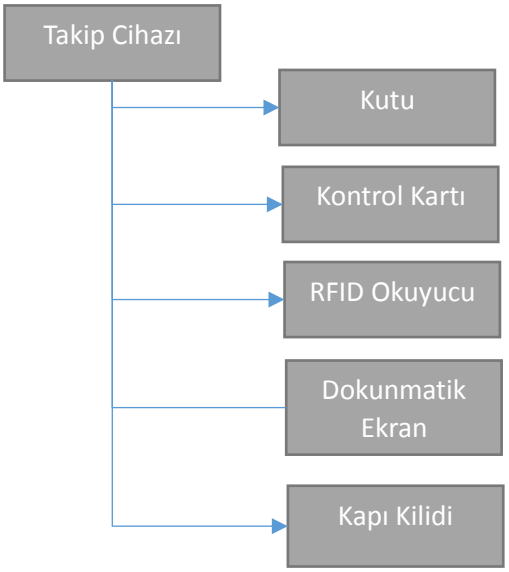
```
function responseOffer(bool accept, uint price,
    uint finishDate) public returns (bool) {

  require(offers[msg.sender].id > 0);
  require(enumJobStatus == JobStatus.
      OffersPending);

  offers[msg.sender].accept = accept;
  offers[msg.sender].price = price;
  offers[msg.sender].finishDate = finishDate;
  offers[msg.sender].responseDate = now;
  offers[msg.sender].enumOfferStatus =
      OfferStatus.OfferResponded;
  return true;
}
```

Şekil 4.16. JSC "responseOffer" fonksiyonu.

Uygulamanın site haritası Şekil 4.17.'de verilmektedir. Uygulamanın iki ana menüsü vardır: Kaynaklar ve iş. Kaynaklar menüsünde, kaynak sağlayıcılarının kaynaklarını yönetebileceği menüler tasarlanmıştır. İşler menüsü altında, bir iş için iş sözleşmeleri oluşturabilen ve yönetebilen ekranlar vardır.



Şekil 4.17. Uygulamanın site haritası.



Şekil 4.18. Örnek Ürün: Takip Cihazı.

DCMApp, Şekil 4.18.'de gösterilen örnek bir ürün üzerinden anlatılmaktadır. Ürün, 5 adet parçadan oluşan bir kartlı takip cihazıdır. Cihazın parçaları ve özellikleri aşağıdaki gibi tanımlanmıştır.

1. Kutu: Takip cihazının dış plastik kutusudur. ABS Plastik malzemeden üretilmeli ve boyutları 150x150mm olmalıdır.
    - type: plastic
    - material: abs
    - size: 150

2. Kontrol Kartı: Cihazın tüm elektronik parçalarını içeren 2 katmanlı bir baskılı devre kartıdır (PCB).
    - type: pcb
    - thickness: 1.6
    - layers: 2

3. RFID Okuyucu: Usb bağlantısını destekleyen, mifare kart tiplerini okuyabilen bir rfid kart okuyucusudur.
    - type: rfid
    - connection: usb

- cardtype: mifare

4. Dokunmatik Ekran: Dokunmatik, 800x480 çözünürlükte görüntü verebilen bir ekrandır.
    - type: screen
    - touch: true
    - resolution: 800x480

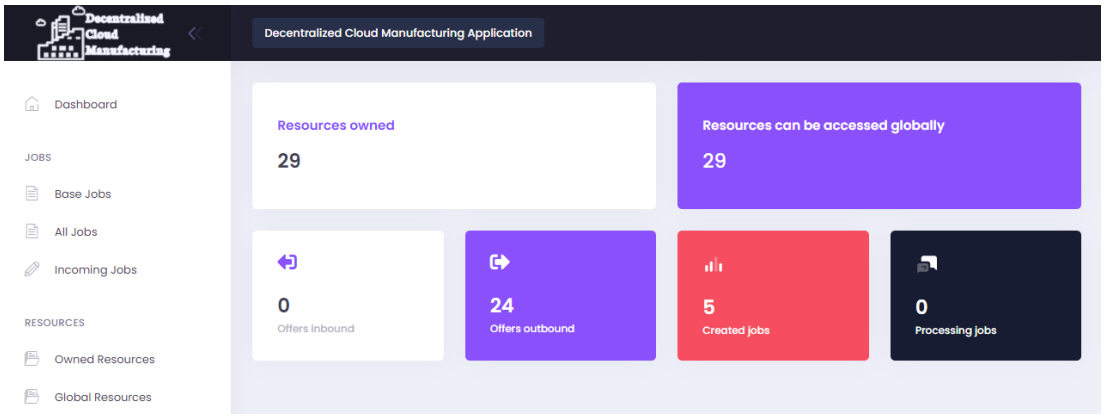
5. Kapı Kilidi: Cihazın kapıları kontrol edebilmesi için kapılara takılacak manyetik kilit.
    - type: lock
    - locktype: magnetic

DCMApp'de 5 ayrı kategoride toplamda 29 adet kaynak tanımlanmıştır. Kaynaklar ve destekledikleri özellik aşağıda verilmiştir.

1. Res Kutu 1; type: plastic, material: abs, size: 500
2. Res Kutu 2; type: plastic, material: abs, size: 200
3. Res Kutu 3; type: plastic, material: abs, size: 400
4. Res Kutu 4; type: plastic, material: abs, size: 500
5. Res Kutu 5; type: plastic, material: abs, size: 100
6. Res Pcb 1; type: pcb, thickness: 1.6, layers: 2
7. Res Pcb 2; type: pcb, thickness: 1.0, layers: 4
8. Res Pcb 3; type: pcb, thickness: 1.6, layers: 2
9. Res Pcb 4; type: pcb, thickness: 1.0, layers: 6
10. Res Pcb 5; type: pcb, thickness: 1.6, layers: 2
11. Res Pcb 5; type: pcb, thickness: 2.0, layers: 1
12. Res Rfid 1; type: rfid, cardtype: mifare, connection: usb
13. Res Rfid 2; type: rfid, cardtype: mifare, connection: usb
14. Res Rfid 3; type: rfid, cardtype: mifare, connection: usb
15. Res Rfid 4; type: rfid, cardtype: mifare, connection: usb
16. Res Rfid 5; type: rfid, cardtype: mifare, connection: usb

17. Res Rfid 6; type: rfid, cardtype: mifare, connection: serial
18. Res Screen 1; type: screen, touch: true, resolution: 800x480
19. Res Screen 2; type: screen, touch: true, resolution: 800x480, resolution: 1024x768
20. Res Screen 3; type: screen, touch: true, resolution: 800x480
21. Res Screen 4; type: screen, touch: true, resolution: 800x480
22. Res Screen 5; type: screen, touch: true, resolution: 800x480
23. Res Screen 6; type: screen, touch: false, resolution: 800x480
24. Res Lock 1; type: lock, locktype: magnetic
25. Res Lock 2; type: lock, locktype: magnetic
26. Res Lock 3; type: lock, locktype: magnetic
27. Res Lock 4; type: lock, locktype: magnetic
28. Res Lock 5; type: lock, locktype: magnetic
29. Res Lock 6; type: lock, locktype: push

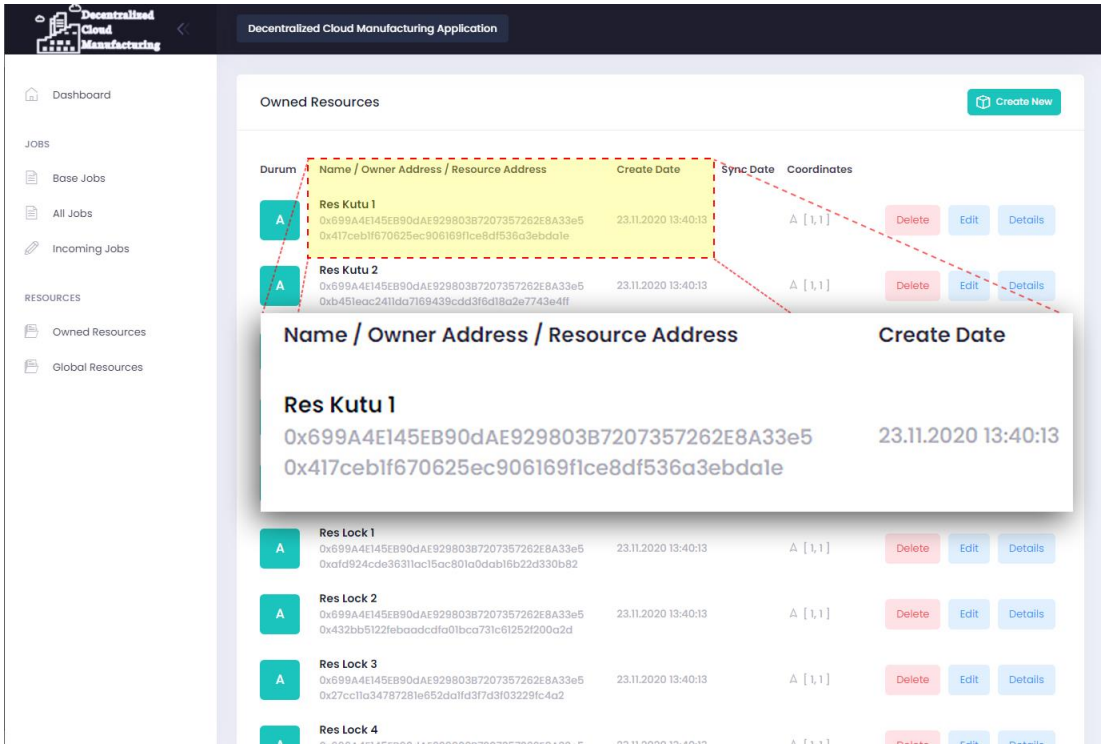
DCMApp'ın kaynaklara sahip veya dışarıya iş yaptırmak isteyen bir kişinin görebileceği örnek ana ekran görüntüsü Şekil 4.19.'da gösterilmektedir. Bu ekranda görüldüğü gibi bu kişi 29 adet kaynağa sahiptir ve kaynaklar herkese açıktır. Bu kaynağa iş teklifi henüz sunulmamıştır. Aynı zamanda bu kullanıcı 5 adet işe sahiptir. Bu ekranda kendi kaynaklarının ve işlerinin özet bilgilerini görüntüler.



Şekil 4.19. DCMApp ana ekran görüntüsü.

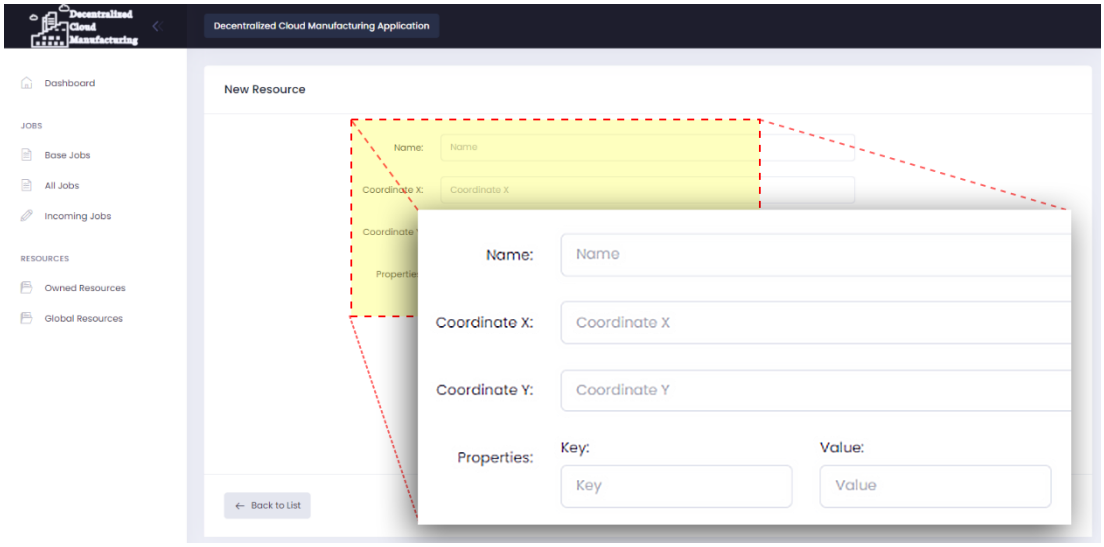
Kaynak sahipleri, kaynaklarını "Resources > Owned Resources" menüsü altından (Şekil 4.20.) görüntüleyebilirler. Bu arayüzü herkes görebilir. Kaynağın oluşturulma

tarihi, kaynak sahibinin ve kaynak konumunun adresleri, kaynağın kullanımın aktif ve pasif durumda olması gibi bilgilere erişim sağlanabilir. Bu arayüzle birlikte kaynaklar düzenlenebilir, sistemden kaynak silinebilir ve kaynağın daha fazla ayrıntılarına erişilebilir.



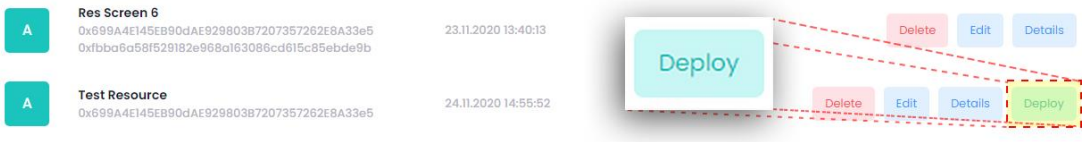
Şekil 4.20. Kaynak listesi.

Yeni bir kaynak oluşturmak için ise kaynak listeleme ekranında bulunan “Create New” tuşu ile yeni kaynak oluşturma ekranına ulaşılır. Yeni kaynak oluşturma ekranı (Şekil 4.21.) ile kaynağın adı, koordinatları ve tüm özellikleri anahtar-değer formatında kaydedilir. Anahtar-değer ikilisi kaynak hakkında tüm detayları barındırır ve istenildiği kadar eklenebilir. Müşteriler işleri için uygun kaynak arama işlemini bu anahtar-değer ikilileri üzerinden yapacaklardır.



Şekil 4.21. Yeni kaynak oluşturma ekranı.

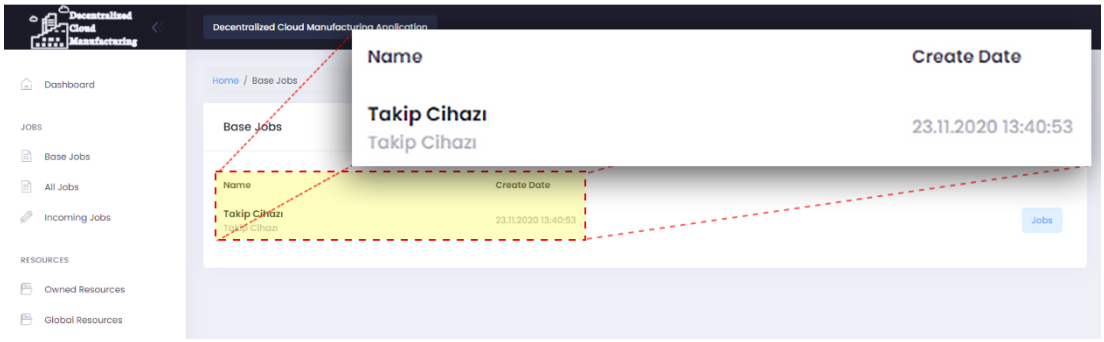
Yeni kaynak kaydedildikten sonra Ethereum ağına yüklenmelidir. Ethereum ağına yüklenirken bu kaynak için bir akıllı sözleşme (SC) oluşturulur. Bu sözleşmenin ağda oluşturulabilmesi için "Deploy" tuşuna basılması gerekir (Şekil 4.22.). Bu işlem sonucunda kullanıcının Ethereum cüzdanından bir miktar Ethereum değeri harcanır. Şekil 4.22.'de görüldüğü üzere kaynağın adının altında iki farklı adres vardır. Birinci adres (0x699A…) kullanıcının Ethereum cüzdanının adresidir. İkinci adres ise bu kaynak için Ethereumda oluşturulan akıllı sözleşmenin adresidir. "Res Screen 6" isimli kaynağın akıllı sözleşme adresi 0xFBBA… olarak görülmektedir. "Test Resource" isimli kaynak için bir akıllı sözleşme oluşturulmadığı ve henüz ağa gönderilmediği için akıllı sözleşme adresi yoktur.



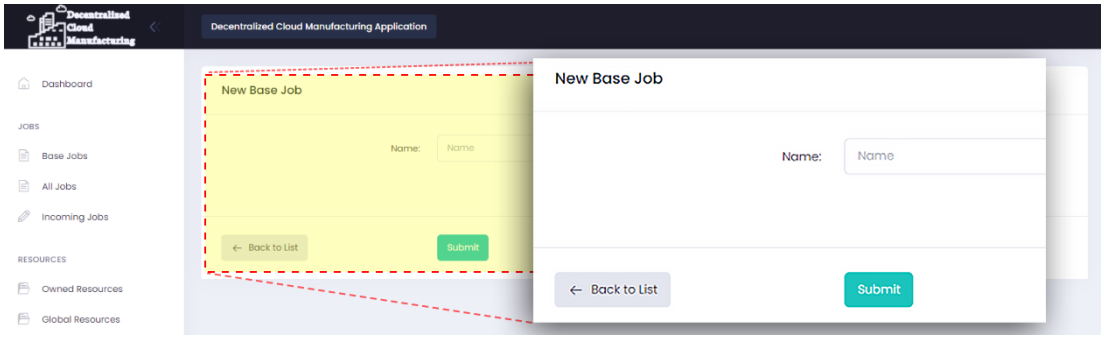
Şekil 4.22. Oluşturulan kaynağın ağa yüklenmesi.

Müşteriler, işlerini kök işler altında gruplayarak sisteme kaydedebilirler. Kök iş, alt işleri bir başlık altında toplar. Böylelikle alt işler üretilmek istenen ürüne göre gruplanabilir. Bu gruplama kullanıcının alt işleri hangi kaynakta yaptıracağına karar verirken kullanılacaktır. Kök işlerin listesi için menüden "Jobs > Base Jobs" sayfası açılır (Şekil 4.23.). Sisteme yeni bir kök iş tanımlamak için ise listedeki "Create New"

tuşu ile tanımlama ekranına girilir (Şekil 4.24.). Yeni kök iş tanımı yapılırken bir isim girilmesi gereklidir.

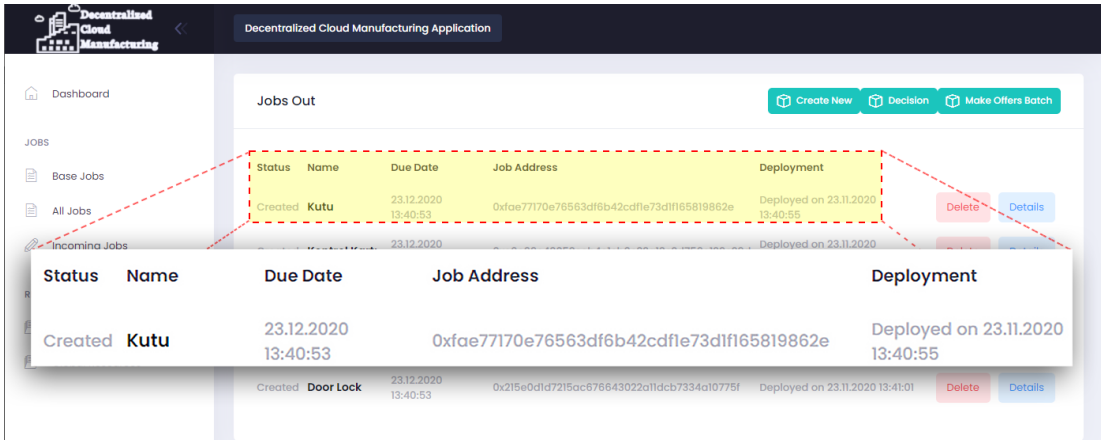


Şekil 4.23. Müşterinin kök iş listesi.



Şekil 4.24. Kök iş tanımlama ekranı.

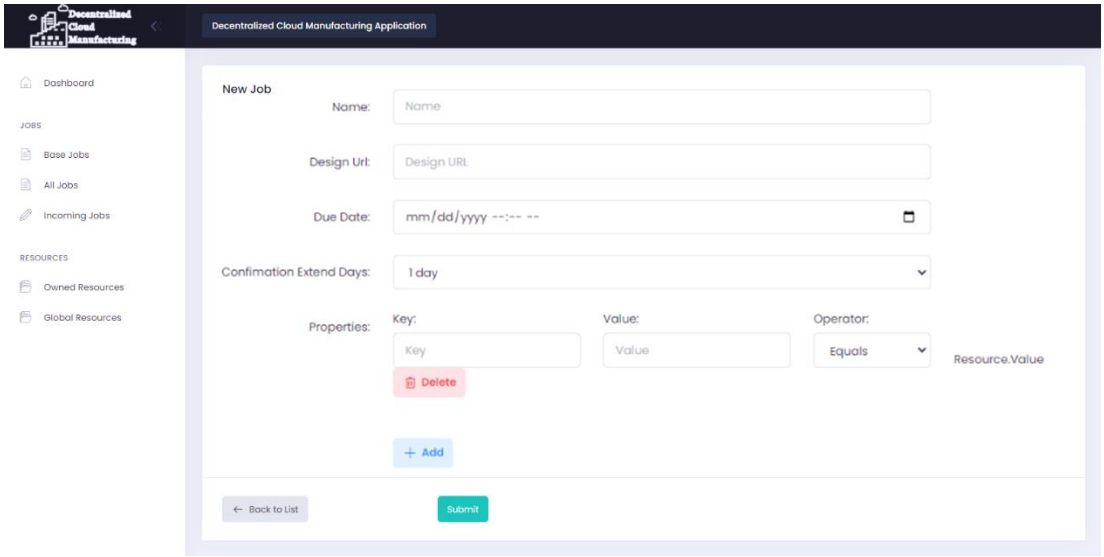
Kök işlerin listesi üzerinden (Şekil 4.23.) ilgili kök iş satırındaki “Jobs” tuşu ile alt işlerin listesine ulaşılır (Şekil 4.25.). Alt işler, müşterinin dışarıya yaptırmak istediği işleri tanımlar. İş listesinde işin durumu, adı, bitiş tarihi, işin akıllı sözleşme adresi ve akıllı sözleşmenin ağdaki durumu görülebilir. Yeni bir alt iş tanımı yapmak için “Create New” tuşu ile iş tanımlama ekranına girilir.



Şekil 4.25. Müşterinin alt iş listesi.

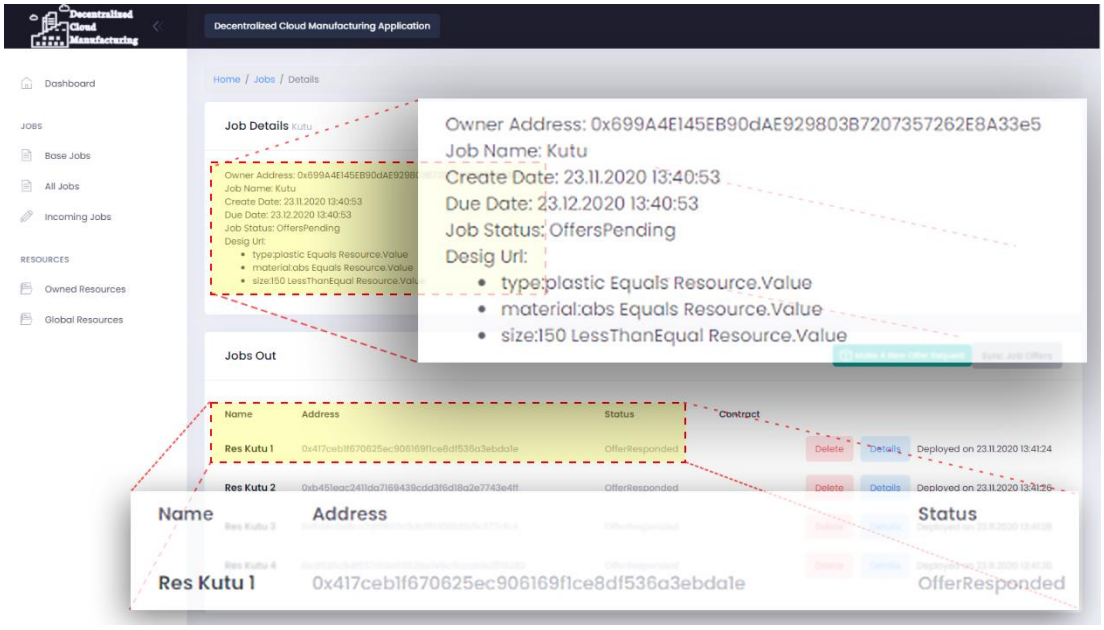
İş tanımlama ekranında (Şekil 4.26.) bulunan alanların detayları aşağıdaki listede açıklanmaktadır;

- Name (Ad): İşin adı tanımlanır.
- Design Url (Tasarım adresi): İşe ait bir tasarım dosyası varsa bu dosyaya erişilebilecek bir url buraya girilir.
- Due Date (Bitiş Tarihi): İşin hangi tarihte biteceği bilgisi girilir.
- Confirmation Extend Days (Onay uzatma günleri): İş tamamlandığında, ücret aktarılmadan kaç gün ertelenebileceği tanımlanır. Müşteri işi teslim aldığında bu süre kadar ücretin karşı tarafa transferini bekletebilir. Süre bittiği zaman akıllı kontratta bulunan Ethereum değeri karşı tarafa aktarılır.
- Properties (Özellikler): İşe ait özellikler anahtar-değer ikilisi şeklinde çoklu tanımlanabilir. Kaynak havuzundan kaynak araması yapılırken burada tanımlanan bilgiler ile uygun kaynaklar filtrelenir. Tanımlanan anahtar, kaynakların tüm özelliklerindeki anahtarlarda aranır. Eşleşen anahtarların değerleri burada tanımlanan operatör tanımına göre karşılaştırılır. Operatör tanımları şu tanımları içerir: eşittir, küçüktür, küçük eşittir, büyüktür ve büyük eşittir. String ifade olarak tüm tanımlar yapılır ancak karşılaştırma anında sayısal değer karşılaştırmaları sayısal olarak karşılaştırılabilmektedir.



Şekil 4.26. Yeni iş tanımlama ekranı.

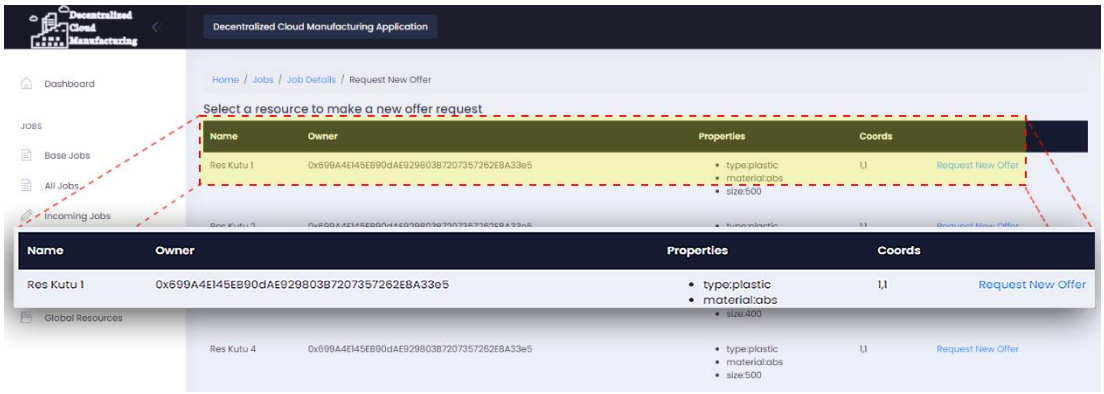
İş ayrıntılarına erişmek için iş listesindeki ilgili işin satırında bulunan "Details" tuşuna basılır. İş ayrıntıları sayfasında (Şekil 4.27.) işin ayrıntılı bilgileri ve talepte bulunulan kaynakların listesi görüntülenir.



Şekil 4.27. Müşterilerin iş ayrıntıları arayüzü.

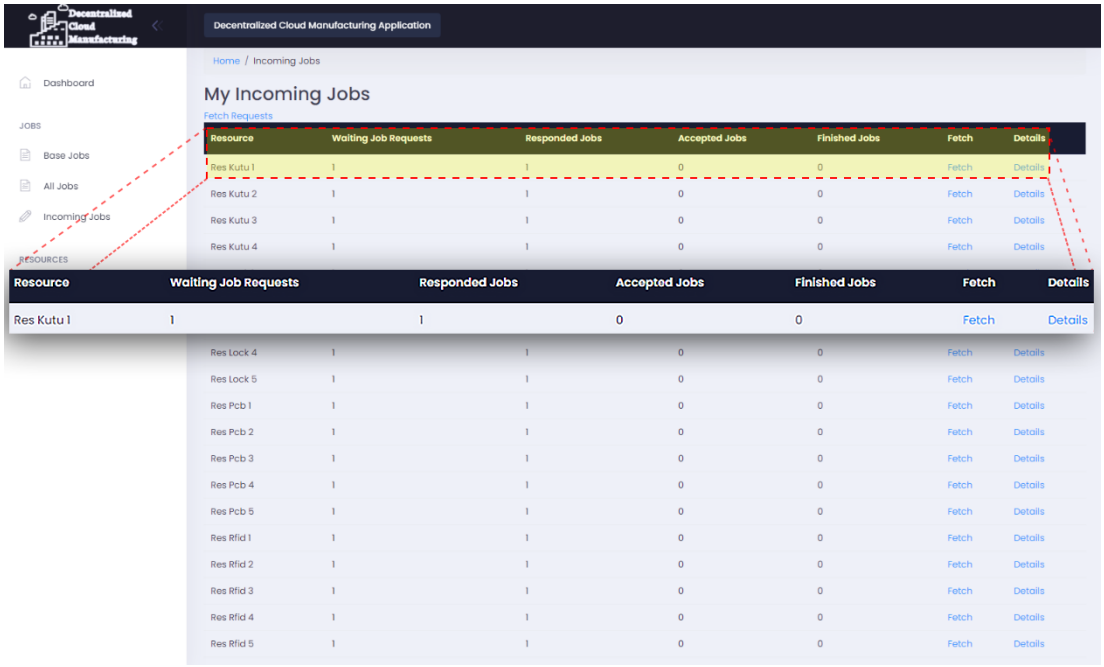
İş sahibi, işine uygun bir kaynağa "Job Details" başlığı altında "Make a new offer request" linkinden teklifte bulunabilir. "Request New Offer" arayüzünde (Şekil 4.28.) işe uygun olan kaynaklar filtrelenerek listelenir. Bu arayüzde iş ile uyumlu kaynaklar,

kaynakların özellikleri ve kaynakların bulundukları konumun koordinat bilgileri bulunur. Bu bilgiler altında iş sahibi istediği kaynağa teklifte bulunabilir. Şekil 4.28.'de örnek bir arayüz verilmiştir. İş ile uyumlu olan tüm kaynaklar özelliklerine göre filtrelenerek getirilmiştir. Burada tanımlanan işin özellikleri; "type:plastic, material:abs, size:150" şeklinde tanımlanmıştır. Buna özelliklere uyan kaynaklar listelenmektedir. "type" ve "material" anahtarlarının değerleri eşitlik operatörü ile, "size" anahtarının değeri ise küçük eşit operatörü ile karşılaştırılmıştır. "size <= Kaynak.Deger" şeklinde işletilmektedir. Kullanıcı taklif almak istediği kaynak için "Request New Offer" bağlantısına tıklar ve teklif almak için ilgili kaynağın akıllı sözleşmesine isteğini yazar. Aynı zamanda yapılan teklifler iş için oluşturulan akıllı sözleşmeye de işlenir.



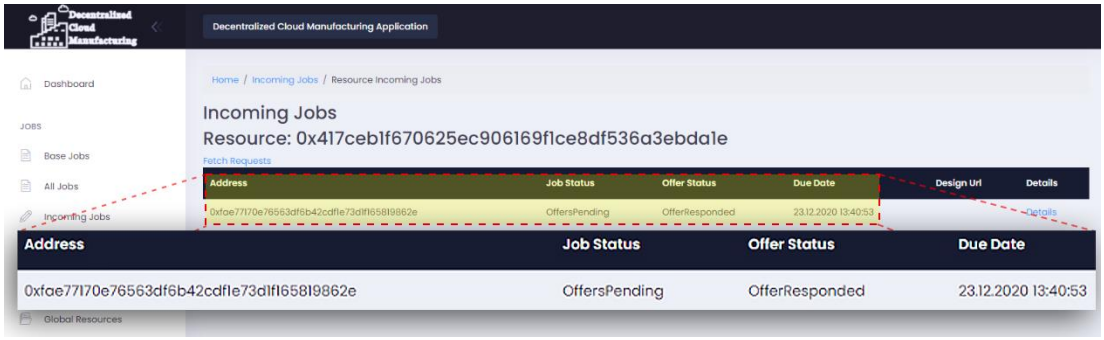
Şekil 4.28. Yeni tekliflerde kaynak arama arayüzü.

Kaynak sahipleri, kaynaklarına yapılan iş tekliflerini "Jobs > Incoming Jobs" menüsü altından görüntüleyebilirler. Gelen iş taleplerinin "Fetch Requests" bağlantısı kullanılarak yerel veritabanına senkronize edilmesi gerekmektedir. Senkronizasyon işlemi, kaynakların akıllı sözleşmelerinin "waitingJobs" adres dizisinden işlerin akıllı sözleşmelerinin adreslerini alır ve yerel veritabanındaki "IncomingJob" tablosu ile işin detaylarını eşitler. Tüm bu bilgiler Şekil 4.29.'da görüldüğü gibi ekranda, kaynaklara gelen talep özetleri olarak listelenir. İlgili kaynağın detaylarına "Details" bağlantısı ile erişerek kaynağa gelen tüm iş talepleri görüntülenebilir.



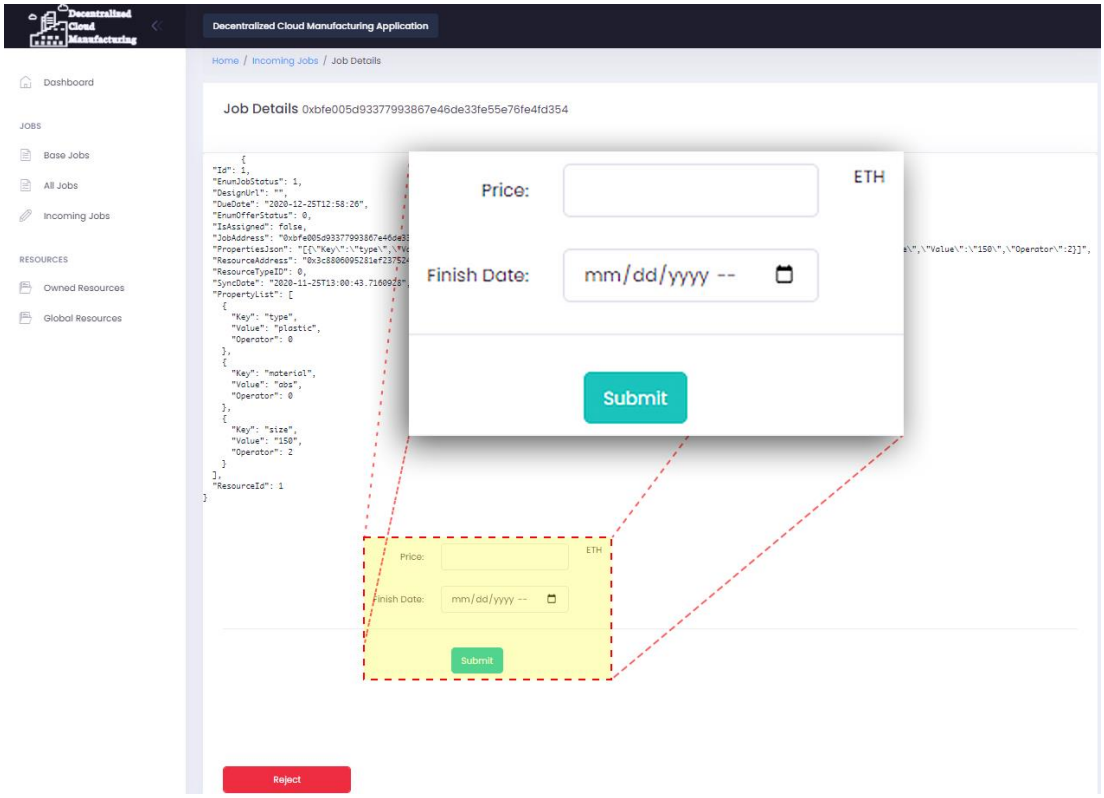
Şekil 4.29. Kaynaklara gelen iş taleplerinin özetleri.

Bir kaynağa gelen iş talepleri Şekil 4.30.'da görüntülenmektedir. İş talebinin akıllı sözleşme adresi, işin durumu, teklifin durumu, işin son tarihi ve varsa işin tasarım urlsi listelenir. Bu işi kabul veya reddetmek için "Details" bağlantısına tıklanır.



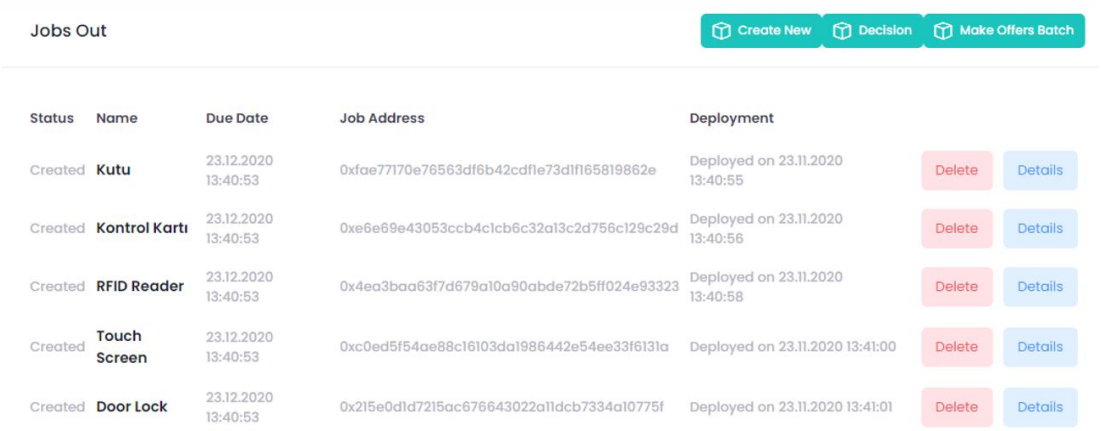
Şekil 4.30. Kaynağa gelen tüm iş talepleri.

Şekil 4.31.'de gelen bir iş teklifinin detayları görülmektedir. Bu sayfada iş kabul edilmek isteniyor ise teklif edilen fiyat Ethereum cinsinden yazılır ve teslim tarihi girilerek gönderilir. İşin akıllı sözleşmesine ilgili teklifin bilgileri yazılır. İş reddedilmek isteniyorsa "Reject" tuşu ile reddedilebilir.



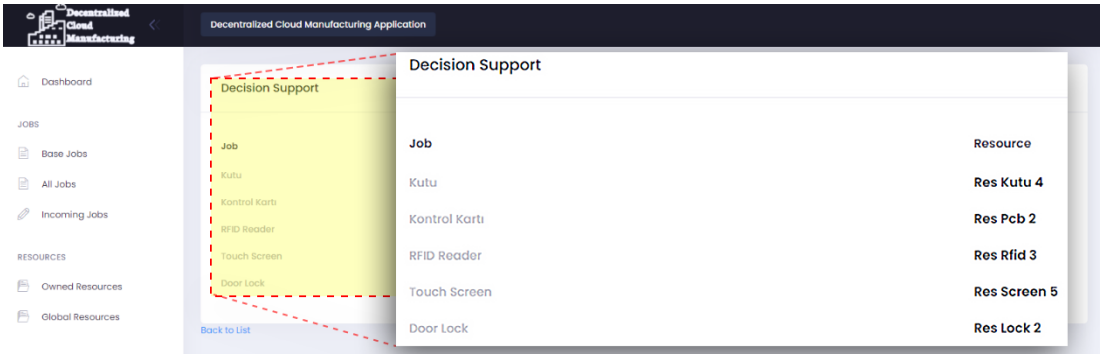
Şekil 4.31. Kaynağa gelen bir iş talebi.

Karar destek modülü, kaynak sağlayıcılardan alınan iş tekliflerinin kullanıcının önceliklerine göre değerlendirildiği bir modüldür. Bu modüle erişmek için Şekil 4.32.'de gösterilen kök iş ekranındaki "Decision" tuşuna basılır.



Şekil 4.32. Kök iş ekranı.

Şekil 4.33.'de gösterilen karar destek ekranı, kök işte bulunan tüm alt işleri ve tüm alt işlerin cevaplanan tekliflerini toplayarak kullanıcının önceden belirlediği öncelik matrisi ile kaynak planlama algoritmasını çalıştırır. Bu algoritma ile bir kök işe ait tüm alt işlerin hangi kaynaklara atanacağı önerisi elde edilir. Alt işlerin atanacağı kaynaklar, maliyet (C), teslimat zamanı (T), kalite (Q) ve risk (R) parametrelerine göre değerlendirilir. Genetik algoritma ile iş-kaynak dizilimi yapılır ve bu dizilimin uygunluğu AHP ile değerlendirmeye alınır. Kullanılan algoritmanın detaylarına "DCMApp Üzerinde Geliştirilen Kaynak Planlama Modeli" başlığından erişilebilir.



Şekil 4.33. Karar destek ekranı.

4.2.5. DCMApp yazılımının değerlendirilmesi

Önerilen model, uygulanabilirlik ve güvenilirlik açısından iki başlık altında değerlendirilmiştir. Uygulanabilirlik bakımından, diğer dağıtık bulut imalat uygulamalarının aksine modelin halka açık bir BC ağında nasıl konfigürasyon

yapılabileceği ile ilgili birkaç kısıtlama mevcuttur. Güvenirlik bakımından, güvenilir aracı ihtiyacının ortadan kaldırılmasının güvenirliğe nasıl katkıda bulunduğu ve nasıl güven ortamı sağlanabileceğine ilişkin değerlendirmeler ifade edilmiştir.

Bulut imalat sistemlerinin halka açık bir BC ağında nasıl çalıştırılabileceğine dair örnek bir çalışma yapılmıştır. Geliştirilen uygulama ile kullanıcılar, bulut üretim sisteminde merkezi bir yapı olmadan iş kaynaklarını dağıtılmış bir şekilde paylaşabilirler. Bu uygulamanın geleneksel imalat uygulamalarından farkı anlaşmaların/sözleşmelerin BC ağında yapılmasıdır.

Halka açık/genel bir ağın kullanılması, uygulamanın herhangi bir izin veya üyelik olmadan herhangi bir kişi tarafından kullanılabilmesine imkân sağlamıştır. Özel BC ağında sağlanan veri güvenliği genel BC ağlarında şifreleme yöntemleri ile sağlanmıştır. Özel BC ağı tarafından sağlanan işlem gizliliği, genel BC ağlarında mümkün değildir. Bu özellik, uygulamada gerçekleştirilen işlemleri daha şeffaf hale getirir.

Bu tez çalışması kapsamında geliştirilen uygulama genel BC ağı üzerinde çalışmak üzere tasarlanmıştır. Buna rağmen bazı gerekli bilgiler genel BC ağında depolanmamıştır. Bu bilgiler yerel veritabanlarında saklanır. Bu veritabanı kaynak bilgisini BC ağı dışındaki başka bir havuzdan alır. Bu depo, merkezi veya dağıtılmış bir yapıdaki sunucu olabilir. Tüm bilgileri işlem ücreti gerektiren genel bir BC ağında saklamak yerine ürün tasarımı, kaynak bilgileri, özel mesajlar gibi detaylı bilgiler farklı kaynaklarda saklanabilir, sistem farklı kaynaklardan ayrıntılı bilgileri alabilir. Bu kanallar özel sunuculardaki veya bulut hizmetlerindeki depolama alanları olabilir. Ethereum ağında bir işlem ücreti bulunduğundan ağda tutulan bilgilerin boyutu çok önemlidir. Ethereum ağındaki bilgiler, taraflar arasındaki anlaşmanında sadece temel bilgilerini içermelidir. BC ağı dışına aktarılan bilgiler taraflar arasındaki anlaşmayı ihmal etmemelidir.

Hâlihazırda kullanılan genel BC ağının kullanılması ağın oluşturulması için gerekli olan ek maliyetten kurtarır. Bu şekilde kullanıcılar mevcut BC ağını herhangi bir

yatırım yapmadan kullanabilirler. Genel BC ağını kullanan kullanıcılar yalnızca yazma işlemi için bir ödeme yapar. Özel ve konsorsiyum BC ağlarında yönetici(ler) vardır. Bu ağlar belirli kişiler tarafından yönetilir. Ağın sürekli ayakta kalabilmesi için işletme maliyeti oluşur. Bununla birlikte bu uygulamada ağın hayatta kalabilmesi için ek bir ücrete gerek yoktur, alınmamaktadır.

DCMApp ile kullanıcılar, gruba üye olmadan doğrudan genel BC ağları üzerinden sisteme katılabilirler. Ancak; özel ve konsorsiyum BC ağ modellerinde bir ağ oluşturulmalıdır. Bu ağın bir parçası olabilmek için, ağı oluşturan yöneticilerden izin alınması gerekir ve belki de bir ücret talep edilebilir. Bu nedenle; önerilen model yukarıda belirtilen nedenlerle özel ve konsorsiyum ağlarından daha ekonomiktir. Ağ üzerinde harcanan toplam maliyet, uygulamanın amacına ve ölçeğine göre değişir.

SC'lar, kullanıcıların BC ağında ikili anlaşmalar yapmalarını sağlar. SC ile yapılan bu anlaşmalar BC ağının güvenlik özelliklerini içerir. Bu nedenle, sözleşmeleri güvence altına almak için ek güvenlik önlemlerine gerek yoktur. Bu tez çalışması kapsamında BC ağının sadece değer aktarmak için değil, aynı zamanda SC'lar ile bir üretim uygulamasının nasıl çalıştırılabileceği gösterilmektedir.

Bu tez çalışması kapsamında SC'ı destekleyen halka açık BC ağından biri olan Ethereum ağı kullanılmıştır. Ethereum ağı, SC destekleyen en popüler ağdır ve belgelerine kolaylıkla erişilebilmektedir. SC destekleyen başka bir BC ağını kullanmakta mümkündür. Gelecekte; BC ağındaki yeni teknolojik gelişmeler, daha uygun maliyetli yeni BC ağları geliştirilebilir. DCMApp bu ağlar kullanılarak tekrar kurulabilir. Böylece, daha uygun maliyetli DCMApp geliştirilebilir.

Kullanıcıların uygulamada oturum açarken herhangi bir üyelik sistemine ihtiyaç duymaması, kullanıcı adı ve şifre gibi bilgilerin merkezi bir noktada saklanmadığını gösterir. Kullanıcılar uygulamayı kullanırken uygulamayı doğrudan bilgisayarlarından açabilir ve kriptopara cüzdanlarının özel anahtar bilgilerini uygulamaya girerek genel BC ağında özel işlemler gerçekleştirebilirler.

Uygulama dağıtık bir yapıya sahip olduğu için ve sadece kullanıcı verilerinin yerel veritabanında depolandığından uygulama herhangi bir kesintiden etkilenmeyecektir. Kesintiden etkilenmemesinin sebebi anlaşmaların Ethereum ağında depolanması ve ağın dağıtık olmasıdır. Ethereum ağına bağlı herhangi bir noktadan bilgiye erişim mümkündür. Ayrıntılı bilgilerin merkezi bir noktadan uygulamanın lokal veritabanına aktarılması zayıf bir nokta olarak görülebilir. Bununla birlikte, depolardaki bir kesinti yalnızca güncel bilgilerin alınmasını önleyecektir. Yerel veritabanındaki bilgilerle uygulama kesintisiz olarak çalışmaya devam edebilecektir. Örnek olarak; merkezi olarak düşünülen kaynak havuzları hizmet veremezse; kullanıcılar uygulamalarındaki yerel veritabanlarından mevcut bilgilerle çalışmalarına devam edebileceklerdir. Kaynak havuzlarından hizmet alınamaması yalnızca kaynakların yeni durumu hakkında bilgi alınmasını engelleyecektir. Bu iki farklı soruna neden olabilir. İlk durumda, diğer kullanıcılar ağda yağa yeni bir kaynak eklendiğinin farkında olmayacaktır. Sistem DCMApp'ın yerel veritabanlarında bulunan eski kaynak bilgileri ile sürdürülecektir. İkinci durumda, mevcut bir kaynağın bilgileri değiştirilebilir veya kaynak pasif olabilir. Bu durumda, eski bilgilere dayanarak bu kaynağa iş teklifleri gönderilmeye devam edecektir. Kaynak sahibi, kaynağın mevcut durumunu diğer kullanıcılarla paylaşamaz, bu nedenle önceki durumdan gelen iş teklifleri kabul edilmeyebilir. Böylece, kaynak bilgilerinin değiştirilmesi sistemin çalışmasını engellemeyecektir. Depo hizmeti gerekirse yeni sunucularla da çoğaltılabilir. Hükümetler tarafından erişim kısıtlaması gibi olası engellere karşı, BC ağları kesintisiz çalışmaya devam edebilmektedir. Ethereum ağına erişebilecek herhangi bir noktaya erişim yeterli olacaktır. Bu nedenle hükümetlerin DCMApp'i kısıtlaması mümkün değildir.

DCMApp, daha önce söylediğimiz gibi dağıtılmış bir yapıda çalışma yeteneğini Ethereum ağına borçludur. Ethereum ağı çok güvenilir bir ağdır ve bu ağda gerçekleştirilen işlemleri değiştirmek neredeyse imkânsızdır. Ağdaki geçmiş işlemleri değiştirmek için değişim gerçekleştirmek isteyen düğüm sayısının toplam düğüm sayısının en az %50'si olması gerekir. Bu saldırıya %51 saldırısı adı verilir. Bu saldırıya teşebbüs edilebilmesi için, anlık işlem gücünün yarısından fazlasına uzun süre sahip olunması gerekir. 30.11.2020 tarihinde Ethereum ağının işlem gücü

ortalama 150 TH/saat'dir. Yalnızca 1 saat boyunca bu gücün %51'ini elde etmenin ortalama maliyeti 458,000$'dır. Bu değerler, bilinen Ethereum madencilik şirketlerinin ortalama değerlerinden elde edilmektedir. Böylece, Ethereum ağında veri depolamanın, üçüncü taraf bir şirketin bilgisayarında verileri depolamaktan çok daha güvenli olduğu kolayca anlaşılabilir.

Bir JSC oluşturmanın maliyeti, testlerde yaklaşık 0.02 ETH olarak hesaplanır. Teklif verme maliyeti 0.01 ETH olarak hesaplanılır. Diğer işlemlerin yaklaşık 0.01 ETH'a mal olduğu bulunmuştur. Bu değerler, küçük bir işletme için yüksek değerler olarak görülebilir. Ancak; güvenli ve dağıtık bir ağda yapılan dijital anlaşmaların bir maliyeti vardır. Bu anlaşmalar yalnızca bir veya birden fazla bilgisayarda saklanmazlar. Ayrıca Ethereum ağına hizmet veren binlerce bilgisayarda saklanır ve işletilirler. Bu bilgilerin Ethereum ağında bilgileri saklamak ve işletebilmek için verilen ücretler aslında Ethereum ağında sonsuza kadar saklanması için ödenmiştir. Ömür boyu veri depolayabilen böyle bir hizmet için bu ücretler çok düşüktür.

Bu çalışmayı bir merkezi bulut imalat platformuyla karşılaştırırsak, merkezi uygulamanın güvenlik açısından başarısız olduğu açıktır. Merkezi bir uygulama, bir kişiye veya şirkete ait bir sunucuda çalıştırılmalıdır ve bunun bir maliyeti vardır. Merkezi bir uygulamada yalnızca sunucu maliyetleri değil, aynı zamanda uygulamayı geliştirmek, sürdürmek ve yönetmek için de bir finansmana ihtiyaç vardır. Merkezi bir sistemin ticari kaygısı olduğundan, bir ücret yatırımcılar için tahsis edilir. Kullanıcıların hizmet alabilmeleri için sisteme bir ücret ödemeleri gerekir. Fiyatlandırma politikası şirkete göre değişebilir. Bazı şirketler periyodik ücret alırken, bazıları işlem başına komisyon alır.

Merkezi bir bulut imalat platformu, özel bir BC ağı ile tasarlanmış bulut imalat platformu ve DCMApp örneğindeki gibi herkese açık BC ağı ile geliştirilmiş bulut imalat platformu maliyetleri açısından kıyaslanmıştır. Bu kıyaslamada yazılım geliştirme süreçlerinin oluşturduğu maliyetler göz ardı edilmiştir.

Merkezi bir bulut imalat uygulamasının 1 aylık süre ile Amazon üzerinden çalıştırılmak istenmesi durumunda ihtiyaç duyulabilecek kaynaklar ve fiyatları belirtilmektedir. Uygulama sunucusu olarak, 4 vCPU, 8 GiB RAM ve 100 GB SSD depolama özelliklerine sahip bir a1.xlarge tipinde EC2 sunucusunun maliyeti aylık 65.41 USD'dır. Veritabanı sunucusu olarak en az aynı özelliklerde bir sunucu daha gerekli olacaktır. Toplamda aylık 130,82 USD maliyeti olacaktır. Bu konfigürasyon minimum düzeyde tutulmuştur ve ihtiyaca göre arttırılması gerekmektedir.

Özel bir BC ağının 1 aylık süre ile işletilmesi için Amazon Managed Blockchain hizmetinin saatlik ücretleri şu şekildedir; standart ağ üyelik ücreti $0.55, 2 eş düğüm maliyeti $0.139, her bir düğüm için 500 GiB depolama maliyeti $0.139, 100 MB'lık saatte veri yazma ücreti de $0.01 olacaktır. Toplamda saatlik $1.93'lık bir maliyet söz konusudur.

DCMApp'de ise uygulama sunucuya ihtiyaç duymamaktadır. Sadece anlaşmaların işletilebilmesi için ağa işlem ücreti verilmelidir. Bir JSC oluşturmanın maliyeti daha önceden de belirtildiği gibi 0.02 ETH, teklif verme ve diğer işlemlerin maliyeti yaklaşık 0.01 ETH'dir. Görüldüğü üzere DCMApp, işlem sayısına göre ücret almaktadır. Yüksek miktarda işlem yapılmak istendiğinde veya büyük miktarda veri saklanmak istendiğinde özel BC ağlarının maliyet açısından avantajlı olduğu görülmektedir. Altyapı maliyet açısından ise herhangi bir sunucu masrafı olmadığı için DCMApp oldukça avantajlıdır. Maliyet açısından kıyaslamalar yapılırken diğer çözümlerin güvenlik açısından getirdiği riskler de unutulmamalıdır.

Bulut üretim ve BC destekli bulut üretimi karşılaştırmasında öne çıkan özellikler Tablo 4.1.'de özetlenmiştir. Ayrıca DCMApp'ın artıları ve eksikleri
Tablo 4.2.'de verilmektedir.

Tablo 4.1. Bulut imalat ve BC destekli bulut imalat

	Bulut İmalat	BC Destekli Bulut İmalat
Güvenilirlik	3. taraf kadar güvenilir.	Güvenirliği bütün BC ağı sağlar.
Sürdürülebilirlik	Merkezi sunuculara bağlıdır.	BC ağına bağlıdır.
Fiyat	7/24 çalışan sunucuların maliyeti, bakım maliyeti, kurulum maliyeti	Yalnızca işlem için ücretlendirme
Geliştirme	Kolay	Zor
Veri güvenliği	Veri değiştirilebilir.	Veri asla değiştirilemez.

Tablo 4.2. BC destekli bulut imalat modelinin avantajları ve dezavantajları

Avantajları
Taraflar arasındaki anlaşmalar aracısız yapılır.
SC tarafından belirlenen imalat sözleşmeleri değiştirilemez, dokunulmazdır. Taraflardan herhangi biri sözleşmede belirtilen koşulların ötesine geçemez.
İşlemler geri döndürülemez. Tüm işlemler güvence altına alınmıştır.
Mevcut genel BC altyapısı kullanıldığından, altyapı yatırımına ihtiyaç duymaz.
Verimli bir nakit akışı vardır. Ödemeler, kripto para birimi kullanılarak SC'lar aracılığıyla taraflar arasında yapılır.
Hükümetler tarafından kısıtlanamazlar.
Kesintiye uğramaz. Sürdürülebilirlik daimidir. Dağıtık bulut imalat uygulaması daima erişilebilirdir.
Dezavantajları
Uygulamayı genişletebilmek için, uygulamada kullanılan yazılım dilinin yanı sıra SC yazımı için Solidity dilini de bilmek gerekir.
Şeffaftır. Anlaşmaların verileri genel BC ağında tutulduğundan, verileri okumak mümkündür. Verileri okumak uzmanlık gerektirir.
İşlemler geri döndürülemez. Yanlışlıkla yapılan işlemleri düzeltmek imkânsızdır.
İşlem ücreti her yazma işlemi için ödenmelidir.

- Güvenilirlik: Bulut imalatta (CM) kullanıcılar, kullanıcılara uygulamayı sunan aracılara güvenir. Sistem aracının güvenirliği kadar güvenirdir. Bununla birlikte; BC destekli CM'de ise merkezi bir yapı yoktur, tek bir noktaya bağlı değildir ve tüm BC ağı sistemin güvenirliğini sağlar.
- Kullanılabilirlik-Sürdürülebilirlik: CM uygulamaları merkezi sunucularda çalışır. CM uygulamasının çalışma süresi tamamen merkezi sunucu sistemlerine bağlıdır. BC destekli CM uygulamalarında, uygulamalar BC ağında çalıştırıldığından, BC destekli CM uygulamalarının merkezi sunuculara

ihtiyaçları yoktur, BC ağında çalışırlar. BC ağındaki herhangi bir düğümün başarısız olması durumunda, ağdaki diğer düğümler ağı çalışır durumda tutar.

- Fiyat: CM uygulamaları 7/24 çalışıyor olmalıdır. Uygulamaların sürekli çalışması için bir sunucu gereklidir. Sunuculara ödenen ücret belirli süreler boyunca sürekli olarak devam eder. BC destekli CM'de, uygulamalar BC ağı üzerinde dağıtık bir şekilde çalışacak şekilde tasarlanmışsa, yalnızca işlem ücretleri alınır. BC ağının ayakta kalması için sürekli olarak ödenen bir ücret yoktur.
- Geliştirme: BC destekli CM uygulamaları bir DApp olacağından, DApp'in geliştirilmesi geleneksel uygulama geliştirme süreçlerinden çok daha farklı ve zordur.
- Veri güvenliği: CM'de veritabanına erişimi olan herhangi bir kişi verileri değiştirebilir. Verilerin güvenliği, veritabanına erişen kişiye bağlıdır. Bununla birlikte, BC destekli CM'de değişim çok zordur çünkü veriler bloklara zincir şeklinde yazılır. Bu genel BC ağlarında neredeyse imkânsızdır.

Geliştirilen uygulama, sürekli yaşayan bir ağ üzerinde dağıtılmış bir şekilde çalışabilir. Bu mimaride hazırlanan yazılımın tek bir sürümünün kullanılması zorunlu değildir. Aynı zamanda Ethereum ağında bulunan kullanıcılar tarafından farklı sürümler kullanılabilir. Kodun yeni sürümünün gelişimlerinden faydalanmak isteyen kullanıcılar yeni anlaşmalar için uygulamalarını güncelleyebilir. Uygulama açık kaynak olarak sunulduğundan, kullanıcılar onların istedikleri şekilde kodu değiştirerek farklı sürümlerle sistemi kullanmaya devam edebilir. Kişi veya organizasyondan bağımsız olarak, herhangi bir zamanda sürekli olarak geliştirilebilecek ve erişilebilir bir uygulamaya sahip olmak çok heyecan verici olacaktır. Merkezi uygulamalarda, şirket tarafından sunulan sürümün kullanılması zorunludur. Yapılacak anlaşmaların tarihi şirketin ömrü kadar olacaktır.

4.3. DCMApp Üzerinde Geliştirilen Kaynak Planlama Modeli

Merkezi olmayan ve dağıtık yapıdaki bir uygulamada kaynak yönetiminin gerçekleştirilmesi oldukça zor bir görevdir ve bu görevi, kurallar çerçevesinde sistemin otomatik olarak yapması gerekir.

Kaynak yönetiminin optimizasyonu için kaynakların ve işlerin müşteri isteklerine göre verimli bir şekilde eşleştirilmesi ve süreçlerin iyileştirilmesi gerekir. Bu bölümde kaynak tahsisi için zaman, maliyet, kalite, risk, şirket önceliği, erken teslim ve geç teslim parametrelerini dikkate alacak şekilde kaynak tahsis etme modeli önerilmiştir. Modelin matematiksel ifadesi " Bulut imalat modelinde kaynak tahsis etme" başlığı altında ayrıntılanmıştır. AHP ve genetik algoritmadan faydalanarak optimum kaynak atama işlemini gerçekleştiren hibrit kaynak planlama algoritması önerilmiştir. Algoritmanın ayrıntıları " Bulut imalat modelinde kaynak planlama modeli" başlığı altında anlatılmaktadır.

4.3.1. Bulut imalat modelinde kaynak tahsis etme

Tasarlanan bulut imalat sisteminde kaynaklar bir havuzdan servis olarak sunulmaktadır. Ancak servislerin sadece sunulması yeterli değildir. Kullanıcıların yüklediği işlerin analiz edilerek işin gereksinim duyduğu özelliklere göre en uygun servislere yönlendirilmesi gerekir. İlgili servislerin bulunması için iki ana adımdan oluşan bir süreç tanımlanır. Kaynak havuzundan gereksinimlerin karşılanabileceği ilgili servisler filtrelenir. Daha verimli bir hizmet için filtrelenen bu servisler arasından en uygun olabilecek servislerin seçilmesi gerekir.

Filtreleme işlemleri sistemde tanımlanan kaynak servislerinin özelliklerine göre gerçekleştirilir. Kaynakların özellikleri anahtar-değer ikilisi olarak sistemde hiyerarşik olarak tutulmaktadır. İhtiyaç duyulan kriterler mantıksal karşılaştırma operatörleri aracılığı ile seçilir. Bu işlem ile arama yapılacak servis miktarı azaltılarak çözüm uzayı daraltılmış olur. Elde edilen sonuçlar sadece koşulları sağlayan sonuçlardır. Talep edilenden fazla özelliğe sahip olan bir makine kullanılması, kapasitenin âtıl durumda

kalmasına neden olacaktır. Tam olarak istenilen özelliklere sahip bir servisin seçilmesi öncelikli olmalıdır.

Bu çalışmada imalat kaynakları; malzeme kaynakları, cihaz kaynakları, yazılım kaynakları, bilgi kaynakları, insan kaynakları, lojistik kaynaklar olmak üzere 6 farklı türde ele alınmıştır [72], [82]. Müşteriler tarafından istenilen işler öncelikle istemcinin tasarım bilgilerine göre alt işlere ayrılır. Alt işlemler belirli bir planda müşterinin istekleri doğrultusunda planlanır ve tamamlanmış bir şekilde sunulur. İşlemlerin gerçekleştirilmesi bulut imalat kaynaklarının durumuna ve hizmet koşullarına bağlıdır.

Tüm imalat kaynakları (MRG – Manufacturing Resource Group) imalat kaynaklarının tüm türlerini (RT – Resource Type) barındırır (Denklem 4.1).

$$\text{MRG} = \{RT_{ts1}, RT_{ts2}, RT_{ts3}, \dots\dots\ RT_{tsn}\} \tag{4.1}$$

Kaynakların türü t ile temsil edilirken, o türe ait kaynağın sayısı s ile temsil edilmektedir. n ise MRG içerisinde bulunan kaynak türü sayısını ifade etmektedir.

MRG içerisinde şu anda hizmet verebilecek kaynakları mevcut imalat kaynakları grubu (AMRG – Available Manufacturing Resource Group) olarak temsil edelim (Denklem 4.2). RT_{ts_w} içindeki hizmet verebilecek k. kaynak birimi $RU_{w,k}$ olarak ifade edilir.

$$\text{AMRG} = \{RU_{1,k}, RU_{2k}, \dots., RU_{w,k}, \dots\dots\ RU_{p,k}\}\ (1 \leq k) \tag{4.2}$$

Bu bölümde, nihai kaynağı bazı standartlara göre belirlemek için bir matematiksel model sunulmaktadır. Servislerin maliyeti (C), teslimat zamanı (T), kalite (Q) ve risk (R) gibi kriterlerin servis seçiminde dikkate alınması servis arama algoritmasını daha yetkin kılacaktır. TCQR parametreleri farklı işletmeler için farklı önceliklere sahip olabilir. Bazı işletmeciler için zaman daha önemliyken, bazısı için kalite, bazı işletmeler içinde maliyet daha ön plana çıkabilir. İşletmeler parametrelere kendi önem derecelerine göre bir ağırlık verirler.

Servis planlama algoritması, işletmelerin TCQR, erken teslim (ED) ve geç teslim (LD) parametrelerine faklı ağırlıklar ile değerlendirmesine izin vermelidir. Örnek olarak zamanın maliyetten daha önemli olduğu bir iş talebinde, zaman parametresi maliyet parametresine göre daha büyük bir ağırlığa sahip olmalıdır. Bu çalışmada TCQR parametreleri ile birlikte şirket önceliği, erken teslim (ED) ve geç teslim (LD) parametreleride dikkate alınmaktadır. Zaman (T) parametresi belirtilen kaynaktaki işlem süresini temsil etmektedir. Optimizasyon probleminin çözümü için geliştirilen model aşağıdaki varsayımları kabul eder:

- Aynı anda bir kaynak en fazla bir işe atanabilir.
- Her bir alt iş, yalnızca bir kaynağa atanabilir.
- Hazır makineler planlamaya dâhil edilir.
- Bir işlem bir makinede işletilmeye başlatıldığında, işlem tamamlanıncaya kadar durdurulamaz.

Optimizasyon probleminin çözümü için tasarlanan amaç fonksiyonu Denklem 4.3'te tanımlanmaktadır. w değerleri, her bir kıstas için müşterinin belirttiği ağırlıkları ifade eder.

$$\text{Min Z} = w_1T + w_2C + w_3(1-Q) + w_4R + w_5(1-P) + w_6ED + w_7LD \qquad (4.3)$$

En kısa teslimat süresi: Teslimat süresi imalat süresi ve farklı üretim kaynakları arasındaki lojistik süresi olmak üzere 2'ye ayrılır (Denklem 4.4).

$$T = imalat\ süresi + lojistik\ süresi \qquad (4.4)$$

$$T = \min\left[\sum_{w=1}^{n}\left(\sum_{k=1}^{n_w}\varphi RT_{ts_{wk}}t(wk)\right) + \sum_{w=1}^{n-1}\left(\sum_{k=1}^{n_w}\sum_{p=1}^{n_{w+1}}\varphi RT_{ts_{wk}}\varphi RT_{ts_{(w+1)p}}t(wk,(w+1)p)\right)\right] \qquad (4.5)$$

Bir iş parçası n adet iş parçasından oluşabilir. w iş parçasını, k iş parçasına atanan kaynağı temsil eder. Denklem 4.5'te bulunan $\sum_{k=1}^{n_w} \varphi RT_{ts_{wk}} t(wk)$, k kaynağının w alt işini üretme süresini temsil eder. Yine Denklem 4.5'te bulunan $\sum_{k=1}^{n_w} \sum_{p=1}^{n_{w+1}} t(wk,(w+1)p)$, w iş parçasının ihtiyaç duyduğu k kaynağından (w+1) iş parçasının ihtiyaç duyduğu p kaynağına nesnelerin taşınması için gerekli olan süreyi tanımlar.

En düşük maliyet: Üretim sırasında meydana gelen (imalat maliyeti ve lojistik maliyet) tüm maliyeti temsil eder (Denklem 4.6).

$$C = imalat\ maliyeti + lojistik\ maliyeti \tag{4.6}$$

$$C = \min\left[\sum_{w=1}^{n}\left(\sum_{k=1}^{n_w} \varphi RT_{ts_{wk}} c(wk)\right) + \sum_{w=1}^{n-1}\left(\sum_{k=1}^{n_w}\sum_{p=1}^{n_{w+1}} \varphi RT_{ts_{wk}} \varphi RT_{ts_{(w+1)p}} c(wk,(w+1)p)\right)\right] \tag{4.7}$$

Denklem 4.7'de verilen $\sum_{k=1}^{n_w} c(wk)$ ifadesi w. alt iş parçacığının k kaynağı tarafından üretilme maliyetini temsil eder. $\sum_{k=1}^{n_w} \sum_{p=1}^{n_{w+1}} c(wk,(w+1)p)$ ifadesi ise k kaynağından p kaynağına nesnelerin taşınma maliyetini temsil eder.

En yüksek kalite:

$$Q = \max\left[\prod_{i=1}^{jobs} \sum_{r=1}^{res} x_{ir} * Q(r)\right] \tag{4.8}$$

Tasarlanan amaç fonksiyonunda Q fonksiyonu (Denklem 4.8) kaliteyi temsil etmektedir ve 0 ile 1 arasında bir değer alır. En büyük değer en yüksek kalite oranını temsil eder.

En az risk:

$$R = \min\left[\prod_{i=1}^{jobs} \sum_{r=1}^{res} x_{ir} * R(r)\right] \tag{4.9}$$

R fonksiyonu (Denklem 4.9) bu makinede yapılmış olan daha önceki işlerin başarı oranına göre 0 ile 1 arasında değer üretir. R değeri, önceki işlerin gecikme miktarı, erken teslim oranı, makinenin arıza oranları değerlerinden hesaplanır. Değer ne kadar büyük olursa, risk oranı da o kadar yüksek olur.

Şirket deneyimi: Müşteriler, önceden belirlediği işletmelerin kaynaklarını öncelikli olarak kullanmak isteyebilirler. Önceliklendirilmiş olan bu kaynakların diğer kaynaklara göre daha ön sıralarda çözüm listesinde gösterilmesi gerekir. P değeri, her bir makine için 0 ile 1 arasında bir katsayı olarak belirlenir. 1 değeri tercih edilen en yüksek değer, 0 ise tercih edilen en düşük değer olarak nitelendirilir (Denklem 4.10).

$$P = \max\left[\prod_{i=1}^{jobs} \sum_{r=1}^{res} x_{ri} * P(r)\right] \quad (4.10)$$

Teslim süresinin optimizasyona Denklem 4.11 ve Denklem 4.12 ile dahil edilmiştir. Denklem 4.13 de ilgili kısıtlar verilmektedir.

$$ED = bitiş\ tarihi - (başlama\ tarihi + T) \quad (4.11)$$

$$LD = (başlama\ tarihi + T) - bitiş\ tarihi \quad (4.12)$$

$$ED \geq 0; LD \geq 0 \quad (4.13)$$

ED: Erken teslim; LD: Geç teslim; jobs: imalat işleri; res: imalat kaynakları

Optimizasyon modelinin kısıtları toplam süre, toplam maliyet, temel düzeyde kalite ve risk olmak üzere 4 terimden oluşmaktadır.

$$T_{mak.} - T \geq 0;\ C_{mak.} - C \geq 0; Q - Q_{min} \geq 0;\ R_{mak} - R \geq 0 \quad (4.14)$$

$T_{mak}, C_{mak}, Q_{min}, R_{mak}$ kullanıcılar tarafından belirtilir. Toplam süre, T_{mak}'dan, toplam maliyet C_{mak} 'dan, toplam risk R_{mak}'dan büyük olmamalıdır (Denklem 4.14). Ürünün kalitesi müşterinin istediği Q_{min} değerinden daha az olmamalıdır.

Kısıtlamalardan hiçbirine ulaşmayan çözüm ihmal edilmeli, diğerleri ise çözümü optimize etmek için ayrılmalıdır. Optimum çözüm, her bir göstergenin kendi kısıtlaması dâhilinde mümkün olduğunca kendi optimum değerine ulaştığı değerleri içerir.

4.3.2. Bulut imalat modelinde kaynak planlama modeli

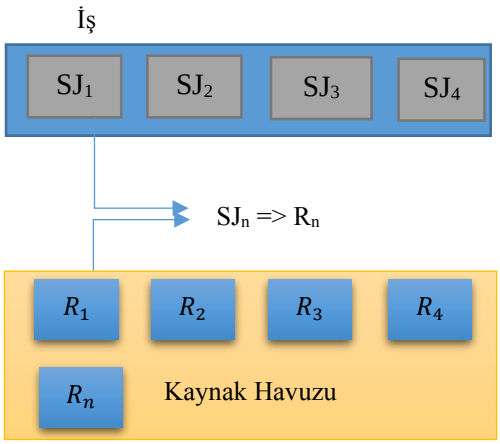
Bulut imalatta, binlerce kaynağın olduğu bir kaynak havuzu ile binlerce iş ve alt iş parçalarının müşteriye göre eşleştirilmesi matematiksel yöntemlerle çözülemeyecek kadar büyük bir problemdir. Bu nedenle bulut imalat ortamında alt işlere sahip olan bir işe ait kaynak planlamasının yapılabilmesi için bir sezgisel algoritmaya ihtiyaç vardır. Sezgisel algoritma, çözüm uzayının tamamının taranması yerine sezgisel olarak arama yaparak daha hızlı uygun bir çözüme ulaşmamızı sağlayacaktır. Bu çalışmada genetik algoritma kullanılarak çözüm kümesi araştırılmaktadır. Sezgisel algoritma kullanabilmek için uygunluk fonksiyonunun öncelikle tanımlanması gerekir. Bu problemde uygunluk fonksiyonu Denklem 4.15 de tanımlanmaktadır. x değeri, alt iş parçalarını, r seçilen kaynağı tanımlamaktadır.

$$Z_{mak.} = \sum_{j=1}^{n} f_{AHP}(x_j, r) \tag{4.15}$$

Bu çalışmada her bir siparişin alt işlerden oluştuğu ve bu alt işlerin sırası ile yapıldığı kabul edilmektedir. Her bir alt iş parçasının ilgili kaynakta işlenmesi durumunda bir $f(x_j, r)$ fonksiyonu çalıştırılarak seçilen kaynağın (r) bu iş için uygunluk derecesi AHP yöntemi ile hesaplanır. Aynı şekilde tüm alt parçalar için ayrı ayrı AHP fonksiyonu çalıştırılarak elde edilen uygunluk dereceleri toplanır ve çözümün toplam uygunluk derecesi elde edilir. Sezgisel algoritmanın sunduğu çözüm, bu uygunluk fonksiyonu ile değerlendirilir.

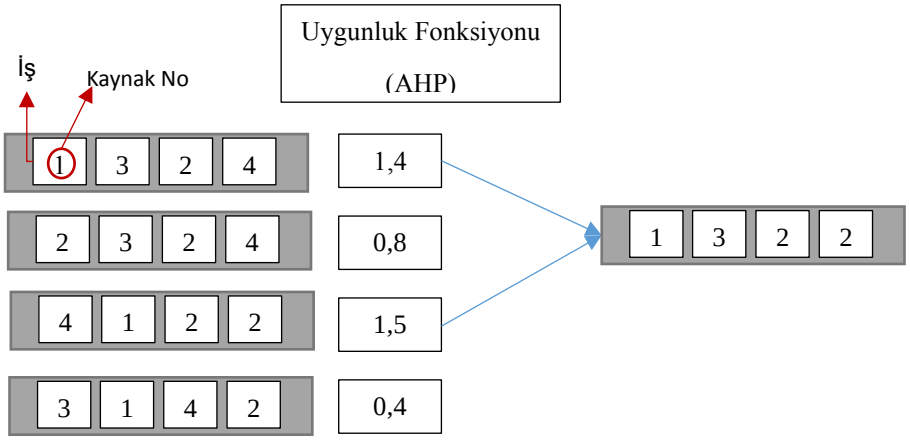
Şekil 4.34.'de bir siparişin 4 adet alt iş parçasından oluştuğu görülmektedir. Her bir iş parçasının ayrı ayrı hangi kaynakta işlenmesi gerektiği hesaplanmalıdır. Mevcut kaynaklar arasından kısıtları sağlayan çözümleri elde edebilmek için genetik algoritmadan faydalanılabilir. Genetik algoritma ile çözüm popülasyonu elde edilir.

Her bir olası çözüm, kromozondur ve her bir çözüm, bir gen dizilimi şeklinde ifade edilmelidir.



Şekil 4.34. İş kaynak eşleştirmesi.

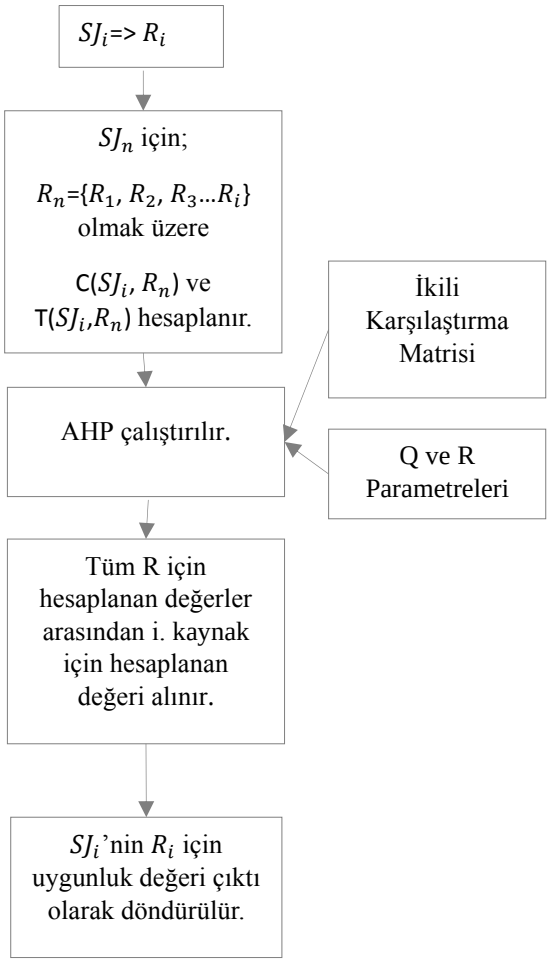
Şekil 4.35.'de her bir gen bir iş parçasını temsil etmektedir. Genlerin içinde bulunan sayı değerleri bu iş parçasının hangi kaynak ile eşleştirileceği ile ilgili bilgiyi temsil eder. Bir gen dizilimi (kromozom) bir çözüm olarak düşünülmektedir. Ancak bu çözümün ne kadar uygun olduğunu uygunluk fonksiyonu ile belirlememiz gerekmektedir. Uygunluk fonksiyonu AHP yöntemiyle elde edilir.



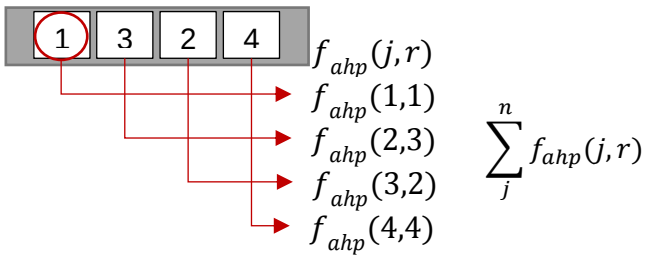
Şekil 4.35. Genetik algoritma.

Şekil 4.36.'de bir iş parçasının R_i kaynağına atanması durumunu değerlendiren fonksiyonun akış diyagramı verilmektedir. SJ_n alt iş parçasının R_n kaynağına atanması bir çözümün alt bileşenidir. Bu noktada çözümün bir bileşeninin değerlendirilebilmesi için SJ_niş parçasının tüm kaynaklar için AHP yöntemi ile değerlendirilmesi

gerekmektedir. AHP yöntemi alternatif kaynaklar için elde edilen uygunluk derecelerini hesaplamaktadır. AHP yönteminin çıktısı önerilen kaynak için hesaplanan uygunluk derecesidir. Elde edilen bu çıktı değeri her bir iş parçası için toplanır ve bir gen için toplam uygunluk değeri elde edilir (Şekil 4.37.).

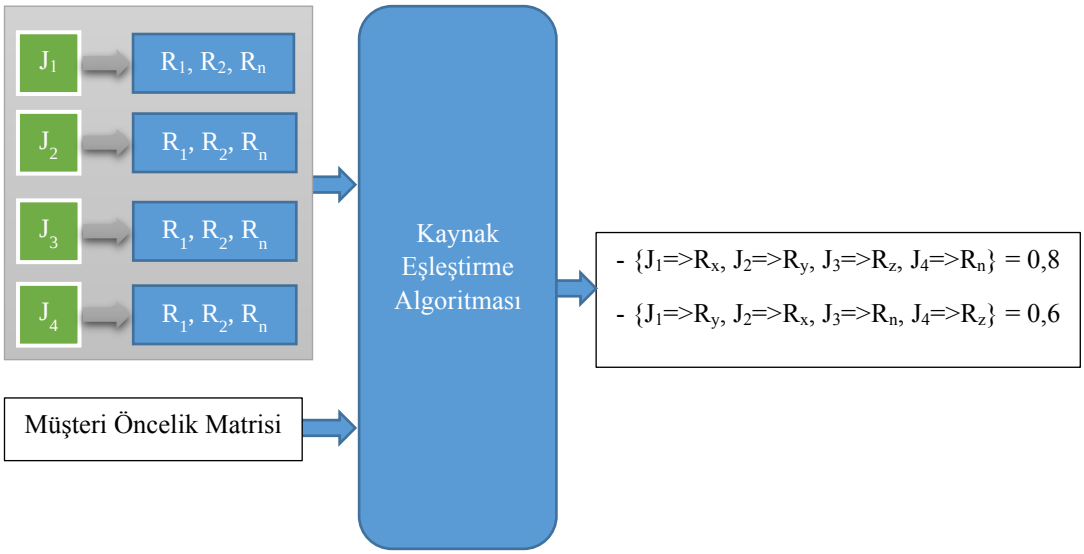


Şekil 4.36. İş parçası-kaynak ataması değerlendirme diyagramı.



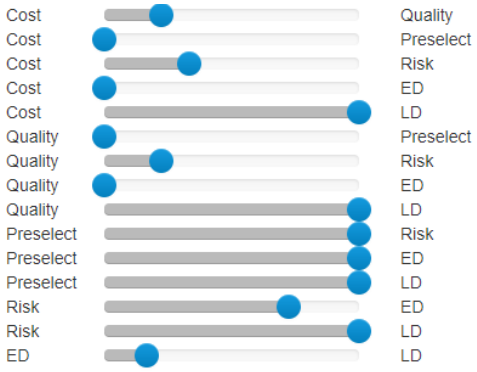
Şekil 4.37. Genin uygunluk değerinin hesaplanması.

Kaynakların eşleştirilmesi algoritmasına 2 parametre verilmektedir (Şekil 4.38.). Bu parametrelerden birincisi, her bir alt iş parçası için ayrı ayrı uygun olabilecek kaynakların listesidir. İkinci parametre ise müşterinin öncelik matrisidir. Algoritmadan beklenilen çıktı, her bir iş parçası için en uygun olan kaynağın belirlendiği alternatif çözüm listesidir.



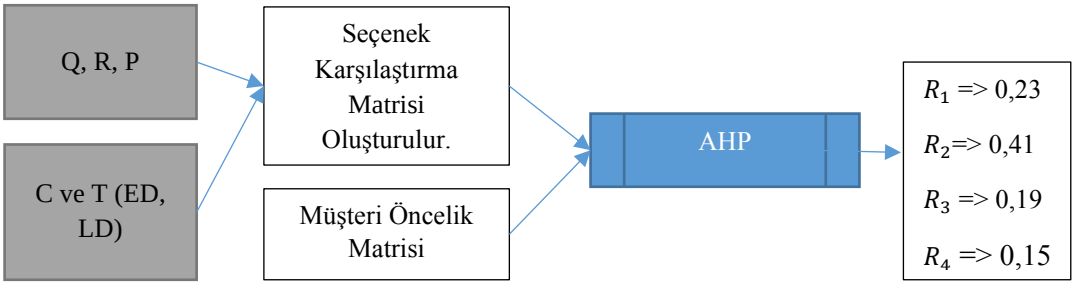
Şekil 4.38. Kaynak eşleştirme algoritması girdi ve çıktı diyagramı.

Genetik algoritma ile çözüm araştırılmaktadır, AHP ile çözümün uygunluk derecesi belirlenir. Genetik algoritmanın sunduğu çözümlerin müşteriye özel, kısıtlamalar dikkate alınacak şekilde değerlendirilmesi AHP fonksiyonu ile yapılır. Müşteri öncelik matrisinde; müşterinin T, C, Q, R, P, ED ve LD parametrelerinin ikili olarak karşılaştırma bilgileri bulunmaktadır. Şekil 4.39.'de müşteri öncelik matrisinin örnek bir kullanıcı arayüzü verilmektedir. Bu arayüz ile müşteri ikili karşılaştırmalardan hangisinin diğerine göre daha üstün olduğunu seçerek istediği önem derecelerini sisteme girer. İkili karşılaştırmalarda işaret hangi parametreye yakın ise o parametre diğerine göre daha üstün olarak kabul edilmektedir. Bu arayüzden elde edilen değerler 1 ile 9 arasında değerlere dönüştürülerek sisteme işlenir. Bu matris bir müşteriye ait sabit bir değer olarak hafızada saklanır. Bu çalışmada bu değerler kodda bir sabit olarak verilerek çalıştırılmıştır.



Şekil 4.39. Müşteri Öncelik Matrisi Kullanıcı Arayüzü.

Kaynakların ikili karşılaştırma matrisleri çalışma zamanında hesaplanılmaktadır. Zaman ve maliyet parametreleri her bir iş parçası ve her bir kaynak için ayrı ayrı hesaplanır. Kalite ve risk parametreleri ise sistem tarafından geçmiş verilere göre önceden hesaplanmış kabul edilir. Şekil 4.40.'da AHP fonksiyonunun girdileri ve çıktıları özetlenmektedir.



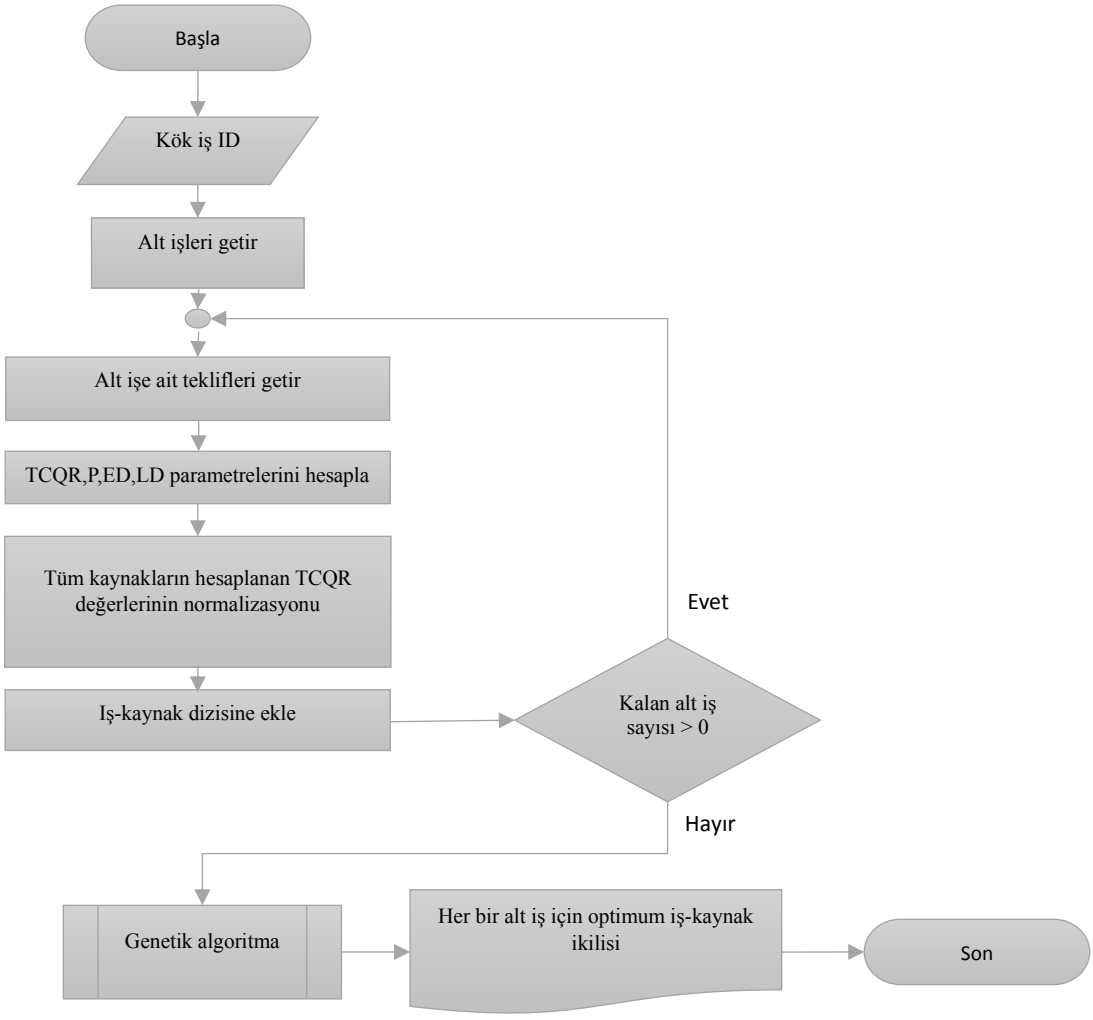
Şekil 4.40. AHP girdi ve çıktıları.

4.3.3. DCMApp kaynak planlama modeli yazılımı

DCMApp, gelen teklifler arasından kullanıcıya uygun olan teklifleri seçmesine yardımcı olan bir karar destek yazılımına sahiptir. Bu yazılım, bir kök işe ait tüm alt işlerine gelen teklifleri toplar ve kullanıcının önceden belirlemiş olduğu önceliklere göre bir önceki başlıkta bahsi geçen algoritma ile değerlendirmeye alır. Genetik algoritma fonksiyonlarında AForge.Genetic kütüphanesinden, AHP de kullanılan matris işlemlerinde ise MathNet.Numerics kütüphanesinden faydalanılmıştır.

Yazılım aşağıdaki katmanlardan oluşmaktadır.

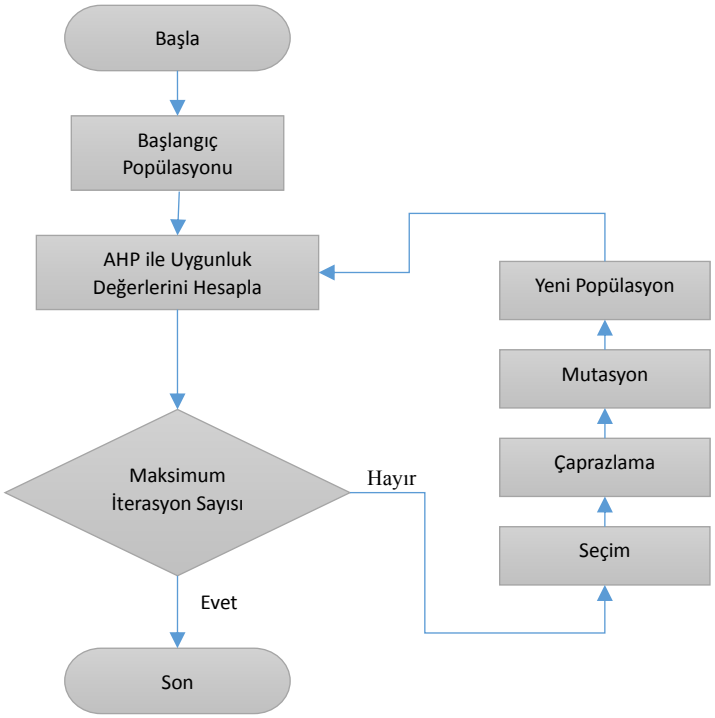
- DCM.Decision.Shared: DCMApp yazılımının tüm katmanları burada tanımlanan class, enum ve yardımcı fonksiyonları kullanırlar.
- DCM.Decision.Scheduler: İş-kaynak atama problemini yapan genetik algoritma fonksiyonlarını içerir
- DCM.Decision.AHP: Genetik algoritma içinde kullanılan uygunluk fonksiyonu olan AHP metodlarını içerir.



Şekil 4.41. DCMApp kaynak planlama algoritması.

Karar destek yazılımının giriş noktası DCM.Decision.Scheduler katmanında bulunur. Kullanıcıdan alınan kök iş ID girdisi ile Şekil 4.41.'da gösterilen algoritma çalıştırılır. Gelen id ile tüm alt işler yerel veritabanından sorgulanır. Gelen her bir teklif için

TCQR parametreleri hesaplanır ve her bir alt iş kapsamında ilgili tekliflerin TCQR parametreleri normalize edilir. Bu normalizasyon işlemi, değerlerin AHP'de kullanılabilmesi için gereklidir. Her bir iş için iş-kaynak alt dizileri oluşturulur. Oluşturulan veri seti genetik algoritmaya girdi olarak gönderilir. Genetik algoritma, AHP'yi içinde uygunluk fonksiyonu olarak kullanır ve optimum değerler hesaplanır. Çıktı olarak her bir iş için birer kaynak önerilir.



Şekil 4.42. Genetik algoritma akış diyagramı.

Genetik algoritmanın akış diyagramı Şekil 4.42.'de gösterilmektedir. Genetik algoritmada mutasyon oranı 0,25 olarak tercih edilmiştir. Maksimum iterasyon sayısı 250 olarak belirlenmiştir. Popülasyon boyutu 10 ve popülasyonlardan seçim metodu olarak ise "Elite Selection" tercih edilmiştir. Bunların dışında genetik algoritmanın ihtiyaç duyduğu kromozom yapısı ve uygunluk fonksiyonu tanımlamaları vardır. Kromozom yapısı "ShortArrayResourceChromosome" sınıfı ile tanımlanmıştır. Kromozomun uzunluğu toplam alt iş sayısı kadardır. Yeni bir jenerasyon oluşturmak için "Generate()" fonksiyonu kullanılır (Şekil 4.43.). Fonksiyon her bir işe atanan kaynak id değerini rastgele değiştirerek yeni bir kromozom oluşturur.

```
/// <summary>
/// Generate random chromosome value.
/// </summary>
///
/// <remarks><para>Regenerates chromosome's value using random number generator.</para>
/// </remarks>
///
1 reference | Baran Kaynak, 46 days ago | 1 author, 1 change
public override void Generate()
{
    for (int i = 0; i < length; i++)
    {
        int max = _dictResource[i];
        // generate next value
        val[i] = (ushort)rand.Next(max);
    }
}
```

Şekil 4.43. Kromozom oluşturma metodu.

"Mutate()" fonksiyonu, bir kromozomun rastgele tek bir geninin değiştirilmesini ve böylelikle mutasyonun gerçekleştirilmesini sağlayan fonksiyondur. Fonksiyon, Şekil 4.44.'de verilmektedir.

```
/// <summary>
/// Mutation operator.
/// </summary>
///
/// <remarks><para>The method performs chromosome's mutation, changing randomly
/// one of its genes (array elements).</para></remarks>
///
0 references | Baran Kaynak, 46 days ago | 1 author, 1 change
public override void Mutate()
{
    // get random index
    int i = rand.Next(length);
    int max = _dictResource[i];
    // randomize the gene
    val[i] = (ushort)rand.Next(max);
}
```

Şekil 4.44. Kromozom mutasyon fonksiyonu.

Kromozomların çaprazlanması, Şekil 4.45.'da gösterildiği gibi "Crossover()" fonksiyonu ile gerçekleştirilmektedir. Bir başka kromozom fonksiyona girdi olarak verilir ve rastgele işaretlenen bir çaprazlama noktasından çaprazlama yapılır.

```
/// <summary>
/// Crossover operator.
/// </summary>
///
/// <param name="pair">Pair chromosome to crossover with.</param>
///
/// <remarks><para>The method performs crossover between two chromosomes • interchanging
/// range of genes (array elements) between these chromosomes.</para></remarks>
///
0 references | Baran Kaynak, 46 days ago | 1 author, 1 change
public override void Crossover(IChromosome pair)
{
    ShortArrayResourceChromosome p = (ShortArrayResourceChromosome)pair;

    // check for correct pair
    if ((p != null) && (p.length == length))
    {
        // crossover point
        int crossOverPoint = rand.Next(length - 1) + 1;
        // length of chromosome to be crossed
        int crossOverLength = length - crossOverPoint;
        // temporary array
        ushort[] temp = new ushort[crossOverLength];

        // copy part of first (this) chromosome to temp
        Array.Copy(val, crossOverPoint, temp, 0, crossOverLength);
        // copy part of second (pair) chromosome to the first
        Array.Copy(p.val, crossOverPoint, val, crossOverPoint, crossOverLength);
        // copy temp to the second
        Array.Copy(temp, 0, p.val, crossOverPoint, crossOverLength);
    }
}
```

Şekil 4.45. Kromozom çaprazlama fonksiyonu.

Genetik algoritmanın ihtiyaç duyduğu uygunluk fonksiyonu Şekil 4.46.'da gösterilmektedir. Bu fonksiyon ile genetik algoritma bir kromozomun hedeflenene göre ne kadar uygun olduğunu hesaplar. Değerlendirme (Evaluate()) fonksiyonu bir kromozomu alır ve bu değeri iş-kaynak çözümüne dönüştürerek DCM.Decision.AHP katmanında bulunan AHP hesaplama (DCM.Decision.AHP.Calculator) modülüne aktarır. Burada AHP hesaplamaları yapılarak her bir iş için atanan kaynağın uygunluk değeri çıktı olarak alınır. Bu hesaplama tüm alt işler için ayrı ayrı yapılır. Her bir alt iş için elde edilen sonuç değerlerinin toplamı uygunluk fonksiyonunun çıktısı olarak dışarı verilir.

```
2 references | Baran Kaynak, 46 days ago | 1 author, 1 change
internal class FitnessFunction : IFitnessFunction
{
    private Dictionary<Job, List<Resource>> _jobResourceDict;
    private List<AHPPair<Criteria>> _criteriaMatrix;
    //private int maxResourceNumber = 0;
    private Dictionary<int, int> _dictJobResCount;

    1 reference | Baran Kaynak, 46 days ago | 1 author, 1 change
    public FitnessFunction(Dictionary<Job, List<Resource>> jobResourceDict,
        List<AHPPair<Criteria>> criteriaMatrix, Dictionary<int, int> dictJobResCount)
    {
        _jobResourceDict = jobResourceDict;
        _criteriaMatrix = criteriaMatrix;
        //maxResourceNumber = _jobResourceDict.Max(x => x.Value.Count);
        _dictJobResCount = dictJobResCount;
    }

    0 references | Baran Kaynak, 46 days ago | 1 author, 1 change
    public double Evaluate(IChromosome chromosome)
    {
        var sChromosome = (ShortArrayResourceChromosome)chromosome;
        var jobResDict = ConvertToSolution(sChromosome.Value);
        ToLog(jobResDict, 0);
        Calculator calc = new Calculator();
        double result = 0;
        foreach (var item in jobResDict)
        {
            var alternativeResources = _jobResourceDict[item.Key];
            Dictionary<Criteria, Dictionary<Resource, double>> criteriaPoints =
                GenerateAlternativeDict(alternativeResources);

            var ahpResult = calc.Calculate(_criteriaMatrix, criteriaPoints);
            var itemValue = ahpResult.First(x => x.Key.ID == item.Value.ID).Value;

            result += itemValue;
        }
        //CloudResource.Shared.Logger.Log(result.ToString());
        Console.WriteLine(result);
        return result;
    }
```

Şekil 4.46. Uygunluk fonksiyonu.

DCM.Decision.AHP katmanında bulunan Calculator sınıfı ve fonksiyonları Ek C'de verilmiştir. AHP'nin ihtiyaç duyduğu kriterler (criteriaPairList) ve alternatifler Calculate sınıfına girdi olarak verilir. criteriaPairList dizisi, içinde kriterleri ikili şekilde tutar ve bunların belirlenen değerlerini saklar. Veri tipi "List<AHPPair<Criteria>>" şeklinde belirlenmiştir. alternativeDict ise her bir kriter için kaynaklara verilen normalize edilmiş değerleri saklar. Veri tipi "Dictionary<Criteria, Dictionary<Resource, double>>" şeklinde belirlenmiştir.

4.3.4. Kaynak planlama modelinin değerlendirilmesi

Müşteri talepleri ile kaynakların eşleştirilebilmesi için müşteri odaklı bir servis planlama algoritması geliştirilmiştir. Bu algoritma bir servis olarak sunularak farklı kaynak tipleri için kullanılması mümkün olabilecektir.

Kaynak planlama algoritmasının çalışmasını gösterebilmek adına örnek bir müşteri öncelik tablosu oluşturulmuştur (Tablo 4.3.).

Tablo 4.3. Örnek bir müşteri öncelik tablosu

	Maliyet (C)	Kalite (Q)	Şirket Deneyimi (P)	Risk (R)	Erken Teslim (ED)	Geç Teslim (LD)
Maliyet (C)	1	5	9	3	9	0,111
Kalite (Q)	0,2	1	9	5	9	0,111
Şirket Önceliği (P)	0,111	0,111	1	0,111	0,111	0,111
Risk (R)	0,33	0,2	9	1	0,25	0,111
Erken Teslim (ED)	0,111	0,111	9	4	1	6
Geç Teslim (LD)	9	9	9	9	0,1667	1
Toplam	10,75556	15,42222	46	22.11111	19,52778	7,444444

Senaryoda 4 farklı şirketin hizmet verdiğini düşünelim (Tablo 4.4.). Bu 4 farklı şirketin C, Q, R, P, ED ve LD kriterlerine göre karşılaştırma matrislerini hesaplayalım (Tablo 4.5. ve Tablo 4.6.).

Tablo 4.4. Şirketlere ait değerler tablosu

Şirketler:	1	2	3	4
Maliyet (C)	1	9	3	7
Kalite (Q)	7	2	9	4
Şirket Önceliği (P)	1	1	1	1
Risk (R)	2	8	2	6
Erken Teslim (ED)	9	4	9	9
Geç Teslim (LD)	2	9	7	9

Tablo 4.5. Şirketlerin C, Q, R, P, ED, LD parametrelerine göre karşılaştırılması

C	1	2	3	4
1	1	9	3	7
2	0,11	1	0,33	0,77
3	0,33	3	1	2,33
4	0,14	1,28	0,42	1

Q	**1**	**2**	**3**	**4**
1	1	0,28	1,28	0,57
2	3,5	1	4,5	2
3	0,77	0,77	1	0,44
4	1,75	0,5	2,25	1

P	**1**	**2**	**3**	**4**
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1

R	1	2	3	4
1	1	4	1	3
2	0,25	1	0,25	0,75
3	1	4	1	3
4	0,33	1,33	0,33	1

ED	**1**	**2**	**3**	**4**
1	1	0,44	1	1
2	2,25	1	2,25	2,25
3	1	0,44	1	1
4	1	0,44	1	1

LD	**1**	**2**	**3**	**4**
1	1	4,5	3,5	4,5
2	0,22	1	0,78	1
3	0,28	1,28	1	1,28
4	0,22	1	0,77	1

Tablo 4.6. Her bir karşılaştırma matrisinin toplam değerlerinin hesaplanması

Cost					
Toplam	1,587302	14,28571	4,761905	11,11111	
	0,63	0,63	0,63	0,63	0,63
	0,07	0,07	0,07	0,07	0,07
	0,21	0,21	0,21	0,21	0,21
	0,09	0,09	0,09	0,09	0,09

Tablo 4.6. (Devamı)

Quality					
Toplam	7,027778	2,563492	9,035714	4,015873	
	0,142292	0,111455	0,142292	0,142292	0,134583
	0,498024	0,390093	0,498024	0,498024	0,471041
	0,110672	0,303406	0,110672	0,110672	0,158855
	0,249012	0,195046	0,249012	0,249012	0,235521
Preselect					
Toplam	4	4	4	4	
	0,25	0,25	0,25	0,25	0,25
	0,25	0,25	0,25	0,25	0,25
	0,25	0,25	0,25	0,25	0,25
	0,25	0,25	0,25	0,25	0,25
Risk					
Toplam	2,583333	10,33333	2,583333	7,75	
	0,387097	0,387097	0,387097	0,387097	0,387097
	0,096774	0,096774	0,096774	0,096774	0,096774
	0,387097	0,387097	0,387097	0,387097	0,387097
	0,129032	0,129032	0,129032	0,129032	0,129032
ED					
Toplam	5,25	2,333333	5,25	5,25	
	0,190476	0,190476	0,190476	0,190476	0,190476
	0,428571	0,428571	0,428571	0,428571	0,428571
	0,190476	0,190476	0,190476	0,190476	0,190476
	0,190476	0,190476	0,190476	0,190476	0,190476
LD					
Toplam	1,730159	7,785714	6,055556	7,785714	
	0,577982	0,577982	0,577982	0,577982	0,577982
	0,12844	0,12844	0,12844	0,12844	0,12844
	0,165138	0,165138	0,165138	0,165138	0,165138
	0,12844	0,12844	0,12844	0,12844	0,12844

Kullanıcıya ait olan kriterler Tablo 4.3.'den alınır ve Tablo 4.7.'de gösterilen ağırlıkları hesaplanır.

Tablo 4.7. Kriterlere ait hesaplanan ağırlık tablosu

Kriterler	**Ağırlık**
C	0,204
Q	0,163
P	0,011
R	0,052
ED	0,208
LD	0,361

Tablo 4.8. Önem puanlarının hesaplanması

	C	Q	P	R	ED	LD
1	0,63	0,13	0,25	0,39	0,19	0,58
2	0,07	0,47	0,25	0,10	0,43	0,13
3	0,21	0,16	0,25	0,39	0,19	0,17
4	0,09	0,24	0,25	0,13	0,19	0,13

X

Kriterler	Ağırlık
C	0,204
Q	0,164
P	0,011
R	0,052
ED	0,209
LD	0,361

=

0,421793654
0,234788567
0,191029948
0,152387832

4 farklı şirket teklifi için, Tablo 4.8.'de gösterildiği gibi önem puanları hesaplanmıştır. AHP uygunluk yüzdeleri Tablo 4.9.'da verilmektedir.

Tablo 4.9. 4 şirket için AHP ile hesaplanan uygunluk yüzdeleri

Şirket	AHP sonucu	Uygunluk yüzdesi
1	0,42	42
2	0,23	23
3	0,19	19
4	0,15	15

AHP yöntemiyle 1. şirket müşterinin öncelik matrisine daha uygun bulunmuştur. Müşteri öncelik matrisi incelendiğinde müşterinin C, Q, R, P, ED, LD parametreleri içerisinde daha öncelikli gördüğü parametreler sırasıyla şirket deneyimi (güven), erken teslim (ED), kalite, risk, maliyet ve geç teslim (LD)'dir.

Kaynak seçim algoritmasının daha kolay karşılaştırılabilmesi için sadece maliyet, kalite ve risk kriterlerini inceleyelim. 24 adet çeşitli özelliklerde 3 boyutlu yazıcı verisi test amaçlı oluşturulmuştur. 3d yazıcılar sisteme birer kaynak olarak tanıtılmıştır. Her bir 3 boyutlu yazıcı için boyut, malzeme, hassasiyet gibi yazıcı özellikleri sistemde tanımlanmıştır. Kaynakların fiziksel özelliklerinin dışında kaynakların birim maliyeti, kalitesi ve riski açısından modelin dikkate aldığı diğer parametreler test verisi içinde tanımlanmıştır. Bu veriler Tablo 4.10.'da görülmektedir.

Tablo 4.10. Test kaynak veri seti

ID	Type	P1	P2	P3	P4	P5		Cost	Quality	Risk
1	3d	300	180	180	ABS	0,05		50	9	5
2	3d	250	210	200	Metal	0,1		100	1	6
3	3d	305	305	610	Carbonfiber	0,03		80	5	4
4	3d	200	200	400	PLA	0,05		35	3	2
5	3d	140	140	140	ABS	0,05		30	6	6
6	3d	300	300	300	ABS	0,05		85	4	4
7	3d	400	400	400	ABS	0,05		95	7	8
8	3d	300	300	400	ABS	0,03		150	9	7
9	3d	340	340	600	PLA	0,1		40	5	5
10	3d	200	200	180	Carbonfiber	0,05		70	2	7
11	3d	100	100	100	Metal	0,05		85	1	3
12	3d	210	210	205	PLA	0,05		30	2	4
13	3d	300	180	180	ABS	0,05		60	3	2
14	3d	250	210	200	Metal	0,1		90	4	4
15	3d	305	305	610	Carbonfiber	0,03		90	7	1
16	3d	300	300	400	PLA	0,05		50	9	5
17	3d	140	140	140	ABS	0,05		40	5	6
18	3d	300	300	300	ABS	0,05		95	6	3
19	3d	400	400	400	ABS	0,05		100	4	7
20	3d	300	300	400	ABS	0,03		70	1	2
21	3d	340	340	600	PLA	0,1		50	5	4
22	3d	200	200	180	Carbonfiber	0,05		80	6	5
23	3d	500	500	500	Metal	0,03		150	9	4
24	3d	210	210	205	PLA	0,05		40	1	1

4 adet alt iş parçasına sahip bir test işi sisteme eklenmiştir. Bu iş parçalarının hepsi 3 boyutlu yazıcılarda işlenmesi gereken parçalardır. Her bir parçanın gereksinim duyduğu özellikler ayrı ayrı belirtilmiştir.Tablo 4.11.'de bu iş parçalarının gereksinim duyduğu kaynak özellikleri gösterilmektedir.

Tablo 4.11. İş parçaları-kaynak gereksinimi

ID		Property Key	Value		ID	Property Key	Value
TestJob							
	SubJob1				SubJob3		
		resourcetype	3d			resourcetype	3d
		x	206			x	302
		y	112			y	292
		z	233			z	222
		material	ABS			material	ABS
		presicion	0,05			presicion	0,05
	SubJob2				SubJob4		
		resourcetype	3d			resourcetype	3d
		x	197			x	160
		y	148			y	332
		z	210			z	293
		material	ABS			material	ABS
		presicion	0,05			presicion	0,1

Kaynaklar arasından seçim yapılırken sisteme yüklenen işin gereksinimleri göz önünde bulundurulurken ayrıca müşterinin öncelikleri de dikkate alınmaktadır. Bu

öncelikler maliyet, kalite, risk ve zaman olarak bu platformda belirlenmiştir. Bu örnekte 2 farklı ağırlık matrisi verilerek farklı önceliklere göre farklı kaynakların seçildiği gösterilmektedir. Tablo 4.12.'de gösterilen birinci ağırlık matrisinde kalitenin diğerlerinden çok daha önemli olduğu ifade edilmektedir. Tablo 4.13.'da gösterilen ikinci ağırlık matrisinde ise maliyetin diğerlerinde çok daha önemli olduğu ifade edilmektedir. Bu öncelik matrislerinin her biri için algoritma çalıştırıldığında birinci durumda uygun kaynaklar arasından kalite parametresi en yüksek olan, ikinci durumda ise maliyeti en düşük olan kaynaklar seçilecektir.

Tablo 4.12. Test1-Ağırlık matrisi

	Cost	Quality	Risk
Cost	1	0,1111111	0,1111111
Quality	9	1	9
Risk	9,00	0,1111111	1

Tablo 4.13. Test2-Ağırlık matrisi

	Cost	Quality	Risk
Cost	1	9	9
Quality	0,111111111	1	9
Risk	0,11	0,111111111	1

Test için oluşturulan iş sisteme yüklendiğinde tüm iş parçaları için uygun kaynaklar araştırılır. Alt iş parçalarını sırasıyla j1, j2, j3 ve j4 olarak isimlendirelim. Her bir iş parçasının işlenebileceği kaynaklar aşağıdaki gibi listelenmiştir.

j1 => 6, 7, 8, 18, 19, 20

j2 => 6, 7, 8, 18, 19, 20

j3 => 7, 19

j4 => 7, 19

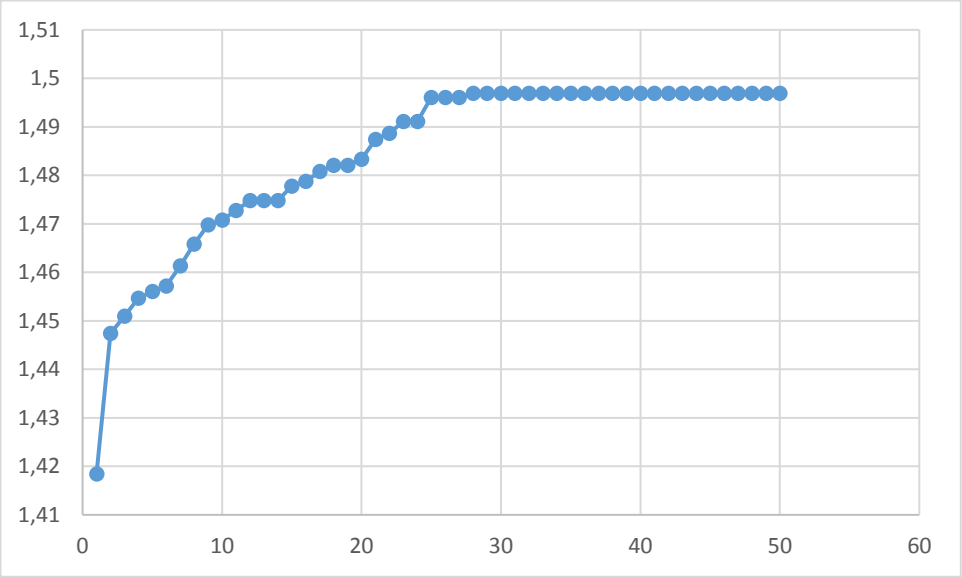
Kalitenin diğerlerinden çok daha önemli olduğu birinci durum için algoritma çalıştırıldığında çözüm olarak j1 ve j2 için 8 numaralı kaynak kullanılması, j3 ve j4 için ise 7 numaralı kaynak kullanılması önerilmektedir.

Maliyetin diğerlerinden çok daha önemli olduğu ikinci durum için algoritma çalıştırıldığında ise çözüm olarak j1 ve j2 için 20 numaralı kaynak kullanılması, j3 ve j4 için ise 7 numaralı kaynak kullanılması önerilmektedir. Çözümlerin karşılaştırılması Tablo 4.14.'da görülmektedir.

Tablo 4.14. Test1-Test2 çözümlerinin karşılaştırılması

	Test 1 (Kalite Öncelikli)	**Test 2 (Maliyet Öncelikli)**
Çözüm	j1=>8, j2=>8, j3=>7, j4=>7	j1=>20, j2=>20, j3=>7, j4=>7
Maliyet (en iyi: min)	150+150+95+95 = 490	70+70+95+95=330
Kalite (en iyi: max)	9+9+7+7=32	1+1+7+7=16
Risk (en iyi: max)	7+7+8+8=30	2+2+8+8=20

Genetik algoritma 50 kez çalıştırılmış ve iterasyon başına elde edilen ortalama uygunluk fonksiyon değerleri Şekil 4.47.'da verilmiştir. 4 adet alt iş parçası ile gerçekleştirilen bu örnekte uygunluk değeri ortalaması için 1. iterasyonda 1,42 değeri ve 23. iterasyonda ise 1,49 değerine ulaşılmıştır. 50. iterasyona kadar genetik algoritma çalıştırılmış ve 1,49 değerini veren sıralama çözüm olarak kabul edilmiştir.



Şekil 4.47. GA ortalama uygunluk fonksiyonu

BÖLÜM 5. SONUÇ VE ÖNERİLER

Bulut imalat yetenekleri ile kaynak sahipleri ve kaynak arayan kullanıcılar, kaynakları platformlar aracılığıyla birbirleri ile kolayca paylaşabilirler. Literatürde sunulan bulut imalat çalışmaları, her iki tarafın bir araya getirilebilmesi için karmaşık yapıda olan imalat bileşenlerinin paylaşılabilir ve her yerden erişilebilir olması için çeşitli modeller önermişlerdir. Bu tez çalışmasında, literatürde var olan bulut imalat modellerinin sunduğu kaynakların sanallaştırması, imalat servislerinin entegrasyonu, kaynak planlaması, servis seçimi gibi bileşenlerin bir servis olarak nasıl sunulabileceğini gösteren bulut imalat modeli tasarımı önerilmiştir. Bulut imalat sistemlerinin merkezileşmeden dolayı getirdiği problemler ortaya konulmuş olup buna çözüm olarak DCMApp adı ile blokzinciri (BC) destekli merkezi olmayan bir bulut imalat uygulaması geliştirilmiştir. DCMApp, kullanıcıların üçüncü bir tarafa ihtiyaç duymadan kendi aralarında anlaşma yapmalarına imkân vermektedir. Geliştirilen uygulamaya ayrıca bir kaynak seçim önerisinde bulunabilecek bir karar destek sistemi yerleştirilerek uygulamanın yeni bileşenlerle geliştirilebileceği gösterilmiştir.

Bulut imalat platformlarının güvenilirliği kullanıcılar için bir soru işareti olarak görülmektedir. Bu çalışmanın özgünlüğü, imalat anlaşmalarının halka açık/genel bir BC ağı olan ve akıllı sözleşmeleri (SC) destekleyen Ethereum ağı üzerinde gerçekleştirilmiş olmasıdır. Uygulama, diğer BC destekli bulut imalat uygulamalarından farklı olarak veri saklama anlamında hibrit bir yapıda tasarlanmıştır. Veri, hem BC ortamında hem de kullanıcı tarafında hibrit bir şekilde saklanarak maliyet azaltılmıştır. Kullanıcı verileri yerel veritabanında depolanmakta ve anlaşmaya ait önemli veriler BC ağında depolanmaktadır. Ethereum ağı üzerinden herhangi bir noktadan anlaşmalara erişim mümkündür. Bu nedenle merkezi bir noktaya bağımlı olmadığı için uygulama herhangi bir kesintiden etkilenmeyecektir.

Uygulamanın hibrit yapısı sayesinde kullanıcılar yalnızca güvence altına alınması gereken sözleşmeler için ödeme yapar. Uygulama tamamen halka açık bir yapıda çalıştırılması durumunda, tüm bilgilerin halka açık bir şekilde depolanmasını gerektirir ki bu da ciddi bir maliyete neden olacaktır. Bütün bunların dışında, özel ağlarda ağın bir kişi veya bir kuruluş tarafından kontrol edilmesi kullanıcılar için ciddi sorunlar doğurmaktadır. Ayrıca, sunucu altyapısı özel ağlarda bir kişi/kurum tarafından hazırlanması gerekir. Bu sebeplerden dolayı hibrit yapı tercih edilmiştir. Bu model ile birlikte herhangi bir aracıya ihtiyaç duyulmadan, herhangi bir yatırım yapmadan, herhangi bir komisyon ücreti ödemeden, sürdürülebilirlik, bakım gibi maliyetleri ödemeden kullanıcılar birbirleriyle imalat anlaşmalarını yapabilecektir.

DCMApp örneğindeki gibi herkese açık BC ağı ile geliştirilmiş olan bulut imalat platformu, altyapı maliyetleri açısından özel BC ağı ile geliştirilen bulut imalat platformları ve merkezi bulut imalat platformları ile kıyaslanmıştır. Merkezi ve özel BC ağları ile geliştirilen platformlarda uygulamanın çalışır halde olması için sürekli olarak bir sunucu gideri olduğu görülmüştür. Yüksek miktarda verinin saklanması gerektiğinde veya yüksek hacimde işlem yapılmak istendiğinde merkezi ve özel BC ağları ile geliştirilen çözümlerin daha avantajlı olduğu ancak güvenlik açısından riskli olduğu görülmüştür.

Geliştirilen modelin öne çıkan özellikleri değerlendirilmiştir. Halka açık BC ağları kullanıldığından herhangi bir altyapı yatırımına ihtiyaç duyulmaz. BC'nin doğası gereği verinin değiştirilemez olması ve işlemlerin geri döndürülememesi uygulamayı daha güvenli hale getirmektedir. Uygulamadaki aracılık görevini BC ağı üstlendiği için üçüncü bir tarafın aracılığına ihtiyaç duymaz. BC teknolojisi ile geliştirilen kripto paralar sayesinde değer transferi güvenli bir şekilde yapılabildiği için ödemeler doğrudan taraflar arasında SC aracılığı ile yapılabilmektedir. SC'lerde tanımlanan koşulların değiştirilememesi ve tarafların bu koşulları kabul ederek ağda işlem yapması, kullanıcıların yaptığı anlaşmaları güvence altına almıştır. Sözleşmelerde belirlenen koşullar sağlandığında gerçekleşecek olan işlemler herhangi bir takibe ihtiyaç duyulmadan icra edilecek olması ve asla iptal edilememesi güvenilirliğini önemli ölçüde arttırmaktadır. BC ağı üzerinden işlem yapılması uygulamanın

sürdürülebilir olmasını sağlamıştır. Uygulamanın çalışması için dağıtık bir yapıda olan BC ağının bir noktasına erişim sağlaması yeterli olacaktır. Tasarlanan DCMApp'in merkezi bulut imalat platformlarından; sürdürülebilirlik, güvenilirlik, maliyet, veri güvenliği alanlarında üstün olduğu açıkça görülmektedir.

DCMApp içinde bir kaynak seçim bileşeni geliştirilmiştir. Bu bir karar destek sistemi olarak rol almakta ve kullanıcıların hangi işi hangi kaynakta yaptırabileceği konusunda öneri sunmaktadır. Kullanıcıların belirledikleri önceliklere göre karar verilmektedir. Geliştirilen algoritma, AHP ve genetik algoritmayı birlikte kullanarak bir çözüm önermektedir. Algoritmanın çalışmasını değerlendirebilmek için 24 adet çeşitli özelliklerde 3 boyutlu yazıcı verisi test amaçlı oluşturulmuştur. Test verisi iş parçalarının her biri 3 boyutlu yazıcılarda işlenmesi gereken 4 adet iş parçasına sahip bir iştir. Test işi sisteme yüklendiğinde tüm iş parçaları için uygun kaynaklar araştırılır. 2 farklı müşteri öncelik matrisi temel alınarak algoritma çalıştırılmış ve öngörülen sonuçları verdiği gözlemlenmiştir. Bulut imalat platformlarındaki kaynak çeşitliliğinin ve kaynak sayısının çok olması, kullanıcının istediği önceliklere göre karar vermesini zorlaştırmaktadır. Geliştirilen karar destek bileşeni ile kullanıcılar, sayısı ve çeşidi fazla olan teklifler arasından hızlı ve doğru seçim yapabilmeleri için desteklenmiştir.

Bu tez çalışması kapsamında, bulut imalatta BC teknolojisinin kullanımı örnek bir uygulama ile açıklanmaya çalışılmıştır. BC ağının SC'lar ile bir üretim uygulamasının nasıl çalıştırıldığı gösterilmektedir. Bu özellikleri ile tez çalışmasının gelecekteki çalışmalara ışık tutması amaçlanmıştır. Gelecekte bu tez çalışması kapsamında geliştirilen DCMApp içinde kullanılan SC'lar geliştirilebilir ve DCMApp dışında başka uygulamalar bu sözleşmeler aracılığı ile sisteme entegre olabilir. DCMApp'in sağladığı altyapı üzerine bu çalışmada gerçekleştirilen kaynak seçim algoritması bileşeni gibi başka bileşenler eklenerek uygulama zenginleştirilebilir.

Bu uygulamada ayrıntılı bilgiler merkezi bir depoda saklanır ve müşterinin yerel veritabanı ile senkronize edilir. Gelecekte bu verilerin IPFS'de depolanması veya diğer bulut depolama çözümlerinde saklanması önerilmektedir.

KAYNAKLAR

[1] D. Wu, "Cloud-based design and manufacturing: a network perspective," Georgia Institute of Technology, 2014.

[2] F. Tao, Y. Cheng, L. Zhang, Y. L. Luo, and L. Ren, "Cloud Manufacturing," *Adv. Mater. Res.*, vol. 201–203, pp. 672–676, 2011.

[3] F. Tao, L. Zhang, V. C. Venkatesh, Y. Luo, and Y. Cheng, "Cloud manufacturing: a computing and service-oriented manufacturing model," *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 225, pp. 1969–1976, 2011.

[4] E. Ada, Y. Kazançoğlu, and C. Tayaksi, "Bulut Üretim: İşlemler Yönetiminde Yeni Bir Bulut Bilişim Modeli," *Ege Akad. Bakis (Ege Acad. Rev.*, vol. 16, no. OZEL, 2016.

[5] U. Bodkhe *et al.*, "Blockchain for Industry 4.0: A Comprehensive Review," *IEEE Access*, vol. 8, pp. 79764–79800, 2020.

[6] CoinMarketCap,"Ethereum" https://coinmarketcap.com/currencies/ethereum/ Erişim Tarihi: 01.12.2020.

[7] Y. Cao, S. Wang, L. Kang, and Y. Gao, "A TQCS-based service selection and scheduling strategy in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 82, no. 1–4, pp. 235–251, 2016.

[8] F. Tao, Y. Hu, D. Zhao, Z. Zhou, H. Zhang, and Z. Lei, "Study on manufacturing grid resource service QoS modeling and evaluation," *Int. J. Adv. Manuf. Technol.*, vol. 41, no. 9–10, pp. 1034–1042, 2009.

[9] F. Tao, D. Zhao, and L. Zhang, "Resource service optimal-selection based on intuitionistic fuzzy set and non-functionality QoS in manufacturing grid system," *Knowl. Inf. Syst.*, vol. 25, no. 1, pp. 185–208, 2010.

[10] B. Huang, C. Li, and F. Tao, "A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system," *Enterp. Inf. Syst.*, vol. 8, no. 4, pp. 445–463, 2014.

[11] F. Xiang, Y. Hu, Y. Yu, and H. Wu, "QoS and energy consumption aware service composition and optimal-selection based on Pareto group leader algorithm in cloud manufacturing system," *Cent. Eur. J. Oper. Res.*, vol. 22, no. 4, pp. 663–685, 2014.

[12] Z. G. Chen, "Research of Cloud Manufacturing Execution Path Optimization Based on Adaptive Ant Colony Algorithm on Hadoop Platform," *Appl. Mech. Mater.*, vol. 628, pp. 417–420, 2014.

[13] X. Xie, L. Liang, and Y. Cao, “Trust model based on feedback evaluation in cloud manufacturing environment,” *Adv. Mater. Res.*, vol. 308–310, pp. 1740–1745, 2011.

[14] F. Macia-perez, J. V. Berna-martinez, D. Marcos-jorquera, I. Lorenzo-fonseca, and A. Ferrandiz-colmeiro, “Cloud Agile Manufacturing SIMaaS restructured PraaS,” vol. 2, no. 5, pp. 1045–1048, 2012.

[15] C. S. Hu, C. D. Xu, X. B. Cao, and J. C. Fu, “Study of Classification and Modeling of Virtual Resources in Cloud Manufacturing,” *Appl. Mech. Mater.*, vol. 121–126, pp. 2274–2280, 2011.

[16] J. Mai, L. Zhang, F. Tao, and L. Ren, “Customized production based on distributed 3D printing services in cloud manufacturing,” *Int. J. Adv. Manuf. Technol.*, vol. 84, no. 1–4, pp. 71–83, 2016.

[17] G. Adamson, L. Wang, M. Holm, and P. Moore, “Cloud manufacturing – a critical review of recent development and future trends,” *Int. J. Comput. Integr. Manuf.*, pp. 1–34, 2015.

[18] O. Fisher, N. Watson, L. Porcu, D. Bacon, M. Rigley, and R. L. Gomes, “Cloud manufacturing as a sustainable process manufacturing route,” *J. Manuf. Syst.*, vol. 47, pp. 53–68, 2018.

[19] D. Wu, D. W. Rosen, L. Wang, and D. Schaefer, “Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation,” *Comput. Des.*, vol. 59, pp. 1–14, 2015.

[20] F. Guillemot and C. Zimmer, “From Cradle to Grave: The Multiple Roles of Fibroblast Growth Factors in Neural Development,” *Neuron*, vol. 71, no. 4, pp. 574–588, 2011.

[21] B. Hayes, “Cloud computing,” *Commun. ACM*, vol. 51, no. 7, pp. 9–11, 2008.

[22] P. Mell and T. Grance, “The NIST-National Institute of Standars and Technology- Definition of Cloud Computing,” *NIST Spec. Publ. 800-145*, p. 7, 2011.

[23] T. Dillon, C. Wu, and E. Chang, “Cloud Computing: Issues and Challenges,” in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 27–33.

[24] Amazon, “Bulut bilişim nedir?,” https://aws.amazon.com/tr/what-is-cloud-computing/. Erişim Tarihi: 01.12.2020.

[25] X. Wang and N. Xu, “Virtualise manufacturing capabilities in the cloud: requirements, architecture and implementation,” *Int. J. Manuf. Res.*, vol. 9, no. 4, p. 348, 2014.

[26] N. Liu and X. Li, “A Resource Virtualization Mechanism for Cloud Manufacturing Systems,” vol. 122, M. van Sinderen, P. Johnson, X. Xu, and G. Doumeingts, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 46–59.

[27] L. Tan and N. Wang, “Future Internet: The Internet of Things,” in *3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, 2010, pp. 376–380.

[28] S. Li, L. Da Xu, and S. Zhao, “The internet of things: a survey,” *Inf. Syst. Front.*, vol. 17, no. 2, pp. 243–259, 2015.

[29] H. Jiang, S. Zhao, Y. Zhang, and Y. Chen, “The cooperative effect between technology standardization and industrial technology innovation based on Newtonian mechanics,” *Inf. Technol. Manag.*, vol. 13, no. 4, pp. 251–262, 2012.

[30] H. Jiang, S. Zhao, S. Qiu, and Y. Chen, “Strategy for technology standardization based on the theory of entropy,” *Inf. Technol. Manag.*, vol. 13, no. 4, pp. 311–320, 2012.

[31] H. Jiang, S. Zhao, X. Wang, and Z. Bi, “Applying Electromagnetic Field Theory to Study the Synergistic Relationships Between Technology Standardization and Technology Development,” *Syst. Res. Behav. Sci.*, vol. 30, no. 3, pp. 272–286, 2013.

[32] R. (Raj) Rajkumar, I. Lee, L. Sha, and J. Stankovic, “Cyber-physical systems,” in *Proceedings of the 47th Design Automation Conference on - DAC ’10*, 2010, p. 731.

[33] D. Georgakopoulos and M. P. Papazoglou, *Service-Oriented Computing*. 2008.

[34] Z. M. Aljazzaf, M. A. M. Capretz, and M. Perry, “Trust-based Service-Oriented Architecture,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 28, no. 4, pp. 470–480, 2016.

[35] A. A. Borse, V. Siddhant, S. Babu, and G. N. Kumar, “Architecture paradigm for Business Intelligence: A survey,” *Int. J. Adv. Res. Ideas Innov. Technol.*, vol. 5, no. 1, pp. 1–4, 2019.

[36] Y. Liu, L. Wang, X. V. Wang, X. Xu, and P. Jiang, “Cloud manufacturing: key issues and future perspectives,” *Int. J. Comput. Integr. Manuf.*, vol. 32, no. 9, pp. 858–874, 2019.

[37] B. Kaynak, S. Kaynak, and O. Uygun, “Cloud Manufacturing Architecture Based on Public Blockchain Technology,” *IEEE Access*, vol. 8, pp. 2163–2177, 2020.

[38] D. Contractor and D. Patel, “Accountability in cloud computing by means of chain of trust,” *Int. J. Netw. Secur.*, vol. 19, no. 2, pp. 251–259, 2017.

[39] M. Hamdaqa and L. Tahvildari, “Cloud Computing Uncovered: A Research Landscape,” in *Advances in Computers*, vol. 86, 2012, pp. 41–85.

[40] G. Škulj, R. Vrabič, P. Butala, and A. Sluga, “Decentralised network architecture for cloud manufacturing,” *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 4–5, pp. 1–14, 2015.

[41] A. V. Barenji, H. Guo, Z. Tian, Z. Li, W. M. Wang, and G. Q. Huang, “Blockchain-based cloud manufacturing: Decentralization,” *Adv. Transdiscipl. Eng.*, vol. 7, pp. 1003–1011, 2018.

[42] M. Jammal, H. Hawilo, A. Kanso, and A. Shami, “Mitigating the Risk of Cloud Services Downtime Using Live Migration and High Availability-Aware Placement,” in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2016, vol. 0, pp. 578–583.

[43] B. H. Li *et al.*, “Cloud manufacturing: A new service-oriented networked manufacturing model,” *Jisuanji Jicheng Zhizao Xitong/Computer Integr. Manuf. Syst. CIMS*, vol. 16, no. 1, 2010.

[44] L. Zhang *et al.*, “Cloud manufacturing: a new manufacturing paradigm,” *Enterp. Inf. Syst.*, vol. 8, no. 2, pp. 167–187, 2012.

[45] L. Ren, L. Zhang, F. Tao, C. Zhao, X. Chai, and X. Zhao, “Cloud manufacturing: from concept to practice,” *Enterp. Inf. Syst.*, vol. 9, no. 2, pp. 186–209, 2015.

[46] X. Xu, “From cloud computing to cloud manufacturing,” *Robot. Comput. Integr. Manuf.*, vol. 28, no. 1, pp. 75–86, 2012.

[47] D. Wu, D. Schaefer, and D. W. Rosen, “Cloud-based design and manufacturing systems: A social network analysis,” in *Proceedings of the International Conference on Engineering Design, ICED*, 2013, vol. 7 DS75-07, pp. 149–158.

[48] L. Guo, “A system design method for cloud manufacturing application system,” *Int. J. Adv. Manuf. Technol.*, vol. 84, no. 1–4, pp. 275–289, 2016.

[49] L. Thames and D. Schaefer, “Software-defined Cloud Manufacturing for Industry 4.0,” *Procedia CIRP*, vol. 52, pp. 12–17, 2016.

[50] Y. Liu and X. Xu, “Industry 4.0 and Cloud Manufacturing: A Comparative Analysis,” *J. Manuf. Sci. Eng.*, vol. 139, no. 3, pp. 1–8, 2017.

[51] Fei Tao, Ying Cheng, Li Da Xu, Lin Zhang, and Bo Hu Li, “CCIoT-CMfg: Cloud Computing and Internet of Things-Based Cloud Manufacturing Service System,” *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1435–1442, 2014.

[52] P. Butala and A. Sluga, “Autonomous Work Systems in Manufacturing Networks,” *CIRP Ann.*, vol. 55, no. 1, pp. 521–524, 2006.

[53] C. Yu, L. Zhang, W. Zhao, and S. Zhang, “A blockchain-based service composition architecture in cloud manufacturing,” *Int. J. Comput. Integr. Manuf.*, vol. 00, no. 00, pp. 1–15, 2019.

[54] Z. Li, A. V. Barenji, and G. Q. Huang, “Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform,” *Robot. Comput. Integr. Manuf.*, vol. 54, pp. 133–144, 2018.

[55] J. Lee, M. Azamfar, and J. Singh, “A blockchain enabled Cyber-Physical System architecture for Industry 4.0 manufacturing systems,” *Manuf. Lett.*, vol. 20, pp. 34–39, 2019.

[56] LI Bo-hu *et al.*, “Further discussion on cloud manufacturing,” *Comput. Integr. Manuf. Syst.*, vol. 3, 2011.

[57] Z. Cheng, D. Zhan, X. Zhao, and H. Wan, "Multitask Oriented Virtual Resource Integration and Optimal Scheduling in Cloud Manufacturing," *J. Appl. Math.*, vol. 2014, no. 1, pp. 1–9, 2014.

[58] S. Wang, L. Guo, L. Kang, C. Li, X. Li, and Y. M. Stephane, "Research on selection strategy of machining equipment in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 71, no. 9–12, pp. 1549–1563, 2014.

[59] Sisi Tian, Q. Liu, W. Xu, and J. Yan, "A Discrete Hybrid Bees Algorithm for Service Aggregation Optimal Selection in Cloud Manufacturing," in *International Conference on Intelligent Data Engineering and Automated Learning*, 2013, pp. 110–117.

[60] L. Wang, S. Guo, X. Li, B. Du, and W. Xu, "Distributed manufacturing resource selection strategy in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 9–12, pp. 3375–3388, 2018.

[61] J. Zhou and X. Yao, "A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition," *Int. J. Adv. Manuf. Technol.*, vol. 88, no. 9–12, pp. 3371–3387, 2017.

[62] Y. Que, W. Zhong, H. Chen, X. Chen, and X. Ji, "Improved adaptive immune genetic algorithm for optimal QoS-aware service composition selection in cloud manufacturing," *Int. J. Adv. Manuf. Technol.*, vol. 96, no. 9–12, pp. 4455–4465, 2018.

[63] W. Li, Y. Zhong, X. Wang, and Y. Cao, "Resource virtualization and service selection in cloud logistics," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1696–1704, 2013.

[64] J. Zhou, X. Yao, Y. Lin, F. T. S. Chan, and Y. Li, "An adaptive multi-population differential artificial bee colony algorithm for many-objective service composition in cloud manufacturing," *Inf. Sci. (Ny).*, vol. 456, pp. 50–82, 2018.

[65] B. M. Lavanya, "Blockchain Technology Beyond Bitcoin: An Overview," *Int. J. Comput. Sci. Mob. Appl.*, vol. 6, no. 1, pp. 76–80, 2018.

[66] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Cryptogr. Mail. List https//metzdowd.com*, 2009.

[67] M. Crosby, Nachiappan, P. Pattanayak, S. Verma, and V. Kalyanaraman, "BlockChain Technology: Beyond Bitcoin," *AIR Appl. Innov. Rev.*, no. 2, 2016.

[68] H. Aşan and H. Avunduk, "Blok Zinciri (Blockchain) Teknolojisi ve İşletme Uygulamaları: Genel Bir Değerlendirme," *Dokuz Eylul Univ. Iktis. ve Idari Bilim. Derg.*, vol. 33, no. 1, pp. 369–384, 2018.

[69] Z. Durğay and E. Karaarslan, "Blokzinciri Teknolojisinin E-Devlet Uygulamalarında Kullanımı: Ön İnceleme," in *Akademik Bilişim 2018*, 2018, p.7.

[70] J. L. Zhao, S. Fan, and J. Yan, "Overview of business innovations and research opportunities in blockchain and introduction to the special issue," *Financ. Innov.*, vol. 2, no. 1, p. 28, 2016.

[71] G. O. Karame, E. Androulaki, and S. Capkun, “Double-spending fast payments in bitcoin,” in *Proceedings of the 2012 ACM conference on Computer and communications security - CCS ’12*, 2012, p. 906.

[72] G. O. Karame, E. Androulaki, and S. Capkun, “Two Bitcoins at the Price of One?Double-spending fast payments in bitcoin,” in *IACR Cryptol. ePrint Arch*, 2012, p. 248.

[73] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, “Decentralized Applications: The Blockchain-Empowered Software System,” *IEEE Access*, vol. 6, pp. 53019–53033, 2018.

[74] S. Kardaş, “Blokzincir Teknolojisi: Uzlaşma Protokolleri,” *DÜMF Mühendislik Derg.*, vol. 10, no. 2, pp. 481–496, 2019.

[75] M. Tanrıverdi, M. Uysal, and M. T. Üstündağ, “Blokzinciri Teknolojisi Nedir ? Ne Değildir ?: Alanyazın İncelemesi,” *Bilişim Teknol. Derg.*, no. July, pp. 203–217, Jul. 2019.

[76] L. M. Bach, B. Mihaljevic, and M. ga. Za, “Comparative Analysis of Blockchain Consensus Algorithms,” in *MIPRO 2018*, 2018, pp. 1545–1550.

[77] S. Akleylek and K. Seyhan, “Blok Zinciri Bileşenleri ve Uygulamaları Üzerine Bir Derleme,” in *İnformasiya təhlükəsizliyinin aktual multidissiplinar elmi-praktiki problemləri IV respublika konfransının materialları*, 2018, pp. 27–37.

[78] N. Szabo, “Formalizing and securing relationships on public networks,” 1996. https://firstmonday.org/ojs/index.php/fm/article/download/548/469 Erişim Tarihi: 01.12.2020.

[79] A. Bahga and V. Madisetti, “Blockchain Platform for Industrial Internet of Things,” *J. Softw. Eng. Appl.*, vol. 09, pp. 533–546, 2016.

[80] P. Wang, R. X. Gao, and Z. Fan, “Cloud Computing for Cloud Manufacturing: Benefits and Limitations,” *J. Manuf. Sci. Eng.*, vol. 137, no. 4, 2015.

[81] M. Armbrust *et al.*, “A view of cloud computing,” *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[82] S. Wang, Z. Zhu, and L. Kang, “Resource allocation model in cloud manufacturing,” *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.*, vol. 230, no. 10, pp. 1726–1741, 2016.

EKLER

EK A: RSC Solidity Kodu

```
pragma solidity ^0.7.0;

contract Resource {
    address owner;
    string name;
    string properties;
    bool isActive;
    uint256 public jobCount;
    uint256 public offerCount;
    uint256 customerPoint;

    constructor(string memory propJson, string memory resourceName) {
        owner = msg.sender;
        properties = propJson;
        name = resourceName;
    }

    //Jobs
    mapping(address => bool) public jobs;
    event OfferRequest(address indexed jobContractAddress);
    address[] public waitingJobs;

    function requestOffer(address jobContractAddress) public {
        waitingJobs.push(jobContractAddress);
        offerCount++;
        emit OfferRequest(jobContractAddress);
    }

    function finishJob(address jobContractAddress) public {
        jobCount++;
        uint256 toDelete = find(jobContractAddress);
        delete waitingJobs[toDelete];
    }

    function find(address value) private view returns (uint256) {
```

```
        uint256 i = 0;
        while (waitingJobs[i] != value) {
            i++;
        }
        return i;
    }
}
```

EK B: JSC Solidity Kodu

```
pragma solidity ^0.7.0;
pragma experimental ABIEncoderV2;

contract Job {
    enum JobStatus {
        Created,
        OffersPending,
        JobAssigned,
        Sent,
        Confirmed,
        Deleted
    }
    enum OfferStatus {OfferRequested, OfferResponded}

    struct Offer {
        //Offerer's wallet address
        address offerer;
        //Date the offer request was created
        uint256 requestDate;
        //Date the offer was responded
        uint256 responseDate;
        //Status of the offer
        OfferStatus enumOfferStatus;
        //The date on which the offer undertakes to complete the job
        uint256 finishDate;
        //Price quotation for this job
        uint256 price;
        //Unique value generated for the offer
        uint256 id;
        //The resource address of the offer
        address resourceAddress;
        //Whether the offer has been accepted
        bool accept;
```

```
}

JobStatus enumJobStatus;
address contractOwner;
uint256 createDate;
uint256 dueDate;

uint256 resourceTypeID;
string propertiesJson;
string designUrl;

address payable acceptedOfferer;
uint256 acceptDate;
uint256 confimationMaxTime;
uint256 confimationExtendTime;
uint256 confirmDate;

string trackingCode;
uint256 sentDate;

bytes1 constant maxExtendCount = 0x03;
bytes1 extendCount;

mapping(address => mapping(address => Offer)) public offers;
uint256 offerCounter;

constructor(
    uint256 _dueDate,
    uint256 _resourceTypeID,
    string memory _propertiesJson,
    string memory _designUrl,
    uint256 _confimationMaxTime,
    uint256 _confimationExtendTime
) {
    dueDate = _dueDate;
    resourceTypeID = _resourceTypeID;
    propertiesJson = _propertiesJson;
    designUrl = _designUrl;
    confimationMaxTime = _confimationMaxTime;
    confimationExtendTime = _confimationExtendTime;
    contractOwner = msg.sender;
    createDate = block.timestamp;
    offerCounter = 0;
    enumJobStatus = JobStatus.Created;
}

function requestOffer(address offerer, address resourceAddress)
```

```
        public
        returns (uint256)
    {
        Offer memory newOffer = Offer({
            offerer: offerer,
            id: offerCounter + 1,
            requestDate: block.timestamp,
            finishDate: 0,
            price: 0,
            resourceAddress: resourceAddress,
            accept: false,
            enumOfferStatus: OfferStatus.OfferRequested,
            responseDate: 0
        });
        offers[offerer][resourceAddress] = newOffer;
        offerCounter++;
        if (enumJobStatus == JobStatus.Created) {
            enumJobStatus = JobStatus.OffersPending;
        }
        return offerCounter;
    }

    function responseOffer(
        bool accept,
        uint256 price,
        uint256 finishDate,
        address resource
    ) public returns (bool) {
        require(offers[msg.sender][resource].id > 0);
        require(enumJobStatus == JobStatus.OffersPending);

        offers[msg.sender][resource].accept = accept;
        offers[msg.sender][resource].price = price;
        offers[msg.sender][resource].finishDate = finishDate;
        offers[msg.sender][resource].responseDate = block.timestamp;
        offers[msg.sender][resource].enumOfferStatus = OfferStatus.Offe
rResponded;
        return true;
    }

    function acceptOffer(address payable offerer, address resource)
        public
        payable
        returns (bool)
    {
        require(contractOwner == msg.sender);
        require(enumJobStatus == JobStatus.OffersPending);
```

```
        require(offers[offerer][resource].price <= msg.value);
        acceptedOfferer = offerer;
        enumJobStatus = JobStatus.JobAssigned;
        acceptDate = block.timestamp;
        return true;
    }

    function deleteJob() public payable returns (bool) {
        require(contractOwner == msg.sender);
        require(
            (enumJobStatus == JobStatus.OffersPending) ||
                (enumJobStatus == JobStatus.Created)
        );
        enumJobStatus = JobStatus.Deleted;
        return true;
    }

    function sendFinishedJob(string memory _trackingCode)
        public
        payable
        returns (bool)
    {
        require(acceptedOfferer == msg.sender);
        require(enumJobStatus == JobStatus.JobAssigned);
        trackingCode = _trackingCode;
        enumJobStatus = JobStatus.Sent;
        sentDate = block.timestamp;
        return true;
    }

    function confirm() public returns (bool) {
        require(acceptedOfferer == msg.sender);
        require(enumJobStatus == JobStatus.Sent);
        confirmDate = block.timestamp;
        enumJobStatus = JobStatus.Confirmed;
        acceptedOfferer.transfer(address(this).balance);
        return true;
    }

    function requestConfirmTimeExtend() public returns (bytes1, uint256
) {
        require(contractOwner == msg.sender);
        require(enumJobStatus == JobStatus.Sent);
        require(extendCount < maxExtendCount);
        confimationMaxTime += confimationExtendTime;
        if (extendCount == 0x00) extendCount = 0x01;
        if (extendCount == 0x01) extendCount = 0x02;
```

```
        if (extendCount == 0x02) extendCount = 0x03;
        return (extendCount, confimationMaxTime);
    }

    function offerCount() public view returns (uint256) {
        return offerCounter;
    }

    function withdraw() public returns (bool) {
        require(acceptedOfferer == msg.sender);
        require((enumJobStatus == JobStatus.Sent));
        require((block.timestamp > sentDate + confimationMaxTime));
        return msg.sender.send(address(this).balance);
    }

    function getJobDetails()
        public
        view
        returns (
            uint256 _dueDate,
            uint256 _resourceTypeID,
            string memory _propertiesJson,
            string memory _designUrl
        )
    {
        return (dueDate, resourceTypeID, propertiesJson, designUrl);
    }

    function getJobState() public view returns (JobStatus) {
        return enumJobStatus;
    }

    function isMyOfferAccepted() public view returns (bool) {
        require((enumJobStatus == JobStatus.JobAssigned));
        if (acceptedOfferer == msg.sender) {
            return true;
        } else {
            return false;
        }
    }
}
```

EK C: DCM.Decision.AHP Calculator Sınıfı C# Kodu

```
using DCM.Decision.Shared;
using DCM.Decision.Shared.AHP;
using MathNet.Numerics.LinearAlgebra;
using System.Collections.Generic;
using System.Linq;

namespace DCM.Decision.AHP
{
    public class Calculator
    {
        public Dictionary<Resource, double> Calculate(List<AHPPair<Criteria>>
criteriaPairList,
            Dictionary<Criteria, Dictionary<Resource, double>>
alternativeDict)

        {
            Dictionary<Criteria, List<AHPPair<Resource>>>
criteriaAlternativeDict =
                ConvertAlternativeMatrix(alternativeDict);

            List<Criteria> criteriaList;
            var criteriaArray = ConvertMatrix(criteriaPairList, out
criteriaList);
            Matrix<double> criteriaMatrix =
CreateMatrix.DenseOfArray(criteriaArray);
            Matrix<double> criteriaW = CalculateW(criteriaMatrix);
            int criteriaCount = criteriaMatrix.ColumnCount;
            int alternativeCount = 0;

            List<Matrix<double>> alternativeWList = new
List<Matrix<double>>();
            List<Resource> alternativeList = null;
            foreach (var criteria in criteriaAlternativeDict.OrderBy(x =>
x.Key.ID))
            {
                var alternativeArray = ConvertMatrix(criteria.Value, out
alternativeList);
                Matrix<double> alternativeMatrix =
CreateMatrix.DenseOfArray(alternativeArray);
                alternativeCount = alternativeMatrix.RowCount;
                Matrix<double> alternativeW = CalculateW(alternativeMatrix);
                alternativeWList.Add(alternativeW);
            }

            double[,] altWArray = new double[alternativeCount, criteriaCount];

            for (int c = 0; c < alternativeWList.Count; c++)
            {
                var wMatrix = alternativeWList[c];
                for (int r = 0; r < alternativeCount; r++)
                {
                    altWArray[r, c] = wMatrix[r, 0];
                }
            }
            Matrix<double> altWMatrix = CreateMatrix.DenseOfArray(altWArray);
            var resultMatrix = altWMatrix * criteriaW;
```

```
            Dictionary<Resource, double> result = new Dictionary<Resource,
double>();
            for (int i = 0; i < alternativeList.Count; i++)
            {
                result.Add(alternativeList[i], resultMatrix[i, 0]);
            }

            return result;
        }

        private Matrix<double> CalculateW(Matrix<double> criteriaMatrix)
        {
            double[,] result = new double[criteriaMatrix.RowCount, 1];
            double[,] normMatrix = new double[criteriaMatrix.RowCount,
criteriaMatrix.ColumnCount];
            for (int c = 0; c < criteriaMatrix.ColumnCount; c++)
            {
                double sum = 0;
                for (int r = 0; r < criteriaMatrix.RowCount; r++)
                {
                    sum += criteriaMatrix[r, c];
                }

                for (int r = 0; r < criteriaMatrix.RowCount; r++)
                {
                    normMatrix[r, c] = criteriaMatrix[r, c] / sum;
                }
            }

            for (int r = 0; r < criteriaMatrix.RowCount; r++)
            {
                double sum = 0;
                for (int c = 0; c < criteriaMatrix.ColumnCount; c++)
                {
                    sum += normMatrix[r, c];
                }
                result[r, 0] = sum / (criteriaMatrix.ColumnCount * 1.0);
            }
            var resultMatrix = CreateMatrix.DenseOfArray(result);
            return resultMatrix;
        }

        private double[,] ConvertMatrix<T>(List<AHPPair<T>> pairList, out
List<T> tList)
            where T : AHPIdentity
        {
            Dictionary<int, T> criteriaDict = new Dictionary<int, T>();
            foreach (var item in pairList)
            {
                if (!criteriaDict.ContainsKey(item.Main.ID))
                {
                    criteriaDict.Add(item.Main.ID, item.Main);
                }
                if (!criteriaDict.ContainsKey(item.Sub.ID))
                {
                    criteriaDict.Add(item.Sub.ID, item.Sub);
                }
            }
            tList = criteriaDict.OrderBy(x => x.Key).Select(x =>
x.Value).ToList();
```

```
            double[,] matrix = new double[tList.Count, tList.Count];
            foreach (var item in tList)
            {
                int mainIndex = tList.IndexOf(item);
                matrix[mainIndex, mainIndex] = 1;
                var mains = pairList.Where(x => x.Main.ID == item.ID);
                foreach (var mCriteria in mains)
                {
                    int subIndex = tList.IndexOf(mCriteria.Sub);
                    matrix[mainIndex, subIndex] = mCriteria.Value;
                    matrix[subIndex, mainIndex] = 1.0 / (mCriteria.Value *
1.0);
                }
            }

            return matrix;
        }

        private Dictionary<Criteria, List<AHPPair<Resource>>>
ConvertAlternativeMatrix(
            Dictionary<Criteria, Dictionary<Resource, double>>
criteriaAlternativeDict)
        {
            Dictionary<Criteria, List<AHPPair<Resource>>> result =
                new Dictionary<Criteria, List<AHPPair<Resource>>>();

            foreach (var criteriaDict in criteriaAlternativeDict.OrderBy(x =>
x.Key.ID))
            {
                var alternativePointList = criteriaDict.Value.OrderBy(x =>
x.Key.ID).ToList();
                List<AHPPair<Resource>> altPairList = new
List<AHPPair<Resource>>();
                for (int i = 0; i < alternativePointList.Count; i++)
                {
                    for (int j = i + 1; j < alternativePointList.Count; j++)
                    {
                        AHPPair<Resource> pair = new AHPPair<Resource>()
                        {
                            Main = alternativePointList[i].Key,
                            Sub = alternativePointList[j].Key,
                            Value = alternativePointList[j].Value /
alternativePointList[i].Value
                        };
                        altPairList.Add(pair);
                    }
                }
                result.Add(criteriaDict.Key, altPairList);
            }
            return result;
        }
    }
}
```