T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OSGI'DA UYGULAMA MODELİ TANIMLANMASI

İLKER RIZA ERNAS

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
YRD. DOÇ. DR. YUNUS EMRE SELÇUK

İSTANBUL, 2011

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

OSGI'DA UYGULAMA MODELİ TANIMLANMASI

İlker Rıza ERNAS tarafından hazırlanan tez çalışması 22.09.2011 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Yunus Emre Selçuk

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Yunus Emre Selçuk

Yıldız Teknik Üniversitesi _____

Prof. Dr. Oya Kalıpsız

Yıldız Teknik Üniversitesi _____

Prof. Dr. Nadia Erdoğan

İstanbul Teknik Üniversitesi _____

ÖNSÖZ

Dinamik Uygulama Platformu sunucu, bilgisayar ve mobil cihazlarda dinamik olarak uygulamaların yüklenmesini ve yönetimini ağ bazlı olarak gerçekleştirmeyi hedeflemektedir. Dinamik Uygulama Platformunun mobil uygulamalarda da bulunduğu ağa göre değişiklik göstermesi sağlanmıştır.

Çalışmamda teknik ve akademik alanlarda yönlendirmelerde bulunan değerli hocam Yrd. Doç. Dr. Yunus Emre Selçuk'a, yüksek lisans çalışmalarına dahil olmam ve mesai saatlerinde çalışmalarımı sürdürmemde yardımcı olan yöneticim Mehmet Sürav'a ve hayatımın her döneminde desteğini eksik tutmayan aileme teşekkürü bir borç bilirim.

Ağustos, 2011

İlker Rıza ERNAS

İÇİNDEKİLER

KISALTMA LİSTESİ

App-Agent	Application Agent
App-Manager	Application Manager
App-Proxy	Application Proxy
DAP	Dynamic Application Platform (Dinamik Uygulama Platformu)
OSGi	Open Service Gateway İnterface

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖZET

OSGI'DA UYGULAMA MODELİ TANIMLANMASI : DİNAMİK UYGULAMA PLATFORMU

İlker Rıza ERNAS

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Yunus Emre Selçuk

Günümüzde hayatımızın her alanında mobil ya da sunucu taraflı bir çok uygulama kullanılmaktadır. Bu noktada dinamik uygulama platformu hem kurumsal hem de bireysel çözümler üzerinde uygulama yönetimi yapmayı hedeflemiştir.

Bireysel çözümler açısından bakıldığında, uygulamaların çalışma zamanında değişim gösterebilmesi için değişim yapabilme (*Hot Swapping*) yeteneğine sahip olması gerekmektedir.

Kurumsal çözümler açısından bakıldığında ise değişim yapabilme özelliğine ek olarak platform üzerinde geliştirme yapılabilmesi de gerekmektedir. Uygulamaların ağ bazlı yönetimi de diğer bir kurumsal ihtiyaç olarak göze çarpmaktadır.

Yukarıda değinilen ihtiyaçları karşılamayı amaçlayan bu tez çalışması kapsamında, bir dinamik uygulama platformu, JAVA dili üzerinde ve OSGi modül sistemi kullanılarak geliştirilmiştir. JAVA dilinin seçilmesinde en önemli etkenler Java'nın Android gibi mobil işletim sistemleri tarafından yaygın olarak desteklenmesi ve geniş bir kod kütüphanesine sahip olmasıdır.

Anahtar Kelimeler: Java, Android, OSGi, Hot Swapping

ABSTRACT

DEFINING APPLICATION MODEL IN OSGI : DYNAMIC APPLICATION PLATFORM

İlker Rıza ERNAS

Computer Engineering Department

Graduate Thesis

Advisor: Assistant Professor Yunus Emre Selçuk

Serverside or mobile applications are used in the most part of daily life. Dynamic application platform's goal is managing both enterprise and end-user solution.

According to the end-user solutions, applications must be capable of changing itself (hot swapping) on runtime.

According to the enterprise solutions, developmen must be possible in addition to being capable of changing itself.

In order to cover requirements of end-user and enterprise applications, java programming language and OSGi modularity system is used. Another reasons of using java programming language are support for Android mobile operating system and high range of library support for Enterprise applications.

Dynamic Application Platform managing applications also in network centric way.

Keywords: Java, Android , OSGi , Hot Swapping

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

OSGi java dünyasının yeni çalışma ortamlarından birisidir. OSGi java'ya dinamik yüklemeler ve daha iyi bağımlılık yönetimi getirmiştir. OSGi, Open Service Gateway Interface(Açık Servis Geçidi Arayüzü)'in kısaltmasıdır. OSGi, servislerin düşük bağlaşımlı olarak eklenip kullanabildiği bir servis çatısıdır. Düşük bağlaşımlı servisleri içermesi, OSGi'ın dinamik yüklemeler yapmasını mümkün kılar.

OSGi uygulamalarının yapıtaşları bundle'dır. Bundle'ların normal jar dosyalarından farkı manifest dosyasında daha fazla bilgi içermesidir. Bu bilgiler arasında bağımlılık bilgisi, uygulamayı başlatacak sınıfın (activator class) bilgisi, içe ve dışa aktarılacak paket bilgisi gibi bilgiler yer almaktadır. Bundle'lar farklı havuz türlerinde yer alabilir. Günümüzde yaygın olarak kullanılan bundle havuzları arasında P2 ve OBR sayılabilir.

OSGi java dünyasına yenilikleri bir takım zorluklarla getirmektedir. Zorlukların en başında, özellikle kurumsal uygulamaların yönetimi yer almaktadır. Her bundle'ın bir kurulum yolu vardır. Bir bundle yüklenmeden önce bağımlılığı olan diğer bundle'ların da yüklenmiş olması gerekmektedir.

1.2 Tezin Amacı

Tez kapsamında gerçeklenecek olan Dinamik uygulama platformu (DAP), OSGi uygulamaları için bir modeldir. DAP sadece ağ merkezcil bir uygulama yöntemi

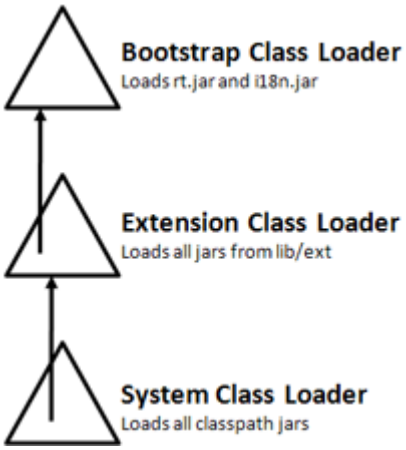
önermekle kalmayıp, geliştiricilere bir geliştirme ortamı ve çalıştırma ortamı da sunmaktadır. DAP, equinox ve felix gibi farklı OSGi içericilerde çalışabilir. Dolayısıyla giriş bölümü OSGi hakkında temel bilgilerin verilmesi, OSGi altında mevcut uygulama tanımı çalışmalarının tanıtılması ve bu çalışmaların birbirleri ile karşılaştırılması amaçlarını taşıyan alt bölümlere ayrılmıştır.

1.3 Hipotez

Open Service Gateway Interface kurumsal şirketler tarafından ortaya konulan ihtiyaçlar ve özellikler doğrultusunda oluşturulmuş spefisikasyonlar bütünüdür. OSGi ittifakını oluşturan şirketler arasında Alcatel-Lucent, Deutsche Telekom, Ericsson AB, Hitachi, IBM, Mitsubishi Electric, NEC, Oracle, Qualcomm, Red Hat, SAP AG , Siemens AG, Software AG, TIBCO, VMWare bulunmaktadır.

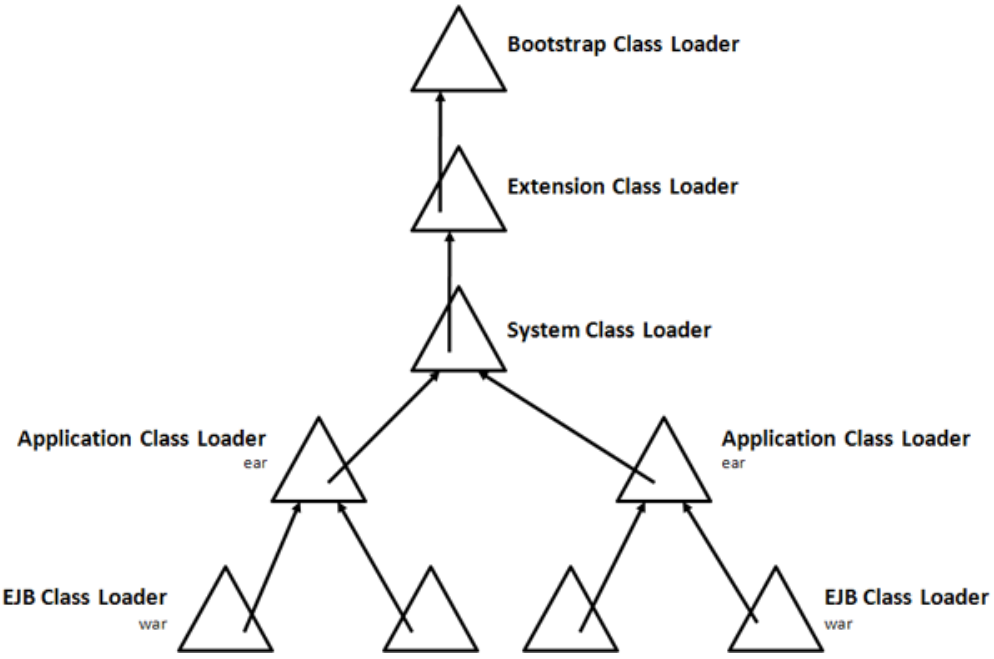
OSGi ile Java dilinin günün gereksinimlerine uygun bir hale getirilmesi amaç olarak edinilmiştir. Söz konusu gereksinim, Java diline eksiksiz bir dinamik bileşen modeli kazandırmaktır. Günümüzde halen Java sanal makinelerine böyle bir dinamik bileşen modeli dâhil değildir. Bu nedenle OSGi ittifakı gibi oluşumların çalışmalarına IT sektörünün duyduğu ihtiyaç sürmektedir.

OSGi kullanılarak Java uygulamalarının modülerlik ve sürümlendirme olanakları arttırılmaktadır. *Modülerlik*, bir uygulamanın yeniden kullanılabilir ve amaca yönelik yapıtaşlarına parçalanabilme derecesini simgeler. Modülerlik arttıkça parçalar arasında soyutlama sağlanacak, karmaşıklık azalacak ve uygulamanın bakımı kolaylaşacaktır. *Sürümlendirme (versioning)* sayesinde ise, uygulamalar aynı parçaların farklı sürümlerini (versiyonlarını) bir arada kullanabilirler. Sürümlendirme OSGi classloader mekanizması sayesinde yapılabilmektedir. OSGi'ın standart Java ve J2EE sınıf yükleme mekanizmaları incelendiğinde, OSGi'da her bundle için sınıf yükleme mekanizması olduğu görülür.



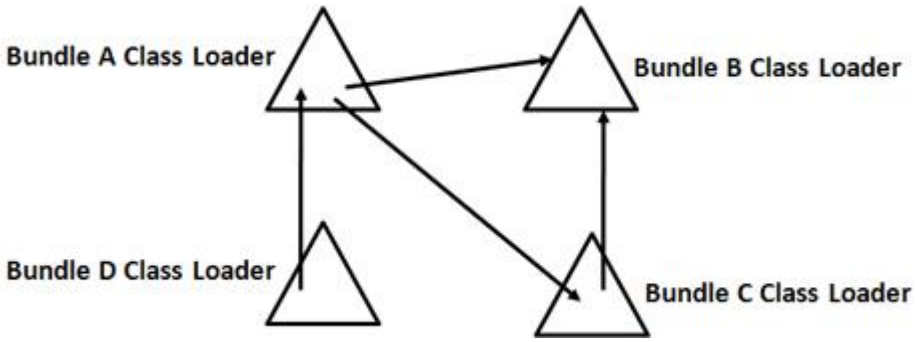
Şekil 1.1 Java'nın sınıf yükleme düzeneği

Şekil 1.1'de görüldüğü üzere, standart Java'da öncelikle rt.jar ve i18n.jar kütüphaneleri sonrasında ext dizini altındaki kütüphaneler, son olarak ise classpath'de tanımlı olan kütüphaneler yüklenir.



Şekil 1.2 JEE'de sınıf yükleme düzeneği

J2EE, Standart Java'dan farklı olarak uygulama seviyesinde (ear-Enterprise Archive) veya web uygulaması seviyesinde(war-Web Archive) kütüphaneleri yükler. Bu durum Şekil 1.2'de görülebilir.



Şekil 1.3 OSGi'ın sınıf yükleme düzeneği

OSGi ile gelen sınıf yükleme mekanizması ise bundle seviyesinde kullanılmaktadır. Bu kullanımdan dolayı aynı sınıfın birden fazla kullanılabilmesi mümkün olmakta ve sürümlendirme desteği verilebilmektedir.(Mak, Rubio [2])

Mevcut OSGi uygulama modellerinden en yaygın kullanılan ve belli bir olgunluk düzeyine ulaşmış olanları arasında Spring Par, Apache Aries ve Apache Ace sayılabilir.

Spring Par (Platform Archive), Spring topluluğu tarafından geliştirilmiş bir uygulama modelidir. Spring'in kendisine ait Spring DM adında OSGi içericisi bulunmaktadır. Spring Par uygulama modeli sadece Spring DM için anlamlı bir uygulama modelidir.

Apache Aries, Apache topluluğu tarafından geliştirilmiş bir uygulama modelidir. Bundle'lar eba uzantılı dosyalarda sıkıştırılır. Şekil 1.4'te tipik bir Eba dosyasının içeriği görülebilir. Eba dosyalarında Application.mf adında, uygulamanın özelliklerini taşıyan bir tanım dosyası da yer alır. Şekil 1.5'te ise örnek bir tanım dosyası verilmiştir. Apache aries P2 ve OBR bundle havuzundan faydalanabilir.

```
META-INF/APPLICATION.MF
bundle1.jar
bundle2.jar
bundle3.jar
```

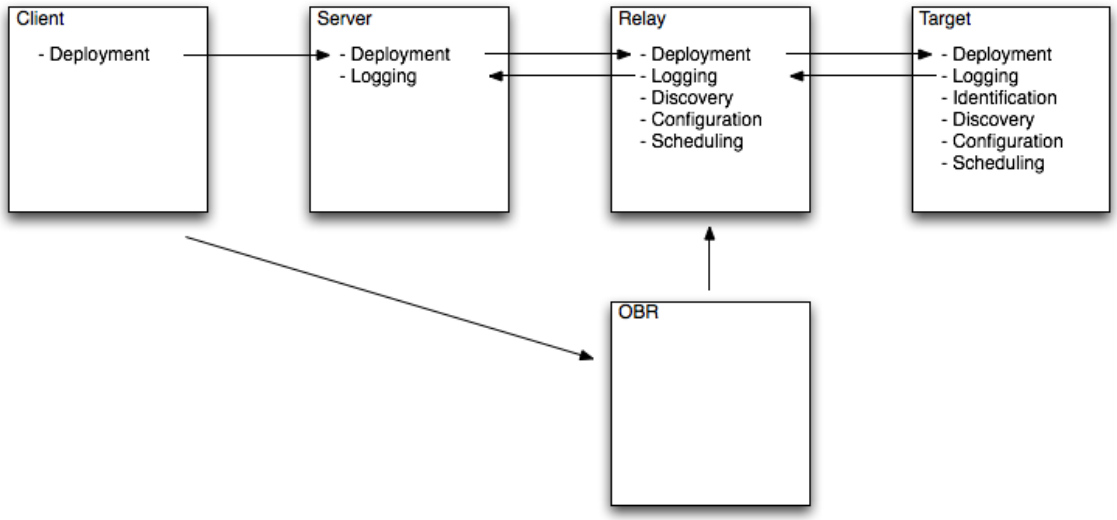
Şekil 1.4 Eba dostyası içeriği örneği

```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
```

```
Application-Name: Bank Account
Application-SymbolicName: com.mybank.account.app
Application-Version: 1.0
Application-Content:
      com.mybank.account.bankWeb; version=1.0.0,
      com.mybank.account.bankAccount; version=1.0.0,
      com.mybank.account.common; version=1.0.0,
      com.mybank.account.utility; version=1.0.0
Application-ExportService: com.mybank.account.service.AccountService
Application-ImportService:
com.mybank.security.UserAuthService;filter="(security=strong)"
```

Şekil 1.5 Application.mf dosyası örneği

Apache Ace, diğer çözümlerden farklı olarak yükleme amaçlı bir uygulama modelidir. Yüklemeler merkezi bir noktadan kontrol edilmektedir. Client, yüklemelerin kontrol edildiği merkezdir. Server, client tarafından yapılan yüklemelerin metadata'sının tutulduğu düğümdür. Relay ise target'ın yüklemelerini yapan düğümdür. Relay yüklemeleri yaparken OBR bundle havuzundan yararlanır. Şekil 1.6'da Ace'in blok şeması görülebilir.



Şekil 1.6 Apache Ace'in Genel Yapısı

Çizelge 1.1'de bu bölümde değinilen çalışmaların çeşitli açılardan karşılaştırılması ile elde edilen sonuçlar görülebilir.

Çizelge 1.1 Önceki Çalışmaların Karşılaştırması

Özellikler	**Spring Par**	**Apache Ace**	**Apache Aries**
Farklı OSGi içericilerde çalışma	Yok (Sadece Spring DM Server üzerinde çalışabilir)	Var (Equinox ve Felix desteği mevcut)	Var (Equinox ve Felix desteği mevcut)
Geliştirme Ortamı Desteği	Var (Eclipse ortamında geliştirilmiş bir geliştrme arayüzü	Yok (Geliştirme desteği bulunmamakta)	Yok (Geliştirme desteği bulunmamakta)
Tek dosya ile yükleme	Var (Par dosyası ile yükleme yapılabilmekte)	Var (Eba dosyası ile yükleme yapılabilmekte.)	Yok (Maven yardımıyla kurulum yapılabilmekte)
P2 ve OBR desteği	Yok (Spring kendi repository'lerini kullanmakta)	Var	Var

Dinamik uygulama platformu diğer uygulama modellerinden farklı olarak uygulama yönetmini ağ merkezcil olarak yapmayı hedeflemektedir.Mobil cihazlar ve bilgisayar sistemlerinde çalışması sağlanmıştır.

BÖLÜM 2

ÖNERİLEN ÇÖZÜM: DİNAMİK UYGULAMA PLATFORMU (DAP)

Spring Par, Apache Aries ve Apache Ace gibi uygulama modelleri geliştirme ortamını ve çoklu OSGi içericilerde çalışma özelliğini beraber bulundurmamaktadır. Hâlbuki DAP (Dynamic Application Platform), yani tez çalışması kapsamında gerçeklenecek olan uygulama modeli, bu iki özelliği birlikte sağlayacaktır. DAP ile Çoklu OSGi içericide çalışma özelliği ile uygulamaların OSGi içericilerden bağımsız olarak geliştirilmesi mümkündür. DAP ile sunulan geliştirme ortamı ise yazılım geliştiricilerin uygulama modelini rahatlıkla oluşturmalarını sağlar. DAP geliştirme ortamını ve çoklu OSGi içericilerde çalışma özelliğini sağlamakta, ayrıca yüklemelerin ağ merkezcil olarak yapılmasını ve tek bir merkezden yönetilmesini de sağlamaktadır.

2.1 Kullanılan Teknolojiler

Dinamik uygulama platformu geliştirme sırasında kütüphane yönetimi işlevine sahip olan Maven tool'unu kullanmaktadır. Çalışma sırasında ise ağdaki cihazları bulamabilmek için çoklu gönderim (Multicasting) kullanılmaktadır.

2.1.1 Maven

Maven, Jakarta Turbine projesinin geliştirilme sürecinde, proje aşamalarını basitleştirmek için kullanılmıştır. Maven ile standart bir yol ile projeleri geliştirmek, anlaşılır bir açıklama ile hangi proje ne içeriyor, projeleri kolayca yayınlamak ve JAR dosyalarını projeler arasında paylaştırmak için tasarlanmıştır.

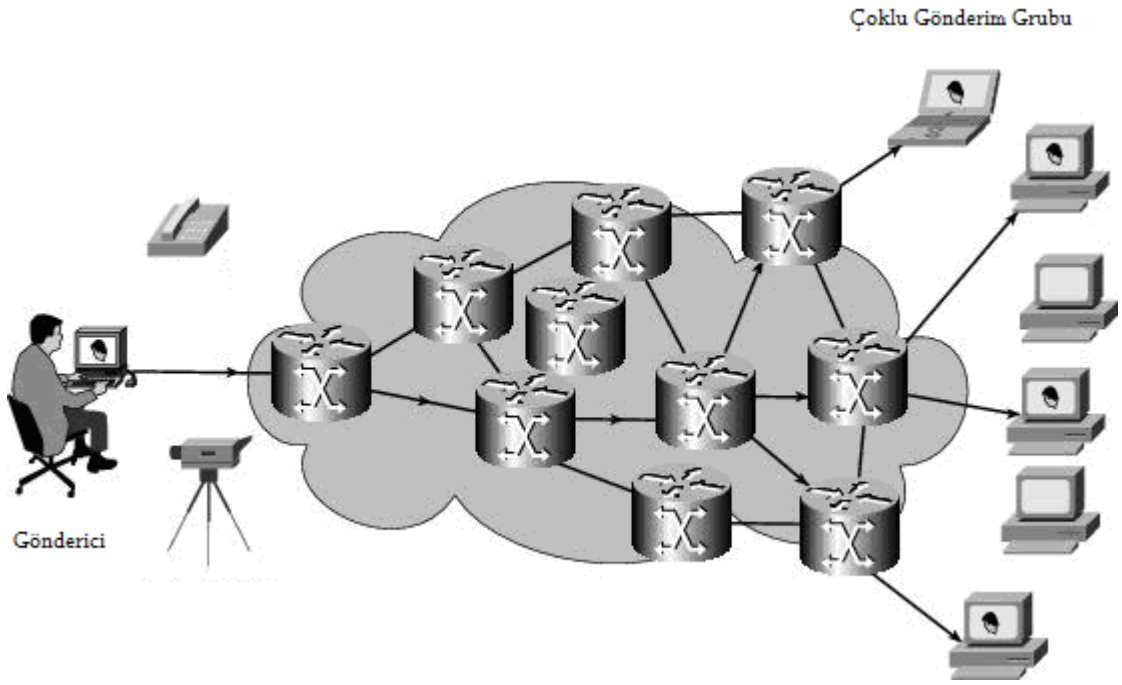
Maven'ın temel amacı, projenin bütün durumlarını en kısa biçimde programcıya anlatabilmektir. Maven bu amaca bir kaç değişik yolla ulaşır:

- Projenin geliştirme adımlarını kolaylaştırmak
- Tek bir geliştirme standardı sağlamak
- Projenin niteliklerini bilgi olarak sağlamak
- Projenin geliştirilmesinde iyi çözümleri yol göstermek
- Projenin yeni özelliklerine anlaşılır bir geçiş sağlamak

Sonuç olarak, bu araç herhangi bir Java tabanlı projenin geliştirilmini ve yöneyimini sağlayabilmektedir. Umulan Maven ile Java geliştiricilerine daha rahat bir ortam yaratmakla birlikte, Java tabanlı projelerin anlaşılabilirliğini arttırmaktır.

2.1.2 Çoklu Gönderim (Multicasting)

Bir grup cihaza veri göndermeye çoklu gönderim (multicast) denir. Grup adresleri kullanılarak, birden fazla cihazın tekil bir adresi dinlemesi (buradan veri beklemesi) sağlanmaktadır. Grup adresine bir frame iletildiğinde, bu grupta olan bütün cihazlar bu veriyi alacaktır. IP protokolünün 802.3 MAC Alt Katman (802.3 MAC Sublayer) protokolünden itibaren, yani OSI katmanlı yapısının 2. seviyesinden itibaren bu tür bir destek gelmektedir. Tüme gönderim (broadcast) ise ağdaki bütün cihazlara veri iletimini sağlayan özelleşmiş bir çoklu gönderimdir. Adres alanının hepsinin 1 olması durumu, tümüne gönderimi bildirir.



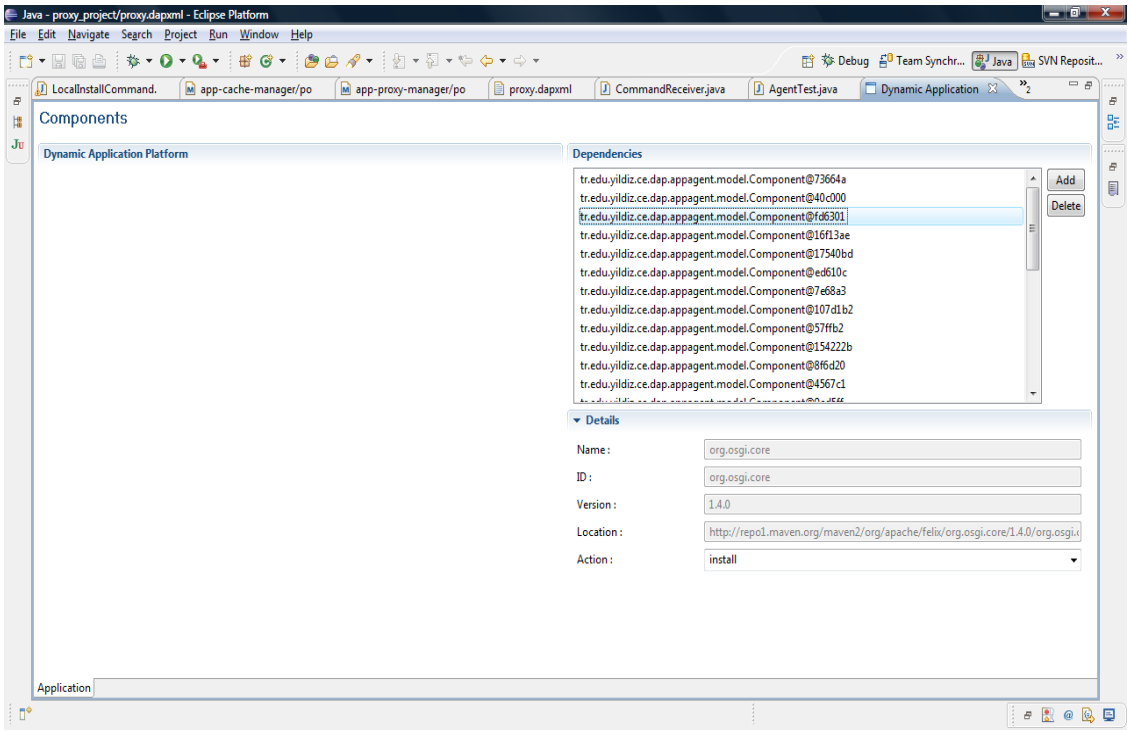
Şekil 2.1 Çoklu Gönderim Örnek Gösterim

IP üzerinden çoklu gönderim yapmaya İnternet Çoklu Gönderimi (Internet Multicasting) denmektedir. Radyo yayınları, video konferans gibi bir çok uygulamada çoklu gönderimden faydalanmaktadır. Böylece var olan ağ alt yapısının daha etkin bir şekilde kullanılması hedeflenmektedir. Çoklu gönderim sisteminde, öncelikle düğüm grupları tanımlanmaktadır. Tüm ağa belirli bir mesajı göndermek yerine, daha önceden tanımlanmış düğüm gruplarına çoklu gönderim sağlanmaktadır.

224.0.0.0/24 çoklu gönderim adresleri yerel ağ (link-local) adreslerdir. Bu adresler için TTL (time to live) değeri 1 olduğundan yerel ağ dışına çıkamayacaktır. Geriye kalan adresler (224.0.1.0 - 238.255.255.255 aralığı) genel kapsam olarak tanımlanmaktadır.

2.2 Geliştirme

Dinamik uygulama platformu ile çalışacak yazılımların daha kolay geliştirilebilmesi için Eclipse IDE'si için bir eklenti yazılmıştır. OSGi Bundle projelerinin Maven projesi olması gerekmektedir. OSGi bundle projeleri yaratılırken Maven Felix eklentisi kullanılabilinir. (Clayberg, Rubel [3])



Şekil 2.2 Dinamik Uygulama Modeli tanımı yapılmasını sağlayan Eclipse Eklentisi

Dinamik Uygulama platfomu eklentisi, projelerin ve bağımlılıklarının bulunduğu bir xml dosyası oluşturmaktadır. Oluşturulan bu dosyasının uzantısı dapxml'dir. Dapxml dosyası App-Proxy'ye yüklenebilinir ya da manüel olarak yüklenebilir. Örnek bir dapxml dosyası Şekil 2.2'de görülebilir.

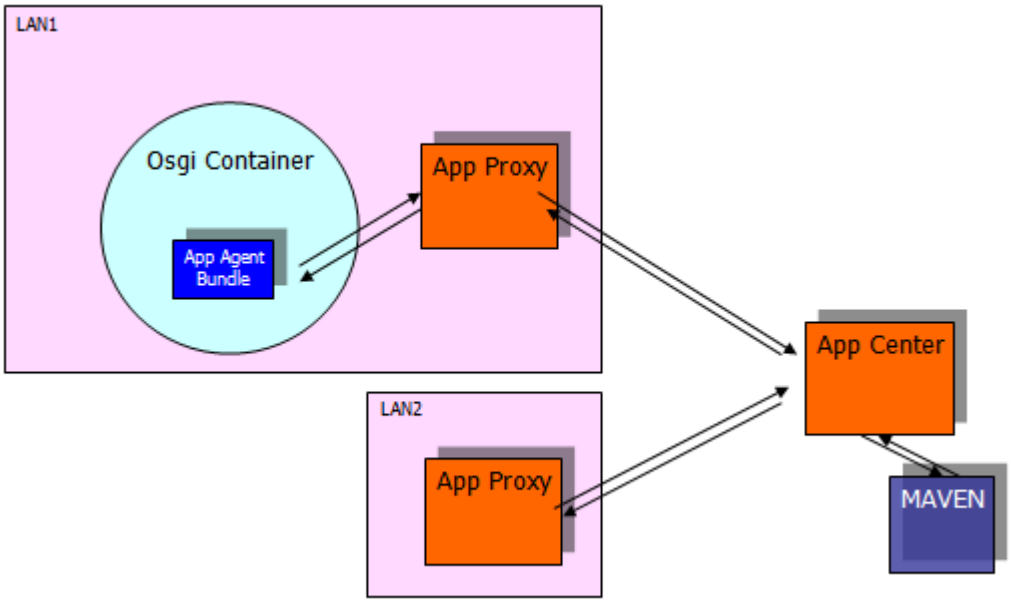
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<appDefinition xmlns="http://www.example.org/app-comm">
    <app version="0.0.1" name="AppProxy"/>
    <componentList>
      <component url="file:/C:/dev/workspaces/runtime-
EclipseApplication/sample1/target/sample1-0.0.1-SNAPSHOT.jar"
version="0.0.1-SNAPSHOT" name="sample1">
        <actions>
          <action>install</action>
        </actions>
      </component>
      <component url="file:/C:/dev/workspaces/runtime-
EclipseApplication/sample2/target/sample2-0.0.1-SNAPSHOT.jar"
version="0.0.1-SNAPSHOT" name="sample2">
        <actions>
          <action>install</action>
        </actions>
      </component>
    </componentList>
</appDefinition>
```

Şekil 2.3 Örnek bir dapxml dosyası

Dapxml dosyasında uygulamayı oluşturan bundle'ların isimleri, sürümleri ve bundle'ın kendisini indirebilineceği bir URL yer almaktadır. Uygulamada başlaması gerek bundle'ların action düğümüne install komutundan sonra start komutu eklenir.

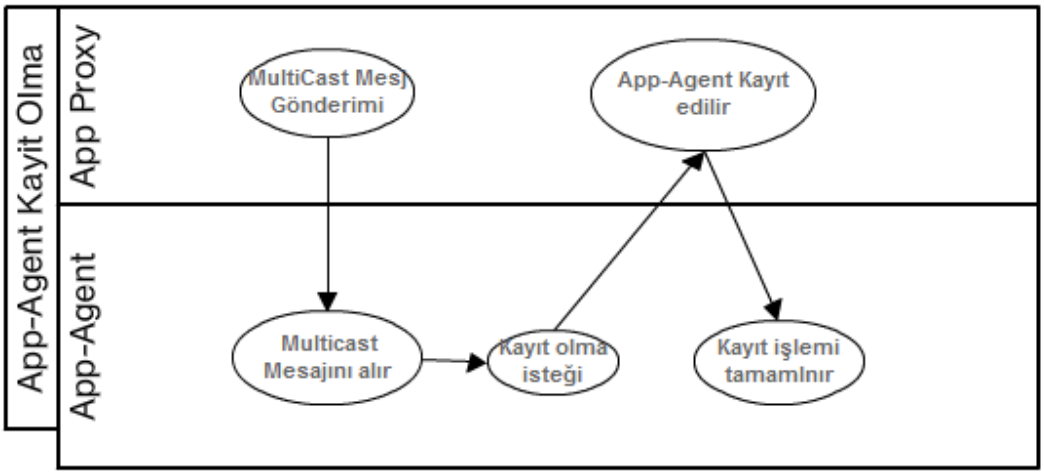
2.3 Çalıştırma

Şekil 2.3'te görüldüğü üzere, Dinamik uygulama platformu 3 tip düğümden oluşmaktadır. Bu düğümler App-Agent, App-Proxy ve App-Center olarak adlandırılmışlardır.



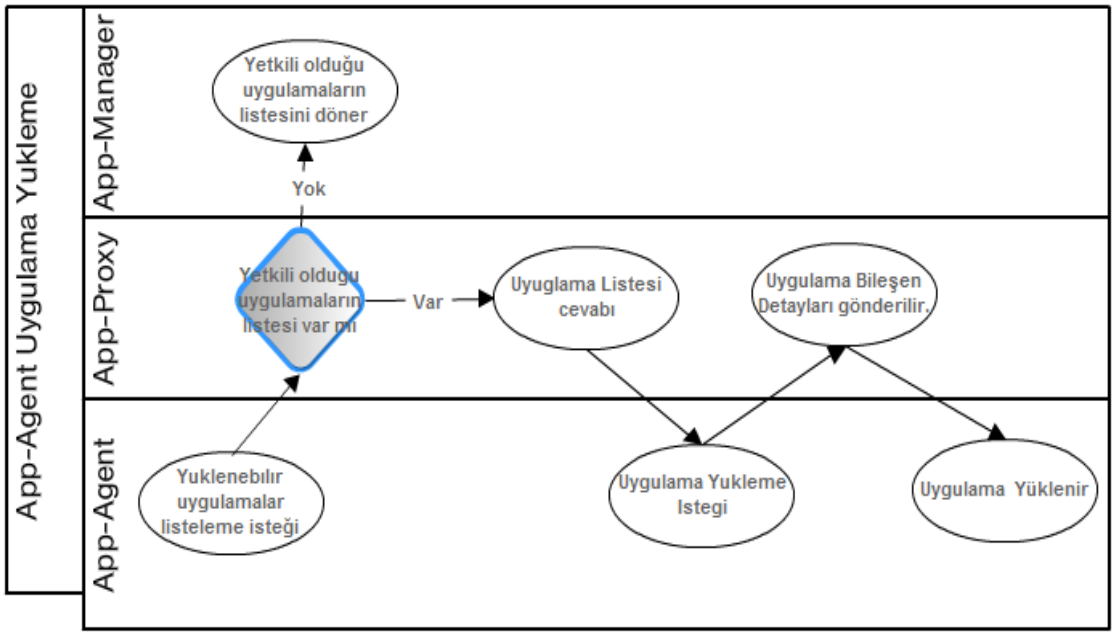
Şekil 2.4 DAP Düğüm Yapısı

DAP'ın ilk bileşeni olan App-Center, uygulamaların yetkilendirilmesinin yapıldığı düğümdür. DAP bileşenlerinden ikincisi olan App-Proxy, belirli bir network alanında yüklenmesine izin verilen uygulamaların metadata'sını saklayan düğümdür. DAP'ın üçüncü bileşeni olan App-Agent ise, hem App-Proxy ile OSGi içericisi (container) arasındaki iletişimi, hem de uygulama için gerekli bundle'ların yüklenmesini ve kaldırılmasını sağlayan bir bundle'dır. App-Proxy, App-Agent'ları bulamabilmek için multicast olarak yayın yapar. App-Agent, App-Proxy'yi bulduktan sonra, kendisini App-Proxy'ye kayıt eder. Kayıt işlemi Şekil 2.4'te gösterilmiştir.



Şekil 2.5 App-Agent Kayıt Olma Mesaj Akışı

App-Proxy'ler yetki için App-Center'a istek göndermektedir. Bu çalışma biçimi, Şekil 2.5'te görselleştirilmiştir.



Şekil 2.6 DAP bileşenlerinin aralarındaki etkileşim

Dinamik Uygulama Platformu'nun en önemli özelliği ağ merkezcil olmasıdır. Şekil 2.4 incelendiğinde, iki ayrı App-Proxy'nin farklı yerel ağlarda bulunabileceği görülebilir. Ancak bir App-Proxy'nin bir OSGi içericisine uygulama yükleyebilmesi için, bu proxy ve

içericinin aynı yerel ağda bulunması tercih edilir. Yükleme işlemini OSGi içericilerde bir bundle olarak yüklenmiş olan App-Agent gerçekleştirmektedir. App-Proxy konfigüre edildiği takdirde bundle'ları da saklayabilmektedir. Ayrıca App-Proxy de bir dinamik uygulama platformu kullanarak geliştirilmiştir.

2.3.1 Dinamik Uygulama Platformu Bileşen Detayları

Dinamik uygulama platformu ana olarak 3 bileşenden oluşmaktadır:

- App-Agent
- App-Proxy
- App-Center

Bu bileşenlerin yanı sıra ayrıntıları Bölüm 2.3.1.4'te verilecek olan CrystallApps mobil uygulaması da Android işletim sistemi için kullanılmaktadır.

App-Agent ve App-Proxy bileşenleri Java Concurrency alt yapısı kullanılarak, multithread bir çalışması sağlanmıştır.(Goetz [5])

2.3.1.1 App-Agent

App-Agent standart OSGi bundle'dır. App-Agent'ın amacı uzaktaki ya da yerel ortamdaki bundle'ları indirip, kurmak ya da sistemde bulunan bundle'ı kaldırmaktır. App-Agent'ın manifest dosyası Şekil 2.7'da verilmiştir. App-Agent'ın tr.edu.yildiz.ce.dap.appagent.model paketini dışa açmış olduğu ve diğer bundle'lar tarafından kullanılmasına izin verdiği, Şekil 2.7 incelendiği zaman görülecektir.

```
Manifest-Version: 1.0
Tool: Bnd-0.0.357
Bundle-Name: app-agent
Created-By: Apache Maven Bundle Plugin
Bundle-Version: 0.0.1.SNAPSHOT
Build-Jdk: 1.6.0_21
Bnd-LastModified: 1298217809021
Bundle-ManifestVersion: 2
Bundle-Activator: tr.edu.yildiz.ce.dap.appagent.AppAgentActivator
Export-Package: tr.edu.yildiz.ce.dap.appagent.model;uses:="javax.xml.b
```

```
 ind.annotation,javax.xml.bind,javax.xml.namespace"
Import-Package: javax.xml.bind,javax.xml.bind.annotation,javax.xml.nam
 espace,org.osgi.framework;version="1.5",
Bundle-SymbolicName: app-agent
```

Şekil 2.7 App-Agent manifest dosyası

2.3.1.2 App-Proxy

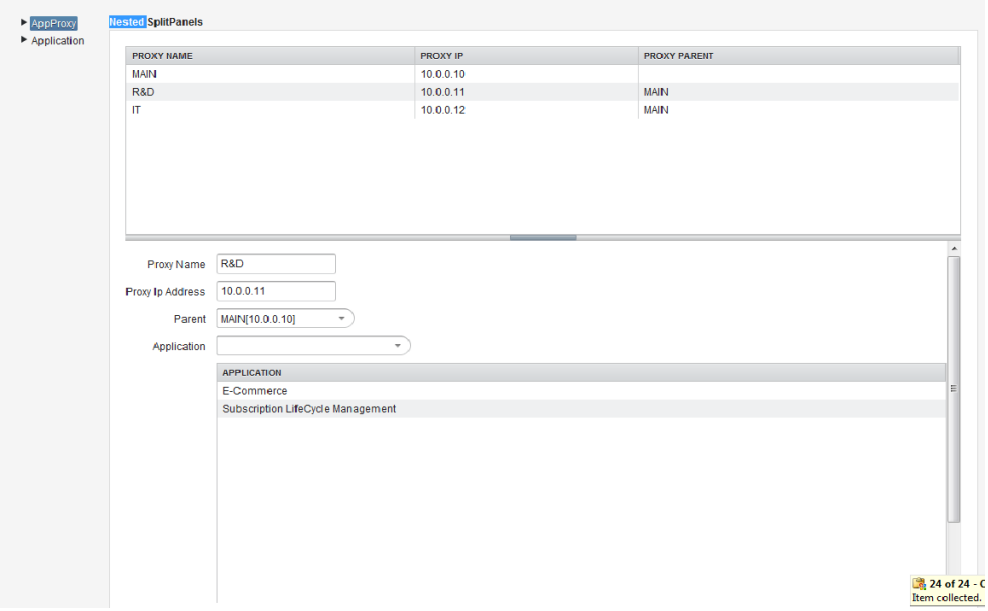
App-Proxy, OSGi içerici üzerinde çalışan bir uygulamadır. App-proxy bir DAP uygulaması olarak tasarlanmıştır. App-Proxy genel olarak aşağıdaki bileşenlerden oluşmaktadır:

- App-Proxy-Manager, App-Agent'lar ile iletişimi sağlayan bileşendir.
- App-Cache-Manager, uygulamaları oluşturan bileşenlerin App-Proxy'ler üzerinde saklanmasını sağlayan bileşendir.
- Http ve OSGi kütüphaneleri.

App-Proxy uygulaması, App-cache-manager ve App-proxy-Manager olmak üzere, 2 ana parçadan oluşmaktadır. App-cache-manager, Proxy üzerinde tutulacak bileşenlerin yönetmini yapmaktadır. App-proxy-manager ise, App-Agent ile iletişim ve yetkilendirme işlemleriyle ilgilenmektedir.

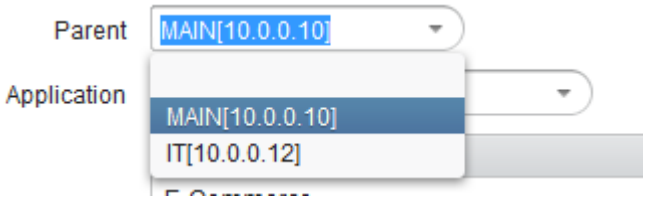
2.3.1.3 App-Center

App-Center ağ üzerinde bulunan App-Proxy'lerin DAP uygulamaları için yetkilendirilmesinin yapıldığı uygulamadır. Şekil 2.8'da App-Proxy tanımlama sayfası gösterilmektedir.



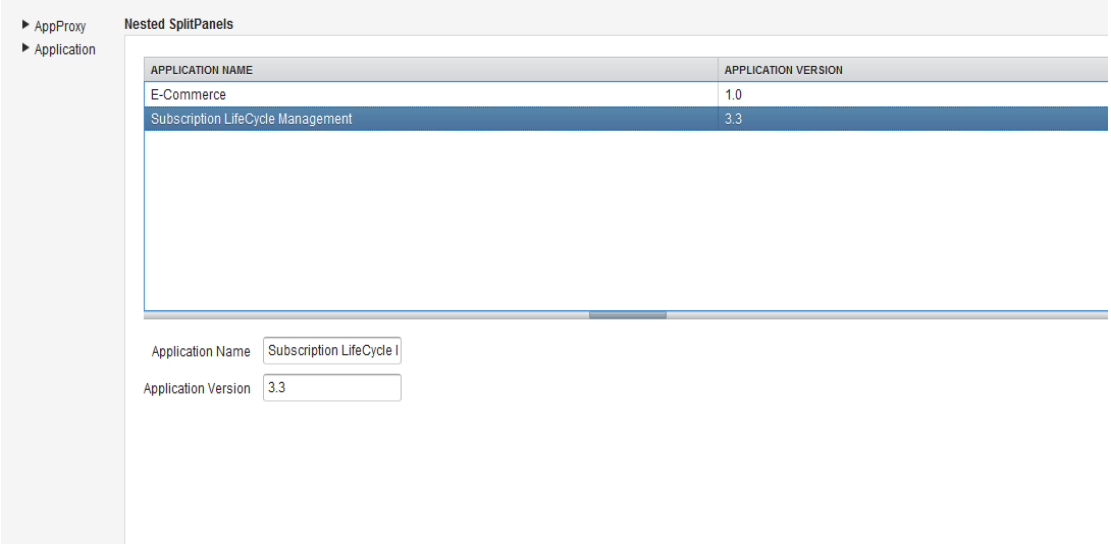
Şekil 2.8 App-Proxy Tanımlama Sayfası

App-Proxy'ler arasındaki hiyerarşi App-center üzerinde tanımlanmaktadır. Şekil 2.9'da da görüldüğü üzere her App-Proxy için bir üst düzey (ebeveyn) App-Proxy seçilebilmektedir.



Şekil 2.9 App-Proxy üst düzey App-Proxy seçme

Uygulamalar App-Manager üzerinde tanımlanabilmektedir. İlgili ekran Şekil 2.10'de görülebilir.



Şekil 2.10 Application Tanınmlama Sayfası

App-Proxy ve App-Center, web servis üzerinden iletişim kurar. App-Proxy yetkisi olan uygulamaları App-Center'dan ister. App-center, App-proxy hiyerarşisine göre uygulamaların listesini App-proxy'ye gönderir. İlgili mesajlar Şekil 2.11 ve Şekil 2.12'de görülebilir.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://ws.app_manager.ce.yildiz.edu.tr/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <q0:listAvailableApplications>
        <arg0>2</arg0>
        </q0:listAvailableApplications>
      </soapenv:Body>

</soapenv:Envelope>
```

Şekil 2.11 App-proxy'den App-Center'a istek mesajı

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
<ns2:listAvailableApplicationsResponse
    xmlns:ns2="http://ws.app_manager.ce.yildiz.edu.tr/"
xmlns:ns3="http://www.yildiz.edu.tr/dap/app-ws">
<return>
   <ns3:application url="http://localhost:9080/app-manager/dl?id=2" />
   <ns3:application url="http://localhost:9080/app-manager/dl?id=1" />
</return>
</ns2:listAvailableApplicationsResponse>
</soap:Body>
</soap:Envelope>
```

Şekil 2.12 App-Center'dan App-Proxy'ye dönen yanıt

2.3.1.4 Crystall App

Android işletim sisteminde OSGi dinamik modül yönetim avantajlarını kullanmak için çeşitli projeler gerçekleştirlmiştir.(Android Music Project [9])

Crystall App bileşeni dinamik uygulama platformunun Android işletim sistemi üzerinde çalışabilmesini sağlayan proje kapsamında geliştirilmiş bileşendir. Bu bileşen altyapısında gömülü felix OSGi içericisini kullanmaktadır.(Apache Felix Documents [7])

Crystall App çalışırken dinamik sınıf yüklemesi yapabilmesi için Android işletim sistemi üzerinde /data/dalvik-cache folder'ının tüm yetkilerinin tüm kullanıcılara verilmesi gerekmektedir. Ayrıca indirilen bundle'ları yükleyebilmesi için /data/dap/bundles klasörü yaratılmalıdır. Ve yetkileri tüm kullanıcılara açılmalıdır.

Crystall App, Felix üzerinde yüklü olan App-Agent bundle sayesinde dinamik uygulama platformuyla iletişim kurabilmektedir. Android işletim sisteminde iletişim biçimi standart javadan farklı olduğundan dolayı app-agent.jar yanı sıra bu farklılıkları içeren app-agent-android.jar da kullanılmaktadır.App-agent.jar ve app-agent-infra.jar resource folder'ı altında yer almaktadır. (IPOJO on Android [8])

InfraManager arayüzü sayesinde OSGi'da tanımlı olan servisler ile Android Uygulamasının ana sınıfı olan Activity sınıfı ile bağlantı kurulabilmektedir.

```
public class InfraManager {
    protected Activity activity;
    public final void initActivity(Activity activity){
      this.activity = activity;
    }
}
```

Şekil 2.13 InfraManager Arayüzü

2.3.2 Dap Bileşen Konfigurasyonları

Dap uygulama platformunu oluşturan düğümler conf.properties, proxy.properties ve cache.properties olmak üzere 3 konfigurasyon dosyası kullanmaktadır. Conf.properties app-agent tarafından kullanıldığından dolayı her düğümde olması gereken bir dosyadır.proxy.properties ve cache.properties ise sadece App-Proxy'ler tarafından kullanılmaktadır.

2.3.2.1 Conf.properties dosyası

App-Agent adres, port ve encoding gibi bilgilerini conf.properties dosyasından almaktadır. Dosyanın içeriği Şekil 2.14'de gösterilmektedir.

```
agent.name=dev1
agent.listeningPort=15454
agent.mcast.address=230.0.0.0
agent.mcast.port=15001
agent.temp.folder=c:/dev/dapTemp
agent.encoding=UTF-8
```

Şekil 2.14 Conf.properties dosyası

Conf.properties dosyasındaki alanların anlamları :

- *agent.name* , App-Agent'ın bulunduğu ağa göre tek olması gereken ismidir.

- *agent.listeningPort* ,App-Agent ‘ın App-Proxy’den komutları almak için kullandığı port’tur.

- *agent.mcast.address* , App-Agent’in dahil olmak istediği App-Proxy’nin adresidir.

- *agent.mcast.port* , App-Agent’in dahil olmak istediği App-Proxy’nin port’tur.

- *Agent.temp.folder*, App-Agent’ın yüklemek istediği uygulama bileşenlerini kopyalandığı dizindir.

- *Agent.encoding* , App-Agent’in App-Proxy ile iletişimde kullandığı encoding’tir.

Crystall App uygulamasında conf.properties dosyası bundle’ın içinde yer almaktadır.

2.3.2.2 Proxy.properties

App-Proxy, kullanacağı adres, port ve dizinler proxy.properties dosyasında yer almaktadır. Proxy.properties dosyası içeriği Şekil 2.15’te gösterilmektedir .

```
proxy.address=127.0.0.1
proxy.port=4446
proxy.mcast.address=230.0.0.0
proxy.mcast.port=15001
proxy.encoding=UTF-8
proxy.apps.folder=C:/dev/proxyTemp/apps
proxy.auto.apps.folder=C:/dev/proxyTemp/auto_apps
proxy.caching.enabled=false
proxy.username=admin
proxy.password=password
proxy.wsurl=http://localhost:9080/app-manager/ws/appManager?wsdl
proxy.id=2
```

Şekil 2.15 Proxy.properties dosyası

Proxy.properties dosyasındaki alanların anlamları :

- *proxy.address*, App-Proxy'nin App-Agent'lar ile iletişim kuracağı ağ cihazının adresidir.

- *proxy.port*, App-Proxy'nin App-Agent'lar ile iletişim kullanacağı port'tur.

- *proxy.mcast.address*, App-Proxy multicast olarak yayın yapacağı adresdir.

- *proxy.mcast.port*, App-Proxy multicast için kullanacağı port'tur.

- *proxy.encoding* , App-Proxy'nin App-Agent ile iletişimde kullandığı encoding'tir.

- *proxy.apps.folder*, App-Proxy'nin App-center'dan yetki alarak yükleyebildiği uygulamaların tanım dosyaları olan dapxml'lerin bulunduğu dizindir.

- *proxy.auto.apps.folder*, App-Proxy'nin App-center'dan yetki alarak yükleyebildiği ve yüklenmesi zorunlu olan uygulamaların tanım dosyaları olan dapxml'lerin bulunduğu dizindir.

- *proxy.caching.enabled* , App-Proxy'nin App-Agent ile iletişimde kullandığı encoding'tir.

- *proxy.id*, App-Center'da tanımlı olan proxy id'sidir.

- *proxy.username ,* App-Manager'a bağlanmak için web servis kullanıcı adıdır.

- *proxy.password ,* App-Manager'a bağlanmak için web servis parolasıdır.

- *proxy.password ,* App-Manager üzerinde çalışan web servis url'dir.

2.3.2.3 Cache.properties

App-Proxy proxy.caching.enabled true oldugu takdirdei uygulamaların bileşenlerini kendi üzerinden OSGi İçericilere vermektedir.Gerekli konfigurasyon değerlerini cache.properties dosyasından almaktadır. Şekil 2.16'te cache.properties dosyasının içeriği gösterilmektedir.

```
cache.folder=C:/dev/proxyTemp/cache
cache.file=C:/dev/proxyTemp/cache.xml
cache.hostname=localhost
cache.port=8080
```

Şekil 2.16 Cache properties

Cache.properties dosyasındaki alanların anlamları :

- *cache.folder*, App-Proxy'nin uygulama bileşenlerini sakladığı dizindir.
- *cache.file*, App-Proxy'nin kendi üzerinde tuttuğu bileşenlerin listesinin tutulduğu dosyadır.
- *cache.hostname*, App-Proxy'nin hostname'dir.
- *cache.port*, App-Proxy kendi üzerinde tuttuğu bileşenleri yükleme isteği gönderilecek port'udur.

2.3.3 Dinamik Uygulama Platformu İletişim Protokolü

App-Agent ve App-Proxy arasındaki mesajlar XML tabanlıdır. App-agent'lar yüklemeleri yönetme amacıyla App-Proxy'ye mesaj gönderebilir, App-Proxy'ler de App-agent'lardaki yüklemeleri yönetme amacıyla mesaj gönderebilir.

App-Agent 2.3.1.1'de anlatılacağı üzere Register, List Applications, Retrieve Application ve Sync mesajlarını gönderebilir. App-Proxy ise 2.3.3.2'de değinileceği üzere Install Application, Uninstall Application ve Broadcast mesajlarını gönderebilir.

2.3.3.1 App-Agent Mesajları

Register

Multicast sonucunda alınan mesajdan App-Proxy'nin adresi elde edilir ve register mesajını gönderilir. App-Proxy register mesajının cevabında bir registration id döner. İstek mesajı ve geri dönen yanıt sırasıyla Şekil 2.17 ve Şekil 2.18'de gösterilmiştir.

```
<dapMessage version="1.0">
      <command>register</command>
      <agentName>e-commerce1</agentName>
      <agentListeningPort>5858</agentListeningPort>
</dapMessage>
```

Şekil 2.17 App-Agent'tan App-Proxy'ye Register mesaj isteği

```
<dapMessage version="1.0">
      <command>register</command>
      <agentame>e-commerce1</agentName>
      <response>00</response>
      <registrationId>3</registrationId>
</dapMessage>
```

Şekil 2.18 App-Proxy'den App-Agent'a Register mesaj yanıtı

List Applications

App-Agent'lar dahil oldukları App-Proxy'lerdeki uygulamaların listesini elde etmek için List Application mesajını kullanır. Aynı zamanda App-Proxy'lerde App-Agent üzerindeki uygulamaları listelemek için aynı komutu kullanabilir. İstek mesajı ve geri dönen yanıt sırasıyla Şekil 2.19 ve Şekil 2.20'te gösterilmiştir.

```
<dapMessage version="1.0">
    <command>list</command>
    <criteriaList>
      <criteria name="appName" operator="like" value="ecommerce"/>
      <criteria name="appVersion" operator="gt" value="1.0"/>
    </criteriaList>
</dapMessage>
```

Şekil 2.19 App-Agent'tan App-Proxy'ye List mesaj isteği

```
<dapMessage version="1.0">
    <command>list</command>
    <appList>
      <app name="ecommerce" version="2.0" />
      <app name="ecommerce" version="3.0" />
    </appList>
</dapMessage>
```

Şekil 2.20 App-Proxy'den App-Agent'a List mesaj yanıtı

Retrieve Application

Uygulama yüklemesi yapılabilmesi için App-Agent, App-Proxy'ye Retrieve Application mesajını gönderir. Bunun üzerine App-Proxy uygulamanın alt bileşenlerini içeren bir cevap döndürür. İstek mesajı ve geri dönen yanıt sırasıyla Şekil 2.21 ve Şekil 2.22'de gösterilmiştir.

```
<dapMessage version="1.0">
    <command>retrieve</command>
    <appName>e-commerce</appName>
    <appVersion>2.1</appVersion>
</dapMessage>
```

Şekil 2.21 App-Agent'tan App-Proxy'ye Retrieve mesaj isteği

```
<dapMessage version="1.0">
    <command>retrieve</command>
    <appName>e-commerce</appName>
    <appVersion>2.1</appVersion>
    <componentList>
       <component name="ecommerce-dao" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=2" >
         <actions>
           <action>install</action>
         </actions>
       </component>
       <component name="ecommerce-service" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=3" >
         <actions>
           <action>install</action>
```

```
          </actions>
        </component>
        <component name="ecommerce-model" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=4" >
          <actions>
            <action>install</action>
          </actions>
        </component>
        <component name="ecommerce-web" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=5" >
          <component-args>
            <component-arg>-XmaxSize=500M</component-arg>
            <component-arg>-XminSize=100M</component-arg>
          </component-args>
          <actions>
            <action>install</action>
            <action>start</action>
          </actions>
         </component>

      </componentList>
</dapMessage>
```

Şekil 2.22 App-Proxy'den App-Agent'a Retrieve mesaj yanıtı

Detail Application

App-Agent alt bileşenleri öğrenmek için Detail Application mesajını App-Proxy'ye gönderir. App-Proxy alt bileşenleri içeren cevap döndürür. İstek mesajı ve geri dönen yanıt sırasıyla Şekil 2.23 ve Şekil 2.24'de gösterilmiştir.

```
<dapMessage version="1.0">
    <command>detail</command>
    <appName>e-commerce</appName>
    <appVersion>2.1</appVersion>
</dapMessage>
```

Şekil 2.23 App-Agent'tan App-Proxy'ye Detail mesaj isteği

```
<dapMessage version="1.0">
    <command>detail</command>
    <appName>e-commerce</appName>
    <appVersion>2.1</appVersion>
    <componentList>
       <component name="ecommerce-dao" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=2" >
         <actions>
           <action>install</action>
         </actions>
       </component>
       <component name="ecommerce-service" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=3" >
         <actions>
           <action>install</action>
         </actions>
       </component>
       <component name="ecommerce-model" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=4" >
         <actions>
           <action>install</action>
         </actions>
       </component>
       <component name="ecommerce-web" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=5" >
         <component-args>
           <component-arg>-XmaxSize=500M</component-arg>
           <component-arg>-XminSize=100M</component-arg>
         </component-args>
         <actions>
           <action>install</action>
           <action>start</action>
         </actions>
        </component>
     </componentList>
</dapMessage>
```

Şekil 2.24 App-Proxy'den App-Agent'a Detail mesaj yanıtı

2.3.3.2 App-Proxy Mesajları

Install Application

App-proxy, App-Agent'lara uygulama yüklenmesi için install mesajı gönderir. İstek mesajı ve geri dönen yanıt sırasıyla Şekil 2.25 ve Şekil 2.26'de gösterilmiştir.

```
<dapMessage version="1.0">
    <command>install</command>
    <appName>e-commerce</appName>
    <appVersion>2.1</appVersion>
    <componentList>
       <component name="ecommerce-dao" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=2" >
         <actions>
           <action>install</action>
         </actions>
       </component>
       <component name="ecommerce-service" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=3" >
         <actions>
           <action>install</action>
         </actions>
       </component>
       <component name="ecommerce-model" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=4" >
         <actions> <action>install</action> </actions>
       </component>
       <component name="ecommerce-web" version="2.1"
url="http://10.0.0.30/app-agent/retrieveComponent?id=5" >
         <component-args>
           <component-arg>-XmaxSize=500M</component-arg>
           <component-arg>-XminSize=100M</component-arg>
         </component-args>
         <actions>
           <action>install</action>
           <action>start</action>
         </actions>
       </component>
     </componentList>
</dapMessage>
```

Şekil 2.25 App-Agent'tan App-Proxy'ye Install mesaj isteği

```
<dapMessage version="1.0">
    <command>install</command>
    <response>00</response>
</dapMessage>
```

Şekil 2.26 App-Proxy'den App-Agent'a Install mesaj yanıtı

App-Agent ve App-Proxy arası iletişimde request ve response aynı bağlantıyı (connection) kullanmaktadır. App-Proxy ve App-Agent Java 1.5 ile gelen NIO apisini kullanmaktadır.

BÖLÜM 3

ÖRNEK ÇALIŞMALAR

3.1 Web Uygulaması

Bu bölümde, OSGi ve Maven kullanılarak hazırlanmış bir örnek web uygulanmasının DAP uygulama modelinin oluşturulması ve DAP bileşenleri kullanılarak kurulumu gösterilecektir.

3.1.1 Web Uygulaması Bileşenleri

Web Uygulaması öğrenci ve öğretmenlerinin bilgilerini kaydetmeye yarayan bir uygulamadır. Web uygulamasını oluşturan bileşenler şunlardır:

- DynaView: Modüllerin OSGi servisi olarak arayüze kendilerini kaydettirdikleri servislerin arayüzlerini içeren bileşendir.
- BaseVaadin: Modüllerin ortak olarak kullanacağı Vaadin arayüz çatısının bileşenlerinin oluşturulduğu bileşendir. Vaadin çatısı OSGi ile entegre edilebilir bir web uygulama çatısı olması nedeniyle tercih edilmiştir.
- ModuleStudent: Öğrenci için kayıt işlemi yapılmasını sağlayan bileşendir.
- ModuleTeacher: Öğretmen için kayıt işlemi yapılmasını sağlayan bileşendir.

BaseVaadin, ModuleStudent ve ModuleTeacher bileşenlerinde servisleri export ve register etmek için OSGi 4.2 ile beraber gelen Declarative Services kullanılmıştır.(McAffer, Vanderlei, Archer [4])

3.1.1.1 DynaView

DynaView; Module, ModuleService ve ModuleServiceListener olmak üzere 3 arayüz içermektedir. Module arayüzü uygulamada her tab içinde olacak modül'ün özelliklerini taşımaktadır.

```
public interface Module<C> {
    public String getName();
    public C createComponent();
}
```

Şekil 3.1 Module Arayüzü

ModuleService uygulamada yeni bir module dahil olduğunda çağıralacak metotları içermekte ve gerekli Module servislerini içermektedir.

```
 public interface ModuleService<C> {
                    public void registerModule(Module<C> module);
    public void unregisterModule(Module<C> module);
    public List<Module<C>> getModules();
    public void addListener(ModuleServiceListener listener);
    public void removeListener(ModuleServiceListener listener);
}
```

Şekil 3.2 ModuleService Arayüzü

ModuleServiceListener ise, moduleService registerModule ve unregisterModule method'larının çağırılması için kullanılan observer rolündedir.(Gamma, Helm, Johnson, Vlissides [1])

```
 public interface ModuleServiceListener<C> {
    public void moduleRegistered(ModuleService<C> source, Module<C>
module);
    public void moduleUnregistered(ModuleService<C> source, Module<C>
module);}
```

Şekil 3.3 ModuleServiceListener Arayüzü

Module, ModuleService ve ModuleServiceListener arayüzleri tasarlanırken sadece Vaadin çatısına özgü olmaması için generic yapı kullanılmıştır.

3.1.1.2 BaseVaadin

BaseVaadin, Vaadin çatısının genel bileşenlerinin oluşturulduğu bileşendir. ModuleStudent ve ModuleTeacher BaseVaadin bileşenin export ettiği servisleri kullanmaktadırlar.

```
<?xml version="1.0"?>
<component name="vaadin-moduleService" immediate="true">
    <implementation
class="tr.edu.yildiz.basevaadin.service.impl.ModuleServiceImpl"/>
    <service>
      <provide
interface="tr.edu.yildiz.basevaadin.service.ModuleService"/>
    </service>
</component>
<component name="vaadin-servletService" immediate="true" >
    <implementation
class="tr.edu.yildiz.basevaadin.service.impl.ServletServiceImpl"/>
    <service>
      <provide
interface="tr.edu.yildiz.basevaadin.service.ServletService"/>
    </service>
    <reference name="vaadin-moduleService"
     interface="tr.edu.yildiz.basevaadin.service.ModuleService"
     bind="setModuleService"
     unbind="unSetModuleService" cardinality="1..1"
     policy="dynamic"/>
    <reference name="http"
     interface="org.osgi.service.http.HttpService"
     bind="registerServlet"
     unbind="unRegisterServlet"
     cardinality="1..1"
     policy="dynamic"/>

</component>
```

Şekil 3.4 BaseVaadin Declarative Services XML

Şekil 3.4'de BaseVaadin'in export ettiği tr.edu.yildiz.service.ModuleService ve register ettiği tr.edu.yildiz.basevaadin.service.impl.ServletService servisi gösterilmektedir.

3.1.1.3 ModuleStudent

ModuleStudent öğrenci bilgi girişlerinin yapılacağı önyüz bileşenidir.

```
<?xml version="1.0"?>
<component name="vaadin-moduleStudent">
    <implementation
class="tr.edu.yildiz.module1.vaadin.ModuleStudent"/>
    <reference name="vaadin-moduleService"
     interface="tr.edu.yildiz.basevaadin.service.ModuleService"
     bind="setModuleService" unbind="unsetModuleService"
     cardinality="1..1" policy="dynamic"/>
</component>
```

Şekil 3.5 ModuleStudent Declarative Services XML

Şekil 3.5'de ModuleStudent'in export ettiği tr.edu.yildiz.modulestudent.vaadin.ModuleStudent servisi ve BaseVaadin bileşeninin export ettiği tr.edu.yildiz.basevaadin.service.ModuleService referans edilmiştir.

3.1.1.4 ModuleTeacher

ModuleTeacher öğretmen bilgi girişlerinin yapılacağı önyüz bileşenidir.

```
<?xml version="1.0"?>
<component name="vaadin-moduleTeacher">
    <implementation
class="tr.edu.yildiz.moduleteacher.vaadin.ModuleTeacher"/>
    <reference name="vaadin-moduleService"
     interface="tr.edu.yildiz.basevaadin.service.ModuleService"
     bind="setModuleService" unbind="unsetModuleService"
     cardinality="1..1" policy="dynamic"/>
</component>
```

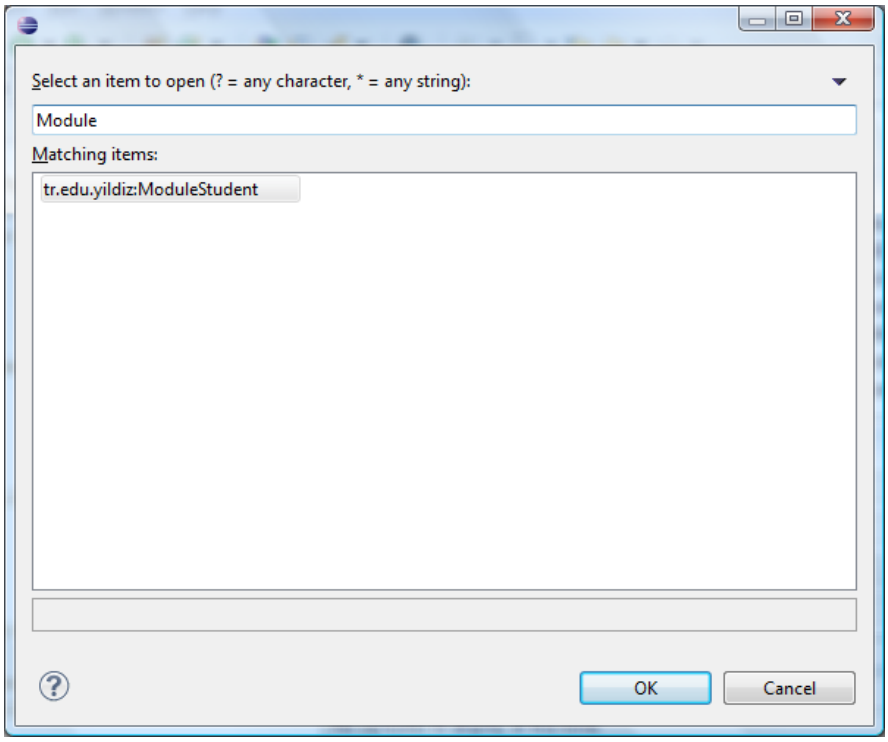
Şekil 3.6 ModuleTeacher Declarative Services XML

Şekil 3.6'da ModuleTeacher'in export ettiği tr.edu.yildiz.moduleteacher.vaadin.ModuleTeacher servisi ve BaseVaadin bileşeninin export ettiği tr.edu.yildiz.basevaadin.service.ModuleService referans edilmiştir.

3.1.2 Kurulum

Web Uygulamasının DAP uygulaması olması için bileşenlerinin tanımlı olduğu dapxml dosyası oluşturulmalıdır. Eclipse Dap Eklentisi oluturularak bu dosya oluşturulabilinir.

Dapxml dosyası oluşturulduktan sonra ModuleStudent ya da ModuleTeacher'nin eklenmesi diğer bileşenleri de bağımlılıkları olduğundan dolayı dapxml'e dahil edecektir.



Şekil 3.7 Dapxml'e ModuleStudent'in eklenmesi

ModuleStudent'ın eklenmesi sonucunda BaseVaadin, DynaView ve birçok web kütüphanesi projeye dahil olacaktır.

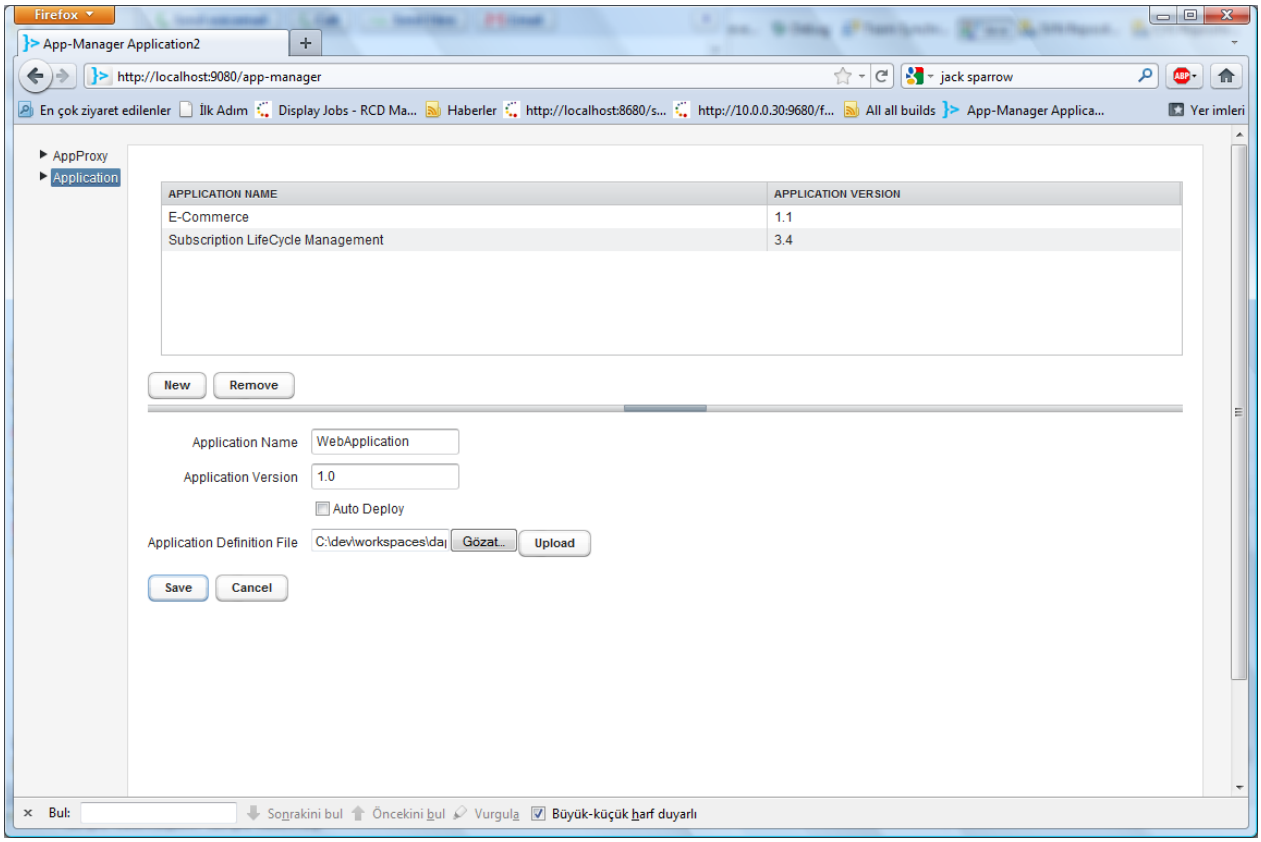
Dapxml kullanılarak OSGi içerici üstüne web uygulaması kurulumu iki şekilde yapılabilinir: İlk yol, Şekil 3.8'de görüldüğü şekilde, komut satırından install komutu vermektir. Bu komutun aldığı f parametresi, takip eden değerin dapxml dosyasının yolunu gösterdiği anlamındadır.

App-install –f C:\dev\workspaces\dap_workspace\sample-vaadin\dap\sample-vaadin.dapxml

Şekil 3.8 Dapxml'de komut satırı kullanılarak uygulama kurulumu

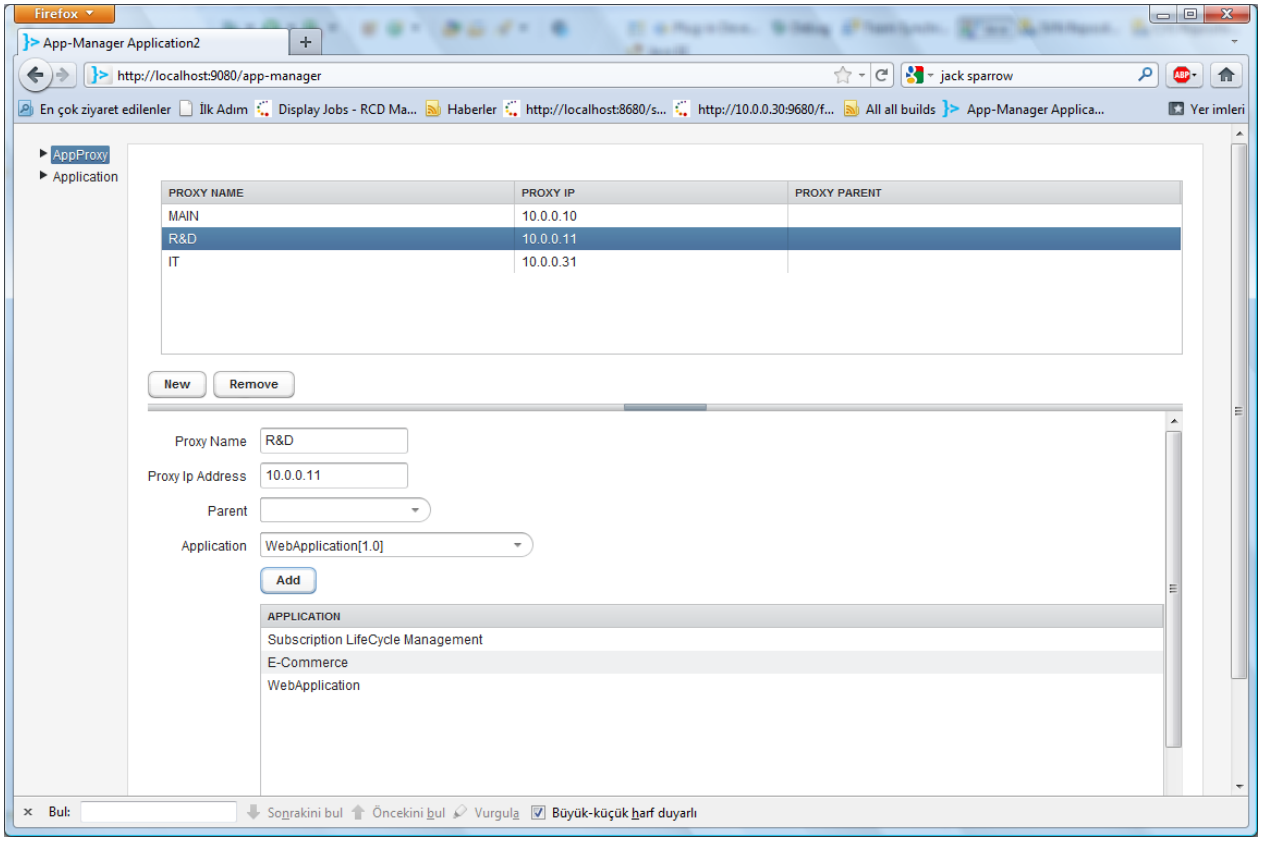
Dapxml'de uygulama kurulumunun ikinci yolu ise, kurulumun App-Proxy üzerinden yapılmasıdır. İşlem adımları, nasıl yapıldıklarını gösteren ekran çıktıları ile birlikte, aşağıda tanımlanmıştır:

App-Manager uygulamasında Application olarak uygulama eklenmeli ve dapxml dosyası upload edilmelidir.



Şekil 3.9 Uygulamanın App-Manager'a eklenmesi

App-Manager'da App-Proxy'nin Application listesine eklenmelidir.



Şekil 3.10 Uygulamanın gerekli App-Proxy'lerle eşleştirilmesi

Eklendiği App-Proxy'ye bağlanan App-agent tarafından OSGi içericiye kurulumu yapılacaktır.

3.2 Mobil Uygulama

Bu bölümde, OSGi kullanılarak hazırlanmış mobil uygulanmanın DAP uygulama modelinin oluşturulması ve DAP bileşenleri kullanılarak kurulumu gösterilecektir.

3.2.1 Mobil Uygulama Bileşenleri

Mobil uygulamalar diğer uygulama türleriyle kıyaslandığında daha küçük olduğundan dolayı az sayıda bileşen gerektirmektedir. Örnek uygulama bu nedenle tek bileşenden oluşmaktadır.

3.2.1.1 SampleGui

SampleGui OSGi altyapısı kullanılarak geliştirilmiş bir mobil flightboard uygulamasıdır. Uçuşlar mobil uygulama sayesinde listelenebilmekte, filtrelenebilmekte ve detayı görüntülenebilmektedir.

Mobil uygulamanın dinamic uygulama platformunda çalışabilmesi için ViewFactory arayüzünü gerçeklemesi gerekmektedir.

```
public interface ViewFactory {
    public View create(Activity activity);
}
```

Şekil 3.11 ViewFactory Arayüzü

ViewFactory arayüzünün create metodu döndüğü View objesi Android işletim sistemi tarafından render edilmektedir.

Mobil uygulama android işletim sistemin önyüz bileşenlerini kullandığından dolayı android java paketlerinin manifest dosyasında import edilmesi gerekmektedir. Örnek uygulamada android.app, android.content, android.database, android.text, android.view, android.widget gibi android paketleri import edilmiştir.

Uygulama aynı zamanda OSGi ve Dinamik uygulama platforumunun java paketlerini de manifest dosyasında import etmelidir. Bu doğrultu da org.osgi.framework;version="1.3", tr.edu.yildiz.crystalapp.view paketleri uygulamaya import edilmiştir.

Android işletim sistemi bilinen Sun Sanal Makinasından farklı olarak Dalvik Sanal Makinası çalıştırdığından dolayı uygulamanın jar'ı android işletim sistemi için çalıştırılabilir bir dosya olmayacaktır. Bu nedenle iki işlem yapılmalıdır: Öncelikle dosya android tool'ları kullanılarak binary class'ların sıkıştırılmış hali olan Dex formatına çevrilmelidir.(Erhinger [6]) İkinci olarak da, önceki adımda oluşan classes.dex dosyası,

jar dosyasının içine koyulmalıdır. İlk işlem için *Dx –dex –output=classes.dex sampleGui-0.0.1-SNAPSHOT.jar* komutu, ikinci işlem ise *Aapt sampleGui-0.0.1-SNAPSHOT.jar classes.dex* komutu kullanılabilir.

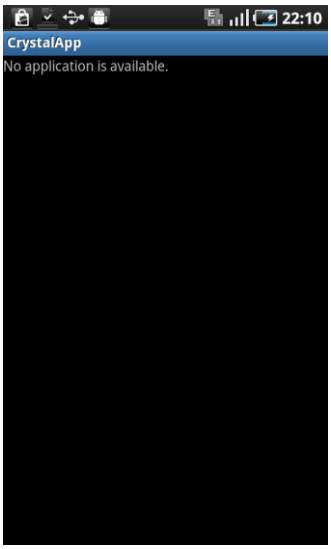
3.2.2 Kurulum

Uygulama'nın DAP'a uyumlu hale gelmesi için dapxml'inin oluşturulması gerekmektedir. Bu dosya uygulama olarak App-Manager'a auto-deploy seçeneği seçili olarak yüklenmeli ve gerekli app-proxy ile eşleştirilmelidir. Bölüm 3.1.2'de web uygulaması app-manager'a deployment'ı gösterilmiştir. Mobil uygulama için aynı adımlar takip edilebilir.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<appDefinition xmlns="http://www.yildiz.edu.tr/dap/app-comm">
 <app version="0.0.1" name="AppProxy"/>
 <componentList>
 <component url="ftp://192.168.1.147/repo/sampleGui.jar"
version="2.0.4" name="sampleGui">
 <actions>
 <action>install</action>
 <action>start</action>
 </actions>
 </component>
 </componentList>
</appDefinition>
```

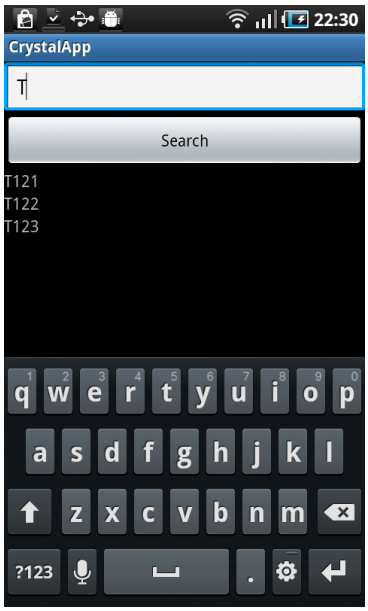
Şekil 3.12 SampleGui dapxml dosyası

CrystallApps uygulaması yüklü olan android işletim sistemi çalıştıran mobil istemci yerel ağa dahil olduğunda uygulama kendiliğinden yüklenecektir. CrystallApp açıldığında bir App-proxy'ye bağlanıp auto deploy bir uygulama yükleyinceye kadar yüklenebilecek uygun bir uygulama olmadığını gösteren bir ekran gösterilecektir.



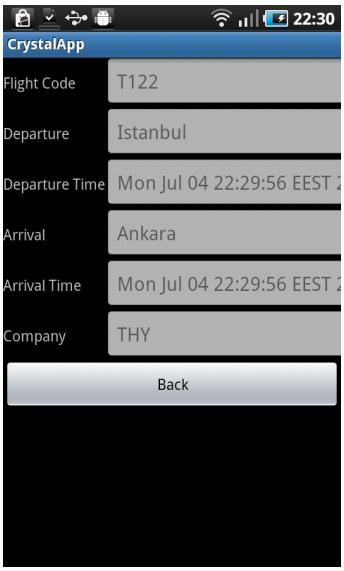
Şekil 3.13 CrystallApp uygulama yüklenmemiş hali

SampleGui yüklendikten sonra uçuşların listesi gösterilecektir.



Şekil 3.14 CrystallApp SampleGui yüklendikten sonraki hali

Listeden bir uçuş seçildiğinde, uçuşun detayı görüntülenebilir.



Şekil 3.15 SampleGui'de Uçuş Detayı

BÖLÜM 4

SONUÇ VE ÖNERİLER

4.1 Diğer Modeller ile Karşılaştırması

DAP uygulama modeli sayesinde, OSGi ile geliştirilen uygulamaların yönetiminin kolaylaştırılması hedeflenmiştir. DAP uygulama modeli şu avantajlara sahiptir:

- Equinox ve Felix üzerinde çalışabilmektedir.
- Eclipse üzerinde geliştirme eklentileri ile uygulama modeli kolayca oluşturabilmektedir.
- Maven kullanarak kütüphaneler elde edilmektedir.
- Uygulamaların ağ merkezcil olarak yüklenmeleri sağlanmaktadır.
- App-Center bileşeni ile merkezi uygulama yetkilendirme ve yönetimi sağlanmaktadır.
- Uygulamaları yönetildiği merkezden OSGi içericilere komut göndererek ve OSGi içericilerden uygulamaların yönetildiği merkeze komut göndererek uygulama yükleme imkanı sunmaktadır.

Çizelge 4.1 DAP'ın Önceki Çalışmalar ile Karşılaştırması

Özellikler	**DAP**	**Spring Par**	**Apache Ace**	**Apache Aries**
Farklı OSGi içericilerde	Var (Equinox ve Felix desteği mevcut)	Yok (Sadece Spring DM Server üzerinde çalışabilir)	Var (Equinox ve Felix desteği mevcut)	Var (Equinox ve Felix desteği
Geliştirme Ortamı Desteği	Eclipse üzerinde geliştirme eklentisi mevcut.	Var (Eclipse ortamında geliştirilmiş bir geliştrme arayüzü mevcut)	Yok (Geliştirme desteği bulunmamakta)	Yok (Geliştirme desteği bulunmamakta)
Tek dosya ile yükleme	Var (dapxml dosyası ile yükleme yapılabilmekedir)	Var (Par dosyası ile yükleme yapılabilmekte)	Var (Eba dosyası ile yükleme yapılabilmekte.)	Yok (Maven yardımıyla kurulum yapılabilmekte)
Mobil Uygulama desteği	Var(Android işletim sistemi)	Yok	Yok	Yok
P2 ve OBR desteği	Var (Maven desteği var.)	Yok (Spring kendi repository'lerini kullanmakta)	Var	Var

Dap uygulama modelinin, aşağıdan yukarı uygulama yüklemesini desteklemesi ve merkezi uygulama yönetimini sağlaması özellikleri sayesinde, mobil uygulamalarda da kullanılabilmesi hedeflenmiştir. Ancak Android işletim sisteminin resmi olarak OSGi desteklememesi nedeniyle işletim sistemi üzerinde super user aktif hale getirilip, data/dalvik-cache dizinine tam yetki verilerek uygulama çalışır hale getirilmiştir.

4.2 Araştırma ve Geliştirmeye Açık Alanlar

Dinamik uygulama platformu, mobil cihazlarda ve bilgisayarlarda çalışan bir uygulama alt yapısı olduğundan dolayı, kendi içinde bir izin yapısı geliştirilerek dinamik

uygulamaların kaynak kullanım kısıtlaması sağlanabilir. Proxy ile agent'lar arasında iletişimin şifreli olarak yapılması ve güçlü proxy doğrulama sistemi geliştirilmesi, dinamik uygulama platformunu daha güvenli olmasını sağlayacaktır.

Dinamik uygulama platformu altyapı olarak teknoloji bağımsız olmasından dolayı farklı işletim sistemlerinde de (IOS,Windows Phone) gerçeklenebilir. Farklı işletim sistemleri için app-agent'lar geliştirildiği takdirde dinamik uygulama platformunun çoklu işletim sistemi desteği de arttırılmış olacaktır.

KAYNAKLAR

[1] Gamma E., Helm R., Johnson R. ve Vlissides J., (1994). Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

[2] Mak G. ve Rubio D., (2009). SpringSource DM Server, Apress Book Co., New York.

[3] Clayberg E. ve Rubel D., (2006). Eclipse: Building Commercial-Quality Plug-ins, Pearson Education Inc., Boston.

[4] McAffer J., Vanderlei P. ve Archer S., (2010). OSGi and Equinox: Creating Highly Modular Java Systems, Pearson Education Inc., Boston.

[5] Goetz B., (2006). Java Concurrency in Practice, Pearson Education Inc., Boston.

[6] Erhinger D., (2011).“The Dalvik Virtual Machine Architecture”, Mart 2011

[7] Apache Felix Documents, http://felix.apache.org/site/documentation.html, 05 Ağustos 2011.

[8] IPOJO on Android, http://ipojo-dark-side.blogspot.com/2008/10/ipojo-on-android.html, 05 Ağustos 2011.

[9] Android Music Project, http://ist-music.berlios.de/site/middleware-android.html, 05 Ağustos 2011.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : İlker Rıza ERNAS

Doğum Tarihi ve Yeri : 25/02/1984 Aydın

Yabancı Dili : İngilizce , Almanca

E-posta : ilker.riza.ernas@ericsson.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh.	Kocaeli Üniversitesi	2006
Lise	Fen-Matematik	Milli Piyango Anadolu Lisesi	2002

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2010	Ericsson	Servis Mühendisi
2008	Cybersoft	Geliştirici
2006	Nexum Boğaziçi	Analist Geliştirici