Konuşma Yoluyla Etkileşilebilen Mobil Yazılımların Geliştirilmesine Yönelik Anotasyon Yönelimli Bir Yazılım Çerçevesi

Çağdaş Evren Gerede[1]

TOBB Ekonomi ve Teknoloji Üniversitesi, Ankara, Türkiye,
cegerede@etu.edu.tr,
WWW sayfası: http://cegerede.etu.edu.tr/

Özet Bu makalede Android işletim sistemi ile çalışan cihazlarda konuşma yoluyla kontrol edilebilen yazılımların yapımını ve bakımını kolaylaştıran bir yazılım çerçevesi önerilmektedir. Günümüzde konuşma yoluyla da etkileşilebilen bir yazılım geliştirmek hem birçok teknik konuda uzmanlık gerektirmekte, hem de pahalı ve zaman alan bir süreci içermektedir. Dolayısıyla birçok yazılım böyle bir arayüz desteği sunamamaktadır. Önermekte olduğumuz yazılım çerçevesi ile halihazırda konuşularak kullanılamayan yazılımların konuşularak da kullanılabilir hale getirilmesi kolaylaşacaktır. Standart grafiksel kullanıcı arayüzlerine sahip yazılımların tüm özelliklerinin konuşularak da etkileşilebilir hale getirilmesi bedensel ve görme engelli kullanıcılar için büyük önem taşımaktadır. Çünkü aksi durumda bu yazılımlar bu tür kullanıcılar tarafından kullanılamaz hale gelmektedir. Bu konu aynı zamanda içinde bulundukları durumdan kaynaklanarak geçici süre ile bedensel veya görme engelli hale gelebilecek tüm kullanıcılar için de önemlidir. Örneğin, arabasını süren bir kişinin, radyosu veya telefonu ile etkileşmek istediğinde direksiyonda olan ellerini kullanamadığından bedensel engelli, cihaz ekranına bakamadığından da görme engelli halde olduğu söylenebilir.

Anahtar Kelimeler yazılım çerçeveleri, engelleri kaldıran destek teknolojileri, konuşma yoluyla etkileşilebilen yazılımlar, Android cihazlarda yazılım geliştirme

1 Giriş

Son yıllarda akıllı telefon ve tablet gibi mobil cihazların kullanımı giderek yaygınlaştı. Birçok alanda kullanılmak üzere mobil uygulamalar üretilmeye ve App-Store ve Google Play gibi dijital uygulama pazarlarından satılmaya başlandı. Bununla beraber mobil işletim sistemi üreticileri, mobil cihazlarda sunulan sanal ekran klavyesi üzerinden veri girişine ek olarak alternatif kullanıcı deneyimini iyileştiren yeni veri girişi yöntemleri arayışına gittiler. Örneğin Apple şirketi Siri[6] adlı uygulama ile beraber kullanıcının kısıtlı sayıda da olsa bazı sesli sorgulamalar yapmasını sağladı. Benzer olarak Google şirketi de Google Now[3] uygulaması ile kullanıcıların google.com web sitesi yerine doğrudan Android cihazlara

konuşarak web üzerinde sorgulamalar yapabilmesini hedefledi. Yine Microsoft Cortana [1] adlı ürünü ile benzer bir yaklaşıma gitti. Bu ürünler sesli sorguları sesten metine çevirme teknikleri kullanarak metine dönüştürmekteler ve çıkan metni de metin kabul eden sorgu sistemlerine yönlendirmekteler. Bu ürünler istenilen düzeyde bir kullanışlılığa henüz ulaşamasalar da her geçen gün desteklediklerini sorgu tiplerini ve tanıyabildikleri dil ve aksan sayılarını arttırmaktalar.

Mobil uygulamaların sesli komutlara cevap verebilmesi özellikle görme engelli veya bedensel engelli (felç geçirmiş, el titremesi olan, ellerini kaybetmiş vb.) kullanıcılar için büyük önem taşımaktadır. Çünkü bu kullanıcıların mobil cihazları dokunmatik ekranlarına dokunarak yönetmeleri mümkün değildir. Bunların yanında engelsiz kullanıcılar da zaman zaman geçici engelli hale gelebilmekte ve dokunarak cihazlarla etkileşim kurma yeteneklerini geçici bir süreliğine kaybedebilmektedir (örneğin elinden ameliyat geçiren bir hasta veya yemek yaparken elleri dolu olan bir aşçı).

Günümüzde mobil cihazlarda 4 tip sesli sorgulama desteklenmeye çalışılmaktadır. Birincisi sorgu metniyle web üzerinde geleneksel olarak metin eşleşmesine bakılarak arama yapmak (örneğin içinde "Ankara" geçen web sayfalarını bulmak). İkincisi İnternet ve diğer kaynaklardan toplanan verilerle oluşturulmuş veri bankalarında anlamsal (İng. semantic) aramalar yapmak (örneğin "Ankara'da bugün sıcaklık kaç derece?" veya "Gün batımı saat kaçta olacak?"). Üçüncüsü mobil cihazı ve içindeki uygulamaları sesle aktive eden sorgular (örneğin "YouTube uygulamasını aç" veya "Veli'nin ev telefonunu ara" vb. komutlar). Dördüncüsü mobil uygulamalarla uzun soluklu etkileşmeyi gerektirebilecek sorgular (örneğin, "Yarın için bir toplantı yarat" komutuna karşılık, cihazdaki bir takvim uygulamasının toplantının hangi saatte olduğunu kullanıcıya sorması; kullanıcının toplantı saatini uygulamaya söyledikten sonra, uygulamanın uygun toplantı odalarını kullanıcıya sunması; kullanıcının odalardan birini konuşarak seçmesi vb.)

Yukarıda saydığımız sorgulama kategorilerinden son kategorinin gerçekleştirilmesinde diğerlerine göre pek fazla ilerleme kaydedilmemiştir. Bunun iki önemli sebebi var. Birincisi Google, Apple ve Microsoft gibi mobil işletim sistemi üreticilerinin (Sırasıyla Android OS / Nexus serisi; iOS / iPad-iPhone serisi; Windows Mobile / Surface-Lümia serisi) öncelikleri bu yönde olmamıştır. Çünkü son yıllarda mobil cihazlardan gelen web üzerinde arama trafiğinde ciddi artışlar meydana gelmiş ve giderek büyüyen bu pazarda pay sahibi olmak isteyen bu şirketler geliştirdikleri mobil işletim sistemlerine sesli sorgulama yapma özelliğini ekleme yoluna gitmişlerdir. Bu şekilde mobil cihaz kullanıcısından gelen arama isteklerinin bir kısmını kendi arama alt yapılarına yönlendirebilmeyi hedeflemişlerdir (Bakınız, ilk 2 kategori).

Mobil cihazlarla uzun soluklu sesle etkileşim kategorisinde fazla bir ilerleme kaydedilememesinin bir diğer sebebi ise mobil uygulamaların genel olarak sesli komutları algılayacak şekilde geliştirilmemiş olmalarıdır. Örneğin, iOS ve Android cihazlarda Google Calendar uygulamasına "bana bugünkü toplantılarımı göster" veya "Ali ile saat 3'teki toplantımı 4'e al" gibi sesli komutlar veremeyiz. Çünkü uygulama kaynak kodu bu tür sesli isteklere cevap verecek şekilde yazılmamıştır. Dolayısıyla işletim sistemi seviyesinde çalışan Siri ve Google Now

vb. bir uygulama işletim sisteminin doğrudan kontrolü altında olan komutları gerçekleştirebilirken (örneğin "google calendar uygulamasını aç" komutu), cihazdaki yüklü uygulamaların sınırları içine giren komutları, uygulamanın desteği olmadan yerine getirememektedirler.

1.1 Mobil Uygulamalarla Konuşarak Etkileşmek

Mobil uygulamalarla konuşarak etkileşmeyi sağlamak için günümüzde 2 tür yazılım geliştirme yaklaşımı söz konusudur.

Uygulamalara Konuşma Arayüzleri Geliştirilmesi Uygulama geliştiricilerin uygulamalarında dokunma ile etkileşilen arayüzlerin yanında, kullanıcının konuşarak etkileşebileceği yeni arayüzler geliştirmeleridir. Bu yaklaşımda kullanıcı uygulama ile konuşarak etkileşebilir. Bu etkileşme uzun soluklu olabilir. Uygulama anlamadığı sorgularda ya da muğlak olan durumlarda kullanıcıya konuya açıklık getirmesi için sorular sorabilir; seçenekler olduğunda kullanıcıya bu seçenekleri sesle sunabilir; eksik olan bir veri gerektiğinde bu veriyi kullanıcıdan sesle isteyebilir. Bu yaklaşımın avantajı kullanıcı deneyiminin en ideal seviyede olmasıdır. Uygulama kullanıcının kendisiyle dokunmadan farklı bir yöntemle iletişim kurduğunu bilir ve ona uygun bir iletişim yöntemi kullanır.

Bu yaklaşım hem parasal hem de zamansal olarak uygulama geliştiriciye ciddi geliştirme maliyetleri getirmektedir. Ayrıca bu yaklaşımın gerçeklenmesi birçok dalda tecrübe gerektirmektedir (konuşma tanıma motorlarının kullanımı, konuşma arayüzü gerçeklenmesi, doğal dil işleme, makine öğrenmesi vb.) Bu yaklaşımın bir alt kümesi olarak bazen uygulamalarda kısıtlı sayıda özellik için özel olarak sesli komutlar desteklenmiştir (örneğin bazı mobil cihazlara "Yarın sabah sekize çalar saati kur" komutu verildiğinde cihazdaki çalar saat uygulaması bu komutu gerçekleştirebilmektedir). Fakat engelli kullanıcılar için bir uygulamadaki bazı özelliklerin değil bütün özelliklerin kapsanması gerekir. Zira engeli olmayan bir kullanıcı konuşma yoluyla gerçekleştiremediği istekleri, her zaman dokunmatik arayüz alternatifine dönerek yapmayı deneyebilir ama engelli kullanıcı için böyle bir alternatif söz konusu değildir.

Uygulamalara Dokunma Eylemlerini Enjekte Edebilen Ayrıcalıklı Bir Uygulama Geliştirmek Bu yöntemde uygulamaların grafiksel arayüzleri ile kullanıcı arasında genel bir yardımcı uygulama inşa edilir. Bu uygulama engelli kullanıcının isteklerini konuşma yoluyla alır. Daha sonra bu istekleri kullanılmakta olan asıl uygulamaya işletim sisteminin yardımıyla sanal dokunma eylemleri olarak enjekte eder[17]. Bunu yapabilmesi için kullanıcının bu yardımcı uygulamayı ayrıcalıklı (İng. privileged) hale getirmesi gerekir. Bu sırada asıl uygulama kullanıcının engelli olup olmadığını bilmez. Bu yaklaşımın avantajı uygulama geliştiricilerden herhangi önemli bir efor gerektirmiyor olmasıdır. Fakat bu yaklaşım pratikte hem kötü bir kullanıcı deneyimi ortaya çıkarmakta hem de uygulamaları yalnızca kısmi olarak konuşmayla kontrol edilebilir hale getirebilmektedir. İlgili çalışmalar kısmında bu konudan daha detaylı bahsedeceğiz.

Bu makalede Android uygulamaların konuşularak da etkileşilebilir hale getirilebilmesini kolaylaştırmak için bir yazılım çerçevesi öneriyoruz. Bu çerçeve, konuşma yoluyla etkileşim için bir uygulamanın ihtiyaç duyacağı gerekli altyapıyı uygulama geliştiriciye sağlayacaktır. Bu sayede hem uygulama geliştirme maliyetleri azaltılmakta, hem de kullanıcılar için iyi bir kullanıcı deneyimi ortaya çıkarılmak istenmektedir.

2 İlgili Çalışmalar

Engelli kullanıcılara yönelik olarak Google tarafından JustSpeak[17] adında prototip bir Android uygulaması geliştirilmiştir. JustSpeak Android cihazlarda işletim sisteminin desteklediği imtiyazlı erişilebilirlik arayüzleri (İng. accessibility APIs) vasıtasıyla o anda ekranda görünen grafiksel kullanıcı arayüzü hiyerarşisini işletim sisteminden alabilmektedir (örneğin ekranda basılabilir kaç düğme var ve bunların etiketleri neler bilgisi)[13]. JustSpeak kullanıcıdan gelen sesli komutu ekrandaki seçeneklerden birine eşler (örneğin ekranda "tamam" etiketli bir düğme varsa, "tamam düğmesine bas" komutunu bu düğmenin basılması eylemine eşler). Daha sonra ekrandaki ilgili nesneye dokunulmuş olduğu sinyalini işletim sistemi vasıtasıyla uygulamaya enjekte eder. Diğer bir değişle JustSpeak kullanıcının sesli komutlarını sanal dokunma işlemlerine dönüştürür. Bunun yanında yukarı/aşağı kaydırma (İng. scroll up/down) gibi bazı standart kullanıcı arayüzü jestleri de (İng. gesture) sanal olarak JustSpeak tarafından uygulamaya enjekte edilebilmektedir.

JustSpeak, kullanıcı çalışmaları ile analiz edildiğinde, ekranda görülen seçenekleri algılamakta zaman zaman başarısız olduğu görülmektedir. Çünkü bu işlemin doğru çalışabilmesi için uygulama geliştiricilerin ekranda çizdikleri grafiksel kullanıcı arayüzü nesnelerini kod içerisinde erişilebilirlik etiketleri (İng. accessibility labels) ile etiketlemeleri gerekmektedir. Birçok uygulama geliştiricisi bu konuyu ya önemsemediğinden ya da bilmediğinden atlamaktadır. Bu da JustSpeak uygulamasının ekranda görünen seçenekleri algılama başarı oranını düşürmektedir.

JustSpeak uygulaması ile ilgili bir diğer problem ise kumanda edilmeye çalışılan uygulamanın kullanıcının sesi yoluyla kullanılmaya çalışıldığından haberdar olmamasıdır. Bu yüzden uygulama kullanıcıya sesli etkileşime daha uygun akışlar veya seçenekler sağlamaya çalışmaz. Örneğin YouTube'ün Android uygulamasında bir video seyrederken uygulamaya "Altyazıyı aç" diye bir komut veremeyiz. Çünkü altyazı seçeneği ancak "Ayarlar" penceresinin "Altyazı Ayarları" alt menüsünden açıp kapatılabilmektedir. Video çalarken altyazı ayarlarını açma kapama ile ilgili bir seçenek ekranda gösterilmediği için JustSpeak uygulaması kullanıcının "Altyazıyı aç" komutunu uygulanabilir bir komut olarak algılayamaz. Böyle bir durumda engelli kullanıcı yapmak istediği işleme önkoşul işlemleri tek tek JustSpeak'e yaptırmak zorunda kalır. Bu örnekte kullanıcı JustSpeak'e sırasıyla şu komutları vermelidir: "Ayarlara git", "Altyazı ayarlarına git", "Altyazıyı aç", "Geriye git", "Geriye git", "Oynata bas". Bu durum iki nedenden ötürü kusurlu bir kullanıcı deneyimi ortaya çıkarır. Birincisi kullanıcının her yapmak istediği işlem için bu işleme önkoşul işlemleri bildiğini varsayamayız (örneğin kullanıcı altyazı

ayarlarının uygulamada hangi pencere ve menüler altından açılıp kapandığını bilemeyebilir). İkincisi böyle uzun bir komut zincirini gerçekleştirmek engelli kullanıcının dakikalarını alabilir. Çünkü ortalama uzunlukta bir komutun kullanıcı tarafından söylenmesinden JustSpeak tarafından gerçekleştirilmesine 5 ila 10 saniye geçmektedir. Bu durumu daha da kötüleştiren kullanıcının aynı komutu birden çok tekrar etmek zorunda kalabilmesidir. Çünkü bazen JustSpeak söylenen sesli komutu metine çevirmede hatalar da yapabilmektedir. JustSpeak teknolojisinin beta sürümü Nisan 2016'da Voice Access ismiyle kullanıcıların kullanımına açılmıştır[7].

Android cihazlarda konuşmayı tanıma/konuşmayı metine çevirme (İng. speech recognition/speech-to-text) teknolojisinin son durumuna baktığımızda, Google tarafından çok önemli ilerlemeler kaydedildiğini görmekteyiz. Örneğin [11]'deki çalışmada derin sinir ağları (İng. deep neural networks) kullanılarak, bu problem bir mobil cihaz üzerinde gerçek zamanlı olarak %15 civarı bir hata oranı ile çözülebilmiştir. [12]'de ise hata oranları %15'in altına indirilebilmiştir. Bu gelişmeler Google'ın destek verdiği Android işletim sistemine sürekli olarak yansıtılmakta ve geliştiricilerin kullanımına açılmaktadır. Örneğin Android işletim sisteminin Marshmallow sürümü ile beraber konuşma tanıma işleminin çevrim dışı da yapılabilmesine olanak verilmiştir.

Konumuzla ilgili konulardan bir diğeri de Java programlama dilinde anotasyon kavramıdır. Anotasyon (İng. "annotation") kaynak koda eklenebilen bir çeşit sözdizimsel üstveri (İng. syntactic metadata) etiketidir. Sınıf, metot, değişken ve parametre tanımlarına bu tür etiketler eklenebilir. Bu etiketler sınıf dosyalarında tutulabilir ve bu sayede çalışma zamanında bunlara yansıma (İng. reflection) yoluyla ulaşılabilir. Java kaynak kodlarda sıkça rastlanan anotasyonlara şöyle örnekler verebiliriz: `Override`, `Author`, `SuppressWarnings`[5]. Android uygulama geliştirme platformunda da geliştirme faaliyetlerini iyileştirmek için `UiThread`, `Nullable`, `NonNull` gibi birçok yeni anotasyon geliştiricilerin kullanımına sunulmuştur[4]. Geliştiriciler aynı zamanda kendi anotasyon tiplerini de tanımlayabilmektedir[2].

Yazılım geliştirmede nesne yönelimli yazılım çerçevelerinin kullanımına baktığımızda ise bu tür çerçevelerin hem geliştiricilerin üretkenliğini hem de ortaya çıkan yazılımın kalitesini, performansını ve güvenilirliğini arttırdığını görmekteyiz[9]. Konuşularak kontrol edilen yazılımların geliştirilmesi için de bu tür yazılım çerçevelerinin başarılı olduğu gösterilmiştir[14, 10]. Bunun yanında kullanıcının verdiği sesli komutları programın grafiksel arayüzüne otomatik veya yarı otomatik olarak dönüştürme üzerine spesifik konu alanlarında (örneğin, konuşma yolu ile yazılım geliştirme ortamlarının kontrol edilebilmesi) çalışmalar yapılmıştır [16, 15, 8]. Fakat Android uygulamalar için burada bahsettiğimiz türden bir yazılım çerçevesi henüz geliştirilmemiştir.

3 Önerilen Yazılım Çerçevesi

Önerdiğimiz yazılım çerçevesini 4 ana başlık altında anlatacağız. Öncelikle bir örnek senaryo üzerinden bu yazılım çerçevesi ile sesle kontrole açılmış bir yazılım

ile kullanıcının nasıl etkileşildiğini göstereceğiz. Daha sonra, yazılım çerçevesini kullanan bir Android uygulamasından hangi değişikliklerin beklendiğini anlatacağız ve olası anotasyonlara örnekler vereceğiz. Üçüncü olarak, yazılım çerçevesinin anotasyon kullanan bir uygulama ile nasıl etkileştiğini göstereceğiz. Son olarak yazılım çerçevesinin ana unsurlarından bahsedeceğiz.

3.1 Örnek Senaryo

Bu kısımda önerdiğimiz yazılım çerçevesi ile sesle kontrole açılmış bir yazılım ile kullanıcının **tamamen** ses yoluyla etkileşimini örnekleyeceğiz. Elimizde randevularımızı yönetebildiğimiz geleneksel bir takvim uygulaması olsun. Kullanıcı takvime yeni bir toplantı zamanı koymak istesin. Bu işlemi gerçekleştirmek için kullanıcı ile uygulama arasında şu diyalog geçer: Uygulamaya kullanıcı "yeni bir toplantı yarat" sesli komutunu verir. Uygulama kullanıcıya toplantıyı hangi gün ve saate koymak istediğini sesli olarak sorar. Kullanıcı "önümüzdeki Pazartesi saat 14'e" ifadesiyle sesli olarak cevap verir. Bunun üzerine uygulama bu toplantı için kullanıcının bir oda ayırtmak isteyip istemediğini sesli olarak sorar. Kullanıcı oda ayırtmak istediğini belirtir. Bunun üzerine uygulama kullanıcıya uygun olan olası bazı toplantı oda bilgilerini seslendirir. Kullanıcı okunan seçenekler içerisinden projektörü olmayan odaların elenmesini ister. Uygulama bu koşula göre ayıklanmış seçenekleri tekrar kullanıcıya okur. Kullanıcı seçimini sesli olarak uygulamaya iletir. Uygulama kullanıcıya toplantıya başka kişilerin davet edilip edilmeyeceğini sorar. Kullanıcı davet etmek istediğini belirtir. Uygulama kullanıcıya telefon rehberinden ulaşılabilen kontaklardan hangisini davet etmesi gerektiğini sorar. Burada seçeneklerin kısıtlı bir kısmı kullanıcıya okunur. Kullanıcı isterse sunulan seçenek kümesini "başka seçenekler sun" gibi komutlarla genişletebilir. Alternatif olarak kullanıcı tüm olası seçenekler arasında bir şarta göre ayıklama işlemi uygulatmak isteyebilir (örneğin, "ismi A harfiyle başlayanlar"). Kullanıcı bir kontak seçtikten ve bir başka davetlinin olmadığını uygulamaya sesli olarak bildirdikten sonra uygulama takvime toplantı bilgilerini kaydeder ve kullanıcıya işlemin başarıyla gerçekleştirildiğini sesli olarak bildirir.

3.2 Android Uygulamada Yapılması Gereken Değişiklikler

Bir uygulamayı, önerdiğimiz yazılım çerçevesi kullanarak sesle kontrole açmak için uygulamada aşağıdaki anlatılan değişikliklerin yapılması gerekmektedir.

Uygulama Kodunun Anotasyonlarla Etiketlenmesi Önerdiğimiz yazılım çerçevesinde desteklediğimiz anotasyonlardan bir tanesi `Voice` isimli bir anotasyondur. Bu anotasyon bir metodu etiketlemek için kullanılmaktadır ve 2 ana kısımdan oluşmaktadır. Birincisi kullanıcının metotta gerçekleştirilecek işlemleri istemek için söyleyebileceği sesli komutların tanımlandığı kısım. İkincisi işlemlerin gerçekleştirilebilmesi için gerekli olan parametrelerin tanımlandığı kısım.

Örnek Kod 1.1. Parametresiz metot için anotasyon

```
@Voice(commands = {"turn on captions"})
public void turnOnCaptions() { ... }
```

Örnek Kod 1.1'de bir video çalıcı uygulamada altyazıları göstermek için yazılmış bir metot ve bu metodun nasıl sesle kontrole açıldığını görmekteyiz. Burada `Voice` anotasyonu içerisinde `commands` alanında kullanıcının söyleyebileceği ifadeler belirtilmekte. Bu örnekte kullanıcı "turn on captions" komutunu verdiğinde bu metodun içinde detayları tanımlanan altyazıları açma ile ilgili işlemler gerçekleştirilmelidir.

Örnek Kod 1.2. Dinamik parametre değerli metot için anotasyon

```
@Voice(commands = {"open calendar item"},
       parameters = {
           @Parameter(
               name = "calendar item"
               description = "an event from the calendar",
               optionGetterMethodName =
                   "getCalendarItems")})
public void openCalendarItem(CalendarItem item) { ... }
```

Örnek Kod 1.2'de bir takvim uygulamasındaki bir toplantı nesnesinin detaylarına ulaşmayı sağlayan bir metot görmekteyiz. Burada metot, detayları öğrenilmek istenen toplantıya karşılık gelen nesneyi parametre olarak beklemekte. Burada `Voice` anotasyonunun `parameters` alanında metot parametreleri tanımlanabilmekte. Kullanıcı "open calendar item" komutunu verdiğinde yazılım çerçevesi bu metodun çalıştırılmak istediğini anlamakta ve metodun parametre beklediğini görmekte. Bunun üzerine yazılım çerçevesi anotasyonda belirtilen `getCalendarItems` metodunu çağırarak o anda uygulamanın içinde bulunduğu bağlamda hangi toplantı seçeneklerinin olduğunu uygulamaya sormakta. Daha sonra bu seçenekleri kullanıcıya sesli olarak sunup kullanıcının bir seçeneği seçmesini beklemekte. Bu seçenekleri sunarken anotasyonda `name` ve `description` alanlarında geçen metinler kullanıcıya sorulan soruyu oluşturmak için kulanılmakta. Örneğin bu örnekte kullanıcıya "please pick a calendar item" (Tr. lütfen bir takvim elemanı seçiniz) isteği sesli olarak sorulur. Bu mekanizmanın çalışması için `getCalendarItems` isminde bir metodun etiketlenen metot ile aynı kapsamda (İng. scope) uygulama geliştirici tarafından tanımlanmış olması gerekmektedir.

Örnek Kod 1.3. Komut içerisinden parametre değerlerini almak

```
@Voice(commands = {"make font size $1"},
       parameters = {
           @Parameter(
               name = "font size"
               optionGetterMethodName =
                   "getFontSizeOptions")})
public void setFontSize(FontSize fontSize) { ... }
```

Örnek Kod 1.3'te kullanıcıdan beklenen komut içerisinde parametre değerlerinin yerlerinin nasıl tanımlanabileceği belirtilmekte. Kullanici bu örnekte "make font size 15" (Tr. yazı büyüklüğünü 15 yap), komutunu verdiğinde en sondaki 15 rakamının metodun beklediği birinci parametre değeri olduğu yazılım çerçevesince varsayılır. Daha sonra 15 rakamı `getFontSizeOptions` metodundan dönen `FontSize` nesnelerinden birine dönüştürülür ve bu nesneyle `setFontSize` metodu çağrılır.

Örnek Kod 1.4. Kullanıcı komutta parametre değerlerini verebilir veya olası seçenekleri uygulamadan isteyebilir

```
@Voice(commands = {"add two numbers", "add $1 and $2"},
       parameters = {
            @Parameter(name = "first number"),
            @Parameter(name = "second number")})
public int add(int firstNumber, int secondNumber) { ... }
```

Örnek Kod 1.4'te parametre değerleri belirtilmemiş ve belirtilmiş iki komut seçeneğini aynı metot için görmekteyiz. Eğer kullanıcı ilk komutu ("add two numbers" - Tr. iki sayıyı topla) verirse o zaman eklenecek 2 sayı kullanıcıdan yazılım çerçevesi tarafından istenecektir. İkinci komut verildiğinde kullanıcı parametre değerlerini zaten vermiş olacağı için (örneğin, "add 5 and 10" - Tr. "5 ile 10'u topla") bu metot yazılım çerçevesi tarafından doğrudan çağırılabilecektir.

Örnek Kod 1.5. Seslendirilecek ifadeler kaynak kod isimlendirmelerinden çıkarılabilir

```
@Voice()
public void setPaintBrushColor(BrushColor color)

enum BrushColor { BLACK, RED, BLUE };
```

Örnek Kod 1.5'te metot anotasyonunda ihtiyaç duyulan bazı değerler kod geliştirici tarafından tanımlanmadığında dahi yazılım çerçevesinin neler yapabileceğini örnekler. Bu örnekte isimlendirmelerde hörgüç yazım stili (İng. camel case) kullanıldığı varsayılarak metot isminden "set paint brush color" ifadesinin kullanıcının verebileceği olası bir sesli komut olabileceği düşünülebilir. Yine, parametre tipinin isminden kullanıcıdan uygun bir parametre değeri isterken istenen değerin ne olduğu ile ilgili "brush color" açıklaması kullanılabilir. Olası parametre değerleri için ise enumerasyon isimlerinden "black", "red", "blue" seçenek isimleri üretilebilir ve bunlar kullanıcıya kullanıcının seçmesi için sesli olarak sunulabilir.

Örnek Kod 1.6. Çoklu parametre değerlerinin belirlenmesi

```
@Voice(parameters = {
            @Parameter(
                 name = "invitee"
                 suboptionGetterMethodName =
                      "getPhoneContacts")})
public void addInvitees(Contact[] contacts) {...}
```

Örnek Kod 1.6'da gösterilen metot takvimde bir toplantıya davetliler ekleyebilmekte. Kullanıcı "add invitees" (Tr. "davetli ekle") sesli komutunu verdiğinde yazılım çerçevesi olası kontak seçeneklerini `getPhoneContacts` metodunu çağırarak üretebilmekte. Burada diğer örneklerin aksine `optionGetterMethodName` yerine `suboptionGetterMethodName` seçeneği kullanılmıştır. Çünkü burada amaç kullanıcıya farklı kontak altkümelerini seçenek olarak sunmak değildir. Aksine beklenen dizinin her bir elemanı için kullanıcıya ayrı ayrı seçenek kümeleri sunmaktır. Kullanıcı sunulan kontaklardan birini seçtikten sonra, uygulama kullanıcıya daha başka davetli olup olmadığını sorar. Her kontak seçiminden sonra da bunu tekrar eder. Kullanıcı başka davetli olmadığını bildirdikten sonra `addInvitees` metodu kullanıcının seçtiği kontakları içeren bir dizi ile çağrılır.

Etiketli Nesnelerin Yazılım Çerçevesine Bildirimi Anotasyon uygulanmış sınıfların (örneğin, bir arayüzde gösterilen pencerenin) nesneleri uygulama çalışırken oluşturulduğunda kullanıcının vereceği sesli komutların gerçekleştirilebilmesi için yazılım çerçevesine bu nesnelerin bildirilmesi gerekir. Örnek Kod 1.7'de bu konu örneklenmiştir. Bu örnekte `BirAktivite` isimli sınıf uygulamada bir grafiksel arayüz penceresini temsil eder ve bu sınıfta kullanıcının sesle komuta etmesine açılmak istenmiş `action` isimli bir metot görülmektedir. Bu pencere uygulama tarafından ekranda gösterilmek istendiğinde bu sınıfın bir nesnesi yaratılır ve Android işletim sistemi tarafından `onActivityCreated` metodu çağrılır. Bu metot içerisinde de yazılım çerçevesine o anki nesne bildirilmektedir (Bakınız, `register` metodunun çağırılması).

Örnek Kod 1.7. Etiketlenmiş Sınıfların Nesnelerinin Yazılım Çerçevesine Bildirilmesi

```
public class BirAktivite extends SampleActivityBase {
  Voicer mVoicer;

  public void onActivityCreated(
      Bundle savedInstanceState) {
    mVoicer = Voicer.createVoicer(...);
    mVoicer.register(this);
  }

  public void onDestroy() {
    super.onDestroy();
    mVoicer.unregister();
  }

  @Voice(...)
  public void action(...) {...}}
```

Yazılım çerçevesi, kayıt edilen nesneyi kullanarak etiketlenen metotları bulur ve etiket içeriklerini kullanarak kullanıcının verebileceği komut detaylarını keşfeder. Daha sonra kullanıcının verdiği sesli komutlar yazılım çerçevesinin o an kayıtlı aktif nesnelerden birinin bir metoduna eşlenirse, yazılım çerçevesi o metodu yansıma (İng. reflection) mekanizması kullanarak kayıtlı nesne üzerinden

çağırır. Çağrılacak metodun gerektirdiği dinamik parametrelerin o anki alternatif değerlerini bulmak ve kullanıcıya sunmak için anotasyonlarda belirtilen parametrelerin getirici metotları da yine kayıtlı nesne üzerinden yansıma yoluyla çağrılır. Etiketlenmiş nesnenin ömrü bittiğinde ve uygulama tarafından serbest bırakılacağında (İng. deallocation), yazılım çerçevesine bu durum bildirilir (Bakınız, `unregister` metodunun çağırılması). Böylece bu nesne üzerinden yapılabilecek sesli komutlara da uygulama yeni geçeceği durumda cevap veremeyecektir.

Mikrofona Ulaşım İzninin Alınması Son olarak uygulamanın kullanıcıdan sesli komutları alabilmek için mikrofona ulaşmak istediğini uygulamanın "Manifest" dosyasında ilan etmesi gerekir:

3.3 Yazılım Çerçevesinin Anotasyon Kullanan Uygulama İle Etkileşimi

Komutların Dinlenmeye Başlanması Yazılım çerçevesinin kullanıcıdan sesli komutları alabilmek için cihazın mikrofonundan kullanıcıyı dinlemesi gerekmektedir. Bu dinlemeyi başlatmak için birkaç farklı mekanizma uygulanabilir. İlki uygulama arayüzüne gömülecek bir arayüz elemanı (menü seçeneği, mikforon resmi olan bir düğme vb.) ve bu eleman ile etkileşiminden kullanıcının sesli komut vermeye başlayacağının anlaşılması. İkinci yöntem cihaz ekranında yapılacak özel bir jest hareketinin algılanması veya cihaz ivme ölçerli bir mobil cihaz ise cihazın özel bir şekilde sallandığının tespiti vb. cihaz üzerinde uygulanan fiziksel eylemlerin kullanıcıyı dinlemeye başlamak için kullanılması. Son yöntem ise ortamdan gelen sesleri yazılım çerçevesinin sürekli dinlemesi ve belirli bir anahtar ifade kullanıldığında (İng. hotword) kullanıcının sesli komut vermeye başlayacağının varsayılması. Bu yöntem Android ve iOS cihazlarda "Ok Google" ve "Hey Siri" ifadeleri ile işletim sistemi seviyesinde kullanılmaktadır. Bu yöntem yazılım çerçevesi için gerçekleştirilmesi en zor yöntemdir. Çünkü bu yöntem işletim sistemi ve donanım desteği olmadan uygulama seviyesinde yapılmaya çalışıldığında yüksek enerji kullanımına ve cihazı yavaşlatacak işlem yoğunluğuna yol açacaktır.

Kullanıcıya Geribildirimlerin Yapılması Yazılım çerçevesinin komutları gerçekleştirme sırasındaki işlemlerin sonuçlarını kullanıcıya görsel ve sesli geribildirimler ile ulaştırması gerekir. Örneğin parametre seçeneklerini kullanıcıya sunarken bu seçenekleri hem seslendirmelidir hem de ekranda göstermelidir. Yazılım çerçevesinden gelen geri bildirimlerin uygulamanın kendi kullanıcı arayüzü ile işbirliği içinde çalışması da gerekmektedir. Bu sebeple bildirimlerin yapılmasına yönelik arayüz elemanlarının yazılım çerçevesine uygulama tarafından sağlanması beklenmektedir.

3.4 Yazılım Çerçevesinin Ana Bileşenleri

Önerdiğimiz yazılım çerçevesi şu anda şu ana bileşenlerden oluşmaktadır:

– `CommandExtractor`: Bu sınıf yazılım çerçevesine bildirilen nesnelerin anotasyonlarını inceler ve bu anotasyonlara göre uygulamanın desteklediği sesli komutları belirler.
– `CommandRegistry`: Bu sınıf uygulamanın desteklediği komutları tutar ve verilen bir sesli sorguyu tutulan komutlardan birine eşlemeyi gerçekleştirir.
– `CommandInterpreter`: Cihazdaki, konuşmadan metine çevirme altyapısını kullanarak kullanıcının verdiği sesli sorguyu bir grup aday metne çevirir. Bu adaylarda gürültü giderici ve genelleyici bazı metinsel işlemler gerçekleştirdikten ve sorgulardaki parametre değerlerini tespit ettikten sonra, bu adayları uygulamanın desteklediği komutlara eşler ve bu komutlardan en muhtemelini seçer.
– `CommandExecutor` Kullanıcının hangi komutu verdiği belirlendikten sonra komutun gerçekleştirilmesi için gerekli eksik parametre değerlerini tespit eder. Her eksik parametre değeri için olası değerler kümesini belirler ve bunları kullanıcıya sunar. Tüm parametre değerleri belirlendikten sonra komuta karşılık gelen metodun çağrılmasını sağlar ve metodun sonucunu kullanıcıya aktarır.
– `InteractionEngine` Kullanıcıyla olan görsel ve işitsel etkileşimi yönetir. Kullanıcının konuşmalarının metine çevrilmesi için cihazda bulunan konuşma tanıma (İng. speech recognizer) altyapısını kullanır. Kullanıcıya sesli geri bildirimler yapmak için cihazda bulunan metinden konuşmaya (İng. text-to-speech) çevirme motorunu kullanır. Görsel bildirimlerin yapılması için uygulama tarafından sağlanan grafiksel arayüz elemanları ile etkileşir.
– `Voicer`: Bu sınıf yukarıdaki sınıfların nesnelerini yönetir ve uygulamaya bu fonksiyonları cephe örüntüsü şeklinde sunar.

4 Sonuçlar ve Gelecek Çalışmalar

Bu çalışmada Android uygulamalarını konuşularak da etkileşilebilir hale getirilebilmesi için uygulama geliştiricilerin kullanabileceği anotasyon yönelimli bir yazılım çerçevesi önerdik. Sesle etkileşimli bir kullanıcı arayüzü geliştirmek için uygulama geliştiricide deneyim gerektiren konuşma tanıma, parametre değerlerinde belirsizliği giderme, doğal dil işleme, makine öğrenmesi gibi bütün benzer uygulamalarda ihtiyaç duyulacak konuların gerçekleştirilmesini yazılım çerçevesinin sorumluluğuna vererek uygulama geliştirme ve bakım maliyetlerini azaltmayı hedefledik. Bu maliyetlerin azalmasıyla daha fazla uygulamanın sesle etkileşilebilen arayüzlere de sahip olacağına inanıyoruz. Bunun da orta ve uzun vadede geleneksel dokunma merkezli etkileşilen uygulamaları kullanamayan bedensel ve görme engelli kullanıcıların kullanabilecekleri uygulama sayısının artmasına yol açacağını düşünüyoruz. Önümüzdeki dönemde yazılım çerçevesinin doğal dil işleme, makine öğrenmesi ve komut algılama mekanizmasının uluslararasılaştırılması (İng. internationalization) gibi konularda zenginleştirilmesi üzerinde çalışmalarımızı yürütmeyi planlamaktayız.

Kaynakça

1. Cortana by Microsoft, https://www.microsoft.com/en/mobile/experiences/cortana/, accessed: 2016-02-23
2. Declaring an Annotation Type, https://docs.oracle.com/javase/tutorial/java/annotations/declaring.html, accessed: 2016-10-06
3. Google Now: Ok Google voice search and actions, https://support.google.com/websearch/answer/2940021?hl=en{\&}rd=1, accessed: 2016-02-23
4. Improved Code Inspection With Annotations (Android), https://developer.android.com/studio/write/annotations.html, accessed: 2016-06-10
5. Java annotation, https://en.wikipedia.org/wiki/Java{_}annotation, accessed: 2016-06-10
6. Siri for iOS, http://www.apple.com/accessibility/ios, accessed: 2016-02-23
7. Voice Access Beta: Building more accessible technology, https://googleblog.blogspot.com.tr/2016/04/building-more-accessible-technology.html, accessed: 2016-04-15
8. Arnold, S.C., Mark, L., Goldthwaite, J.: Programming by voice, VocalProgramming. Proceedings of the fourth international ACM conference on Assistive technologies - Assets '00 pp. 149–155 (2000), http://dl.acm.org/citation.cfm?id=354324.354362
9. Fayad, M., Schmidt, D.C.: Object-oriented application frameworks. Communications of the ACM 40(10), 32–38 (10 1997), http://portal.acm.org/citation.cfm?doid=262793.262798
10. Glass, J., Weinstein, E., Cyphers, S., Polifroni, J., Chung, G., Nakano, M.: A Framework for Developing Conversational User Interfaces. In: Computer-Aided Design of User Interfaces IV, pp. 349–360. Springer-Verlag, Berlin/Heidelberg (2005), http://link.springer.com/10.1007/1-4020-3304-4{_}28
11. Lei, X., Senior, A., Gruenstein, A., Sorensen, J.: Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices (August), 662–665 (2013)
12. McGraw, I., Prabhavalkar, R., Alvarez, R., Arenas, M.G., Rao, K., Rybach, D., Alsharif, O., Sak, H., Gruenstein, A., Beaufays, F., Parada, C.: Personalized Speech Recognition On Mobile Devices. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE (2016)
13. Raman, T., Charles, L.C., Tim, C.: Leveraging Android accessibility APIs to create an accessible experience. In: Google I/O Conference (2011)
14. Srinivasan, S., Vergo, J.: Object oriented reuse: experience in developing a framework for speech recognition applications. Proceedings of the 20th International Conference on Software Engineering pp. 322–330 (1998), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=671385
15. Wagner, A.: Automation of VUIto GUI mapping. In: CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13. p. 1941. ACM Press, New York, New York, USA (2013), http://dl.acm.org/citation.cfm?doid=2468356.2468706
16. Wagner, A.: Programming By Voice: A Hands-Free Approach For Motorically Challenged Children. Ph.D. thesis (2015)
17. Zhong, Y., Raman, T.V., Burkhardt, C., Biadsy, F., Bigham, J.P.: JustSpeak. In: Proceedings of the 11th Web for All Conference on - W4A '14. pp. 1–4. ACM Press, New York, New York, USA (2014), http://dl.acm.org/citation.cfm?doid=2596695.2596720