

VILDIZ TEKNİK ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

Wap Pro. için Wml/Html Geç.
Sağ. Sist. Tas. ve Uyg.

YÜKSEK LİSANS TEZİ

M. Utku Kalay

2003

2 368 101

Katiphan

**YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

**WAP PROTOKOLU İÇİN WML/HTML GEÇİŞİNİ
SAĞLAYAN SİSTEMİN TASARIMI VE UYGULANMASI**

Elektronik Müh. Mustafa Utku KALAY

FBE Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. A. Gökhan Yavuz

İSTANBUL, 2003

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE VE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

Yer No (DDC) : R/368/1010

Kayıt No : 2158

Geldiği Yer : Fen Bilimleri Enk

Tarih : 19/10/2004

Fiyat : 3.650.000

Fatura No :

Ayniyat No : 1/13

Ek :

YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

1X-73

WAP PROTOKOLU İÇİN WML/HTML GEÇİŞİNİ
SAĞLAYAN SİSTEMİN TASARIMI VE UYGULANMASI

Elektronik Müh. Mustafa Utku KALAY

FBE Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında
Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Yrd. Doç. Dr. A. Gökhan Yavuz

Prof. Dr. Oya Kalipsiz
Doç. Dr. Selim AKYOKUŞ

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ.....	iv
ŞEKİL LİSTESİ.....	v
ÇİZELGE LİSTESİ.....	vi
ÖNSÖZ.....	vii
ÖZET.....	viii
ABSTRACT.....	ix
1. GİRİŞ.....	1
2. MOTİVASYON.....	3
2.1 Off-line XHTML/WML Dönüşümünü Sağlayan Program.....	4
2.2 Arayüz.....	4
2.3 XSLT.....	4
3. İNTERNET VE KABLOSUZ İLETİŞİM.....	8
3.1 İnternet ve WWW.....	8
3.2 HTML.....	11
3.3 Kablosuz İletişim.....	12
4. KABLOSUZ ORTAMDA UYGULAMA GELİŞTİRME PROTOKOLU.....	14
4.1 WAP Modeli.....	15
4.2 WAP Protokol Yığını.....	17
4.2.1 WAE (Wireless Application Environment).....	17
4.2.2 WSP (Wireless Session Protocol).....	18
4.2.3 WTP (Wireless Transaction Protocol).....	18
4.2.4 WDP (Wireless Datagram Protocol).....	19
4.2.5 WTLS (Wireless Transport Layer Security).....	19
4.2.6 Taşıyıcı (Bearer).....	19
4.2.7 Diğer Servisler ve Uygulamalar.....	20
4.2.8 Örnek Protokol Yığınları.....	20
4.3 Kablosuz Ortamda Biçimlendirme Dili (Wireless Markup Language-WML).....	21
4.3.1 SGML, HTML, XML.....	21
4.3.2 WML Yapısı ve Elemanları.....	23
4.3.3 WMLScript.....	27
5. HTML/WML GEÇİŞİNİ SAĞLAYAN SİSTEMİN TASARIMI.....	28
5.1 HTML ile WML Dili Arasındaki Farklar.....	29
5.1.1 Karakter Kodları.....	31

5.2	Sistemin Tasarımı ve Temel Yapı Birimleri	33
5.2.1	WAP Kullanıcısı.....	34
5.2.2	HTML/XHTML Dönüşümü (TIDY ve Java TIDY)	35
5.2.2.1	Java TIDY	36
5.2.3	Ayrıştırma.....	36
5.2.4	“XHTMLtoWML” Dönüşümü.....	41
5.2.4.1	Örnek bir Uygulama	43
5.2.5	Dönüşüm Akış Şeması ve Kuralları	46
5.2.6	Uygulamada Kullanılan Temel Algoritmalar	51
5.2.7	Sistemin Test Edilmesi ve Sistemin Sorunları	55
6.	SONUÇ	58
KAYNAKLAR.....		59
INTERNET KAYNAKLARI:.....		59
EKLER		60
Ek 1 Sistemin genel akış diyagramı		61
Ek 2 XHTML/WML arayüzünün genel akış diyagramı		62
Ek 3 traverse(org.w3c.dom.Node node) rekürsif fonksiyonunun akış diyagramı		63
Ek 4 Etiket dönüşüm modu akış diyagramı		64
Ek 5 İkinci moda giriş akış diyagramı		70
Ek 6 İkinci modda çalışma akış diyagramı		71
Ek 7 Metin yazma modu akış diyagramı.....		72
ÖZGEÇMİŞ		73

KISALTMA LİSTESİ

API	Application Programming Interface	5
DOM	Document Object Model	5
DTD	Document Type Definition	6
HTML	Hypertext Markup Language	9
IPv4	Internet Protocol version 4	10
SAX	Simple API for XML	11
SGML	Structured Generalized Markup Language	13
SMS	Short Messaging Service	16
TCP/IP	Transmission Control Protocol/Internet Protocol	17
URL	Uniform Resource Locator	23
UTF	Unicode Transformation Formats	25
XHTML	eXtensible Hypertext Markup Language	29
Xpath	XML Path Language	26
XSLT	eXtensible Stylesheet Language Transformations	28
WAP	Wireless Application Protocol	33
WML	Wireless Markup Language	37
WWW	World Wide Web	40
Şekil 5.5 Java Tidy konfigürasyonları		42
Şekil 5.6 XHTML ağacının WML ağacına dönüşümünü sağlayan komutlar		43
Şekil 5.7 Örnek bir HTML dokümanı		43
Şekil 5.8 TIDY işlemi sonunda oluşan XHTML dokümanı		44
Şekil 5.9 Ayrıştırma sonunda oluşan XHTML ağacı		45
Şekil 5.10 Dönüştürme sonunda oluşan WML ağacı ve dokümanı		45
Şekil 5.11 İki tane tablonun çözümlenmiş örneği(skip mod)		47

ŞEKİL LİSTESİ

Şekil 2.1 XML doküman örnekleri	5
Şekil 2.2 XML doküman örnekleri	5
Şekil 2.3 XHTML WML dönüşümünü yapan XSLT doküman örneği	6
Şekil 3.1 İnternetteki uç birimlerin senelere göre artışı	9
Şekil 3.2 WWW'ın yapısal birimleri.....	10
Şekil 3.3 Basit bir HTML dokümanı.....	11
Şekil 3.4 Cep telefonu İnternet erişimi.....	13
Şekil 4.1 Cep telefonu, arayüz, servis sağlayıcı ilişkisi	16
Şekil 4.2 WAP protokol yığını.....	17
Şekil 4.3 Örnek protokol yığınları.....	20
Şekil 4.4 <p> elemanının yapısı.....	25
Şekil 4.5 Basit bir WML program yapısı	25
Şekil 4.6 Örnek bir WML programı.....	26
Şekil 5.1 HTML dosyasından birden çok WML dosyası oluşturmak.....	28
Şekil 5.2 HTMLtoWML sisteminin temel yapı birimleri	33
Şekil 5.3 Örnek bir ayrıştırma ağacı.....	37
Şekil 5.4 “Tokenizer” nesnesinin ayrıştırma ağacını oluşturması.....	40
Şekil 5.5 Java Tidy konfigürasyonları.....	42
Şekil 5.6 XHTML ağacının WML ağacına dönüşümünü sağlayan komutlar.....	43
Şekil 5.7 Örnek bir HTML dokümanı	43
Şekil 5.8 TIDY işlemi sonunda oluşan XHTML dokümanı	44
Şekil 5.9 Ayrıştırma sonunda oluşan XHTML ağacı	45
Şekil 5.10 Dönüşüm sonunda oluşan WML ağacı ve dokümanı	46
Şekil 5.11 İç içe tabloların çözülmesi örneği(skip mod).....	47

ÇİZELGE LİSTESİ

Çizelge 5.1 Bazı HTML ve WML etiketleri ve özellikleri	30
Çizelge 5.2 Bazı karakter setleri ve ilgili kodlama standardı.....	31
Çizelge 5.3 HTML Tidy örnekleri	35
Çizelge 5.4 Arayüzde uygulanan dönüşüm kuralları	48

ÖNSÖZ

Bu tezin hazırlanmasında bilgi ve tecrübelerini benimle paylaşan danışman hocam Yrd. Doç. Dr. A.Gökhan Yavuz'a teşekkür ederim.

Bu her yenden kolay ve hızlı erişimde kuşkuyla en önemli olan Internet ve teknolojilerinin tasarımı ve uygulanmasıdır.

Günümüzde Internet ve ilgili teknolojiler ilk ortaya atıldıkları zamandaki tahminlerin çok ötesinde boyutlara ulaşmıştır. Bilgi çeşitliliği, sunulan servislerin çeşitliliği, kullanıcı sayısı ve uç birimlerin çeşitliliği gibi artan bir çok parametre Internet'in trafik karakteristiklerini doğrudan etkilemektedir.

Bütün bunların yanında kablosuz haberleşme teknolojilerinin son yıllarda sunduğu servis çeşitliliği tahminlerin üzerinde artış göstermektedir. Kullanılan cep telefonu veya kablosuz uç birimlerin yayılma ve çeşitliliği, sunulan servis çeşitlilikleri bu iletişim ortamındaki haberleşme teknolojilerinin günümüzde geldiği noktayı göstermekte ve gelecekteki hedeflere ışık tutmaktadır.

Yukarıdaki paragraflarda bahsedilen iki farklı ortamdaki haberleşme teknolojilerinin birleşmesi her çeşit bilgi ve servisin hem kablolu hem kablosuz kullanıcılar tarafından erişimi ve paylaşımı yaşadığımız bilgi çağını en önemli hedeflerinden biridir. Bu amaç doğrultusunda kablosuz ortamdaki hareketli kullanıcılar (cep telefonu, laptop vb) Internet ortamındaki veriyi erişmeyi ve kablolu ortamdaki servisleri en uygun şekilde kullanmak isteyeceklerdir. İşte bu iki iletişim ortamındaki verinin ortak kullanılmasına hedefleyen teknolojilerden birisi WAP (*Wireless Application Protocol*)'dir.

Bu tezde WAP protokolü kapsamında bir uygulama gerçekleştirilmiştir. Tasarlanan sistem kablosuz kullanıcıların gerçek zamanlı olarak Internet'e erişmesine olanak verme amacıyla HTML biçimlendirme dilini WML biçimlendirme diline dönüştürmektedir. Bu dönüşümün tasarımı, karşılaşılan sorunlar ve yapıları benzer çalışmalar ile karşılaştırılması incelenmiştir.

Araştırma konuları WAP, WML, Internet, HTML, Biçimlendirme dilinin dönüştürülmesi

ÖZET

Bilgiye hızlı erişim ve bilginin paylaşım etkinliği günümüzdeki bir çok teknolojinin temel yapı taşı oluşturmaktadır. Bilgiye her an her yerden kolay ve hızlı erişimde kuşkusuz en önemli etken Internet ve teknolojilerinin tasarımı ve uygulanmasıdır.

Günümüzde Internet ve ilgili teknolojiler ilk ortaya atıldıkları zamandaki tahminlerin çok ötesinde boyutlara ulaşmıştır. Bilgi çeşitliliği, sunulan servislerin çeşitliliği, kullanıcı sayısı ve uç birimlerin çeşitliliği gibi artan bir çok parametre Internet'in trafik karakteristiklerini doğrudan etkilemektedir.

Bütün bunların yanında kablosuz haberleşme teknolojilerinin son yıllarda sunduğu servis çeşitliliği tahminlerin üzerinde artış göstermektedir. Kullanılan cep telefonu veya kablosuz uç birimlerin sayısı ve çeşitliliği, sunulan servis çeşitlilikleri bu iletim ortamındaki haberleşme teknolojilerinin günümüzde geldiği noktayı göstermekte ve gelecekteki hedeflere ışık tutmaktadır.

Yukarıdaki paragraflarda bahsedilen iki farklı ortamdaki haberleşme teknolojilerinin bütünleşmesi her çeşit bilgi ve servisin hem kablolu hem kablosuz kullanıcılar tarafından erişimi ve paylaşımı yaşadığımız bilgi çağının en önemli hedeflerindendir. Bu amaç doğrultusunda kablosuz ortamdaki hareketli kullanıcılar (cep telefonu, laptop vb) Internet ortamındaki veriye erişmeyi ve kablolu ortamdaki servisleri en uygun şekilde kullanmak isteyeceklerdir. İşte bu iki iletim ortamındaki verinin ortak kullanılmasını hedefleyen teknolojilerden birisi WAP (*Wireless Application Protocol*)dır.

Bu tezde WAP protokolü kapsamında bir uygulama gerçekleştirilmiştir. Tasarlanan sistem kablosuz kullanıcıların gerçek zamanlı olarak Internet'e erişmesine olanak verme amacıyla HTML biçimlendirme dilini WML biçimlendirme diline dönüştürmektedir. Bu dönüşümün tasarımı, karşılaşılan sorunlar ve yapılan benzer çalışmalar ile karşılaştırılması incelenmiştir.

Anahtar kelimeler: WAP, WML, Internet, HTML, Biçimlendirme dillerinin dönüşümü

ABSTRACT

Accessing information in a timely, reliable and recognizable fashion is the most important problem in today's information technologies. Since Internet is the largest information network in the world, all kinds of users around the world (wired or wireless) want to access the Internet.

The Internet is a global digital infrastructure that connects millions of computers. The variety of services currently provided by the Internet is rather behind the estimations at the time of first appearance of Internet technologies. The variety of information format, services, and end user platforms and the ever-increasing number of end users directly effect the structure and effectiveness of the Internet.

On the other hand the wireless communication technologies also provides many different services and solutions. The current trends on the variety of services provided by wireless technologies and their popularity in today's market sheds light on the future's goal in the wireless communication networks.

The communication medias that was introduced in the previous paragraphs can be combined into a larger communication media. At least the current goal is to enable the wireless users to access almost all services over the Internet. WAP, namely Wireless Application Protocol, is one of the leading standards that seeks new solutions for this kind of information access problem. For instance, a WAP gateway is expected to enable wireless end users to access the Internet and receive services such as viewing html sites, e-mail and banking application.

In this thesis, a WAP gateway is designed and implemented. The gateway receives HTML requests from the wireless end users, accesses to WWW sites and then converts the HTML page to the WML format, then eventually sends the information back to the wireless end user in the available format. All the stages during this process, specifically design, implementation, testing and maintenance are discussed in this thesis.

Keywords: WAP, WML, Internet, HTML, Conversion between markup languages

1. GİRİŞ

Bu tezde WAP (*Wireless Application Protocol*) protokolü kapsamında, kablosuz kullanıcı uç birimlerinin HTML (*Hypertext Markup Language*) sayfalarını görüntülemesine olanak veren sistemin tasarımı ve uygulanması gerçekleştirilmiştir.

WAP protokolü, kablosuz haberleşme teknolojileri kullanarak geliştirilen uygulamalarda kablosuz ağın hem kendi içinde hem de kablosuz ağın diğer iletişim ortamları ile haberleşmesinde belirli standartlar tanımlamaktadır. Bu standartlar ile farklı ortamlardaki veri kablosuz uç birime gerçek zamanlı olarak sunulur. Bu noktada ilk akla gelen, Internetin kablosuz ortama sunulabilmesidir. Kablosuz ortamdaki hareketli kullanıcının Internet gibi dünyayı kaplayan bir ağa erişimini hedefleyen bu yönde bir yaklaşım WAP ve benzer protokollerin son yıllarda önem kazanmasına sebep olmuştur.

Kablolu ve kablosuz uç birimlerin kapasitesine ve diğer özelliklerine baktığımız zaman bu birimlerde kullanılan verinin miktarı ve tipinin birbirinden farklı olduğu görülür. Bu yüzden WAP ve benzer protokoller kablolu ortamdaki veriyi kablosuz uç birimin özelliklerine uygun olarak gerekli filtrelerden geçirir. Bununla beraber iki ortamın veriyi biçimlendirme dili de farklıdır. Kablolu ortamda HTML dili kullanılırken, kablosuz ortamda WML (*Wireless Markup Language*) biçimlendirme dili kullanılır.

İki farklı ortamda kullanılan protokollerin birbirine dönüştürülmesi genel anlamda bir Arayüz (*gateway, proxy*) üzerinden olur. Bu tezde bu protokollerin birbirine dönüşümünü sağlayan basit bir arayüz gerçekleştirilmiştir. Bu arayüzde alınan bir HTML sayfasının WML sayfalarına dönüşümü gerçekleştirilmektedir. Yapılan dönüşüm sadece sunum katmanında gerçekleştirilen bir dönüşüm olduğu için gerçek boyutta bir protokol dönüşümüne göre oldukça basit bir sistemdir. Bunun yanında iki biçimlendirme dili arasında birebir dönüşüm imkanı olmadığı gibi, biçimlendirme dilleri arasında böyle bir dönüşüm karmaşıklığı ve hata olasılığı yüksek bir işlemdir. Bu temel sorunlar sistemin hızını ve dönüşümde veri kaybını doğrudan etkileyen parametrelerdir. Bu tezin amacı en az karmaşıklık ve en az veri kaybı ile böyle bir dönüşüm sistemi tasarlamaktır.

Bu bilgiler ışığında geliştirilen simülasyon ortamında sistemin 3 temel yapı birimi vardır: HTML Ayırıştırıcı, HTML/WML Dönüştürücü ve WML Tarayıcı. Tasarlanan dönüşüm sistemi gerçek zamanlı çalışmakta olup JAVA programlama dili kullanılarak gerçekleştirilmiştir. Bunun nedeni sistemin Ayırıştırıcı (*Parser*) ve WML Tarayıcı (*WML Browser*) gibi diğer mevcut parçalarının JAVA dili kullanarak gerçekleştirilmiş olmasıdır.

Böylece dönüşüm programının sistemin diğer mevcut parçaları ile entegrasyonu basit ve hızlı bir şekilde sağlanmıştır. WML tarayıcı, kullanıcı ile etkileşimin sağlandığı grafiksel bir arayüzdür. Bu arayüz üzerinden alınan kullanıcı isteği ağa aktarılır ve alınan HTML sayfası WML formatına dönüştürülerek WML tarayıcısına iletilir. Gerçeklenen sistemin başarı oranı kullanıcının erişmek istediği sayfa türüne ve sayfanın standart HTML kurallarına uygunluğuna doğrudan bağlıdır.

Bundan sonraki bölümde bu tezde incelenen sorunlara getirilen farklı çözümler incelenecektir. Daha sonraki bölümlerde Internet ve kablosuz iletişim ortamında kullanılan protokoller, WAP protokol yapısı ve WML dili örneklerle anlatılacaktır. HTML dokümanlarını WML dokümanlarına dönüştüren sistemin tasarımı, akış diyagramları ve sistemin sorunları bu tezin diğer temel konu başlıklarıdır.

Kablosuz iletişimde telefon haberleşmesi yanında son yıllarda ortaya çıkan Internet ile birleşene teknoloji yeni teknolojileri beraberinde getirmektedir. Temel hedef veriye hızla ulaşarak her yerden erişmeye olanak vermektedir. Kablosuz ağ sistemlerin Internet'e erişmelerine olanak veren en tip protokollere örnek olarak NTT DoCoMo'dan I-mode, WAP Forum'dan WAP (Wireless Access Protocol), Sun Microsystems'dan J2ME, Free Protocols Foundation'dan LEAP (Light Efficient Access Protocol), IDO ve TTA'dan B2WAP, and J-Phone'dan J-Sky Web örnek olarak verilebilir [4].

I-mode, kablosuz mobil cihazlara veri transferi ve Internet erişimi sağlayan bir standarttır. Japonya NTT DoCoMo şirketinin geliştirdiği dünyaya sunduğu bu standart ilk olarak 1999 yılında kullanıma başlandı. WAP'a benzerlikleri olmasına karşın, kablosuz iletişim için daha çok gelişmiş belirli protokolleri kullanan I-mode, WAP'ın kullandığı WML yerine CWML (Compact Wireless Markup Language) kullanmaktadır. I-mode veri hizmetleri ile cep telefonlarına ses dışında renkli görüntü ve video yayını da yapılabilir. Kullanımı geçtikten sonra hızla kullanıcı sayısı artan I-mode, Japonya'da kullanımı geçtiği ilk yılın sonunda 17 milyon kullanıcıya ulaşmayı başardı [5].

Bundan sonraki bölümde HTML biçimlendirme dilinin WML biçimlendirme diline dönüşümü ile ilgili olarak ortaya çıkan bazı yaklaşımlar ele alınacaktır. Daha sonraki bölümlerde bu tezde incelenen sistemin bu bölümlerdeki sistemlere benzerlikleri yer geldiklerinde daha detaylı incelenecektir.

2. MOTİVASYON

Bu bölümde WAP protokolünün ülkemizdeki uygulamaları ve WAP protokolünün hedefleri doğrultusunda benzer diğer bazı protokollerden bahsedilecektir. Son olarak bu tezin temel konusu olan HTML/WML dönüşümü yapan mevcut sistemler ve yaklaşımlar incelenecektir.

Günümüzde WAP teknolojisi bir çok alanda kullanılmaktadır. Bankacılık işlemleri gibi yüksek güvenlik gerektiren uygulamalardan, rezervasyon işlemleri gibi gerçek zamanlı işlemlere kadar geniş bir kullanım yelpazesi oluşturulmaktadır. Örneğin Kısa Mesaj Servisi'ne (*SMS, short messaging service*) gösterilen ilgiye bakıldığında, yazılı iletişimin artık sesli iletişim kadar tercih edildiği anlaşıyor. Bunun yanısıra Türkiye'deki bazı servis sağlayıcılar Interneteki bilgi servislerini doğrudan kullanabilme, arama yapabilme, elektronik mektup alıp gönderebilme gibi kablolu ortamda yapılan pek çok uygulamayı artık mobil telefonundan erişebilmeye olanak veren servisler sunmaktadır.

Kablosuz iletişimde telefon haberleşmesi yanısıra son yıllarda ortaya atılan Internet ile bütünleşme adımları yeni teknolojileri beraberinde getirmektedir. Temel hedef veriye her zaman her yerden erişmeye olanak vermektir. Kablosuz uç birimlerin Internet'e erişmelerine olanak veren bu tip protokollere örnek olarak NTT DoCoMo'dan I-mode, WAP Forum'dan WAP (*Wireless Access Protocol*), Sun Microsystems'dan J2ME, Free Protocols Foundation'dan LEAP (*Light Efficient Access Protocol*), IDO ve Tu-Ka Speed'den EZWeb, and J-Phone'den J-SkyWeb örnek olarak verilebilir [4].

I-mode, kablosuz mobil cihazlara veri transferi ve Internet erişimi sağlayan bir standarttır. Japon NTT DoCoMo şirketinin geliştirerek dünyaya sunduğu bu standart ilk olarak 1999 yılı içinde kullanılmaya başlandı. WAP'a benzerlikleri olmasına karşın, kablosuz iletişim için daha çok geleneksel iletişim protokollerini kullanan i-mode, WAP'ın kullandığı WML yerine CWML'i (*Compact Wireless Markup Language*) kullanmaktadır. I-mode veri hizmetleri ile cep telefonlarına ses dışında renkli görüntü ve video yayını da ulaştırılabilmektedir. Kullanıma geçtikten sonra hızla kullanıcı sayısı artan i-mode, Japonya'da kullanıma geçtiği ilk yılın sonunda 17 milyon kullanıcıya ulaşmayı başardı. [5]

Bundan sonraki bölümde HTML biçimlendirme dilinin WML biçimlendirme diline dönüşümü ile ilgili olarak ortaya atılan bazı yaklaşımlar ele alınacaktır. Daha sonraki bölümlerde bu tezde tasarlanan sistemin bu bölümdeki sistemlere benzerlikleri yeri geldiğinde daha detaylı incelenecektir.

2.1 Off-line XHTML/WML Dönüşümünü Sağlayan Program

Bu yaklaşımda program XML (*eXtensible Markup Language*) dokümanını okur, kelimeleri ayrıştırır (*parsing*), ve oluşan ayrıştırma ağacında gerekli dönüşümleri yapar. Daha sonra oluşan yeni ağaç WML dosyasına aktarılır.

Kansas Eyalet Üniversitesi Bilgisayar Bilimleri Bölümünde yüksek lisans düzeyinde yapılan bir çalışmada XML dokümanlarından oluşturulan ayrıştırma ağacının WML ağacına dönüştürülmesi bir JAVA programı ile gerçekleştirilmiştir[10]. Bu çalışmada giriş her zaman XML dokümanıdır ve gerçek zamanlı bir sistem değildir.

2.2 Arayüz

Birinci yönteme ek olarak dönüştürme işlemi gerçek zamanlı olarak bir arayüz üzerinde yapılabilir. Bu tezde geliştirilen sistem bu yöntem ile geliştirilen sistemlerden yola çıkılarak tasarlanmıştır.

Örneğin Kaasinen vd. (2000) tarafından tasarlanan bir sistemde, HTML/WML dönüşümünü sağlayan arayüzde, bu işlevinin yanısıra önbelleğe alma (*caching*), içerik dönüşümleri (görüntü dosyalarının farklı formatlara dönüşümü) gibi yardımcı uygulamalar vardır. Sisteme kablosuz uç birimden gelen istek, arayüzden kaynağın olduğu servis sağlayıcıya iletilir. Daha sonra arayüze gelen HTML sayfası sözdizimi hataları temizlendikten sonra WML'e dönüştürülür ve kullanıcıya WML dokümanı gönderilir. Bu tarz bir sistemin temelini dönüşüm kuralları oluşturur.

2.3 XSLT

XML Şema Tanımlama Dili XML dokümanlarının yapısını ve tipini tanımlamak için kullanılır [11]. Fakat aynı tanım kümesi üzerinde farklı şema dokümanları olması kaçınılmazdır. Bu da XML dokümanları arasındaki dönüşümün hızını etkin ölçüde azaltacaktır.

XSLT (*eXtensible Stylesheet Language Transformations*) XML dokümanları arasındaki dönüşüm kurallarını belirleyen yine XML tabanlı bir dokümandır. Dönüşüm kuralları XSLT dokümanında belirli bir formatta sunulur. XSLT bir programlama dili olarak düşünülebilir. Örneğin XHTML (*eXtensible Hypertext Markup Language*) formatında bir bilginin başka bir XML dokümanına (WML gibi) dönüşümü XSLT'de yazılan transformasyon kuralları ile gerçekleştirilebilir [11].

XSLT dokümanı birbirinden bağımsız sırayla ifade edilen kurallar zinciridir. Şekil 2.1’de iki farklı XML dokümanı örneği verilmiştir.

<pre> <?xml version “1.0” ?> <head call=“XSLT Example”> <writer name= “Foo Bar” /> <writer name= “John Doe” /> <writer name= “Any Othr” /> </head> </pre>	<pre> <?xml version “1.0” ?> <name>XSLT Example</name> <contributors> <staff> Foo Bar </staff> <staff> John Doe </staff> <staff> Any Othr </staff> </contributors> </pre>
(a)	(b)

Şekil 2.1 XML doküman örnekleri

Yukarıdaki iki XML dokümanı aynı bilgileri içermesine rağmen şema dokümanları birbirinden tamamen farklıdır. Daha önemlisi bu gibi basit iki doküman arasında dahi bir format dönüşümü yapılması algoritmik olarak zordur. Fakat bu dönüşüm XSLT ile kolayca ve hızlı bir şekilde gerçekleşebilir [8]. Şekil 2.2’deki XSLT dokümanı yukarıdaki XML formatları arasında dönüşümü sağlayan şablonları içermektedir.

1.kural ← 2.kural ←	<pre> <?xml version “1.0” ?> <xsl:stylesheet xmlns:xsl=“http://www.w3.org/1999/XSL/Transform version= “1.0”> <xsl:output method= “xml” /> <name><xsl:value-of select= /head/@call/></name> <contributors> <staff><xsl:for-each select =/writer/@name”/></staff> </contributors> </pre>
-----------------------------------	--

Şekil 2.2 XML doküman örnekleri

Şekil 2.2’deki XSLT dokümanındaki birinci kural, Şekil 2.1(a)’daki XML dokümanının *head* etiketinin *call* özelliğindeki veriyi *name* etiketi arasına almaktadır. İkinci kural ise yine Şekil 2.1(a)’daki XML dokümanının *writer* etiketinin *name* isimli özelliğini *staff* etiketi içinde, *staff* etiketini de *contributors* etiketi içerisine almaktadır.

Yukarıdaki basit dönüşüm örneğinden yola çıkarak esas problemimizin çözümünü XSLT transformasyon kuralları ile yazmaya çalışabiliriz. XSLT programını çalıştıran birden çok XSLT çalıştırma ortamı (*XSLT execution engine*) vardır [11]. (Apache-Xalan, Gregor/XSLT gibi)

Apache Xalan-JAVA JAVA ortamında geliştirilen bir XSLT çalıştırma ortamıdır. Bu çalıştırma ortamı XSLT ve Xpath (*XML Path Language*) dillerinin W3C tarafından belirlenen spesifikasyonlarına uygun olarak XML dokümanlarını HTML, metin (*text*) veya başka bir XML dokümanına dönüştürmeye yarayan algoritmaları içerir [11]. Xalan-JAVA version2, SAX2 (*Simple API for XML*) ve DOM (*Document Object Model*) Level 2 XML ayrıştırıcılarını içerir ve içerdiği API (*Application Programming Interface*) ile XML transformasyonlarını gerçekleyen modüler bir çalıştırma ortamı sağlar.

[8]'de HTML'den WML'e dönüşüm kurallarını belirlenmiştir. Bu XSLT dokümanında dönüşüm kuralları şablonlar halinde sıralanmıştır. [8] HTML'in en temel elemanlarını içine alacak şekilde 40 adet dönüşüm kuralını XSLT ile tanımlamıştır. Fazla önemli olmayan ve WML ile desteklenmeyen etiketler sisteme dahil edilmemiştir. Bu örnekte, XSLT programının çalışması yukarıda verilen örnekteki gibidir. XHTML girişiyle alınan her bir etiket için XSLT'de yazılmış olan kural (şablon) bu etiketi içerisindeki bilgi ile yeni bir etikete çevirir. Bu dönüşüm sırasında iç içe olan etiketlere rekürsif olarak uygun XSLT kuralları uygulanır. Böylece program sonlandığında oluşan format WML formatıdır.

Şekil 2.3'de XHTML'deki *frameset* ve *frame* etiketinin WML'e nasıl ve hangi kurallar ile dönüştürüldüğü yazılmıştır.

<pre> =<xsl:template match="frameset"> =<p align="center"> Frames found </p> =<p align="center"> please select... </p> - <!-- Head of a Frame presenting card --> <xsl:apply-templates /> </xsl:template> =<xsl:template match="frame"> - <!-- Option to select a frame --> =<xsl:for-each select="."> =<p> Goto: = <xsl:value-of select="@name" /> </p> </xsl:for-each> </pre>	<pre> =<xsl:for-each select="."> =<xsl:variable name="chosenSRC"> - <!-- Variable instantiated for the frame's name --> <xsl:value-of select="@src" /> </xsl:variable> =<xsl:variable name="hyperlink"> - <!-- Variable instantiated for the URL frame refers to --> <xsl:value-of select="card2" /> </xsl:variable> </xsl:for-each> </xsl:template> </pre>
---	---

Şekil 2.3 XHTML WML dönüşümünü yapan XSLT doküman örneği

Yukarıdaki “<xsl:” ile başlayan komutlar XSLT komutlarıdır. Mesela

```
<xsl:template match= “frameset”>
```

komutu ile üzerinde çalışılan dokümanda *frameset* etiketi yakalanır. Bundan sonra bu komutun kapanmasına kadar (</xsl:template>) olan komutlar işlenir. Yukarıdaki örnekte “Frames found” ve “please select...” paragrafları WML dokümanına yazılmaktadır. Bu etikette son satırdaki

```
<xsl:apply-templates/>
```

komutu XSLT’de rekürsif işlem yapmayı sağlayan en önemli komuttur.

<xsl:template match= “frame”> etiketi ise karşılaşılan her bir <frame> etiketi için yeni bir link tanımlamaktadır. Bu linkin adresi, ismi ve gerekli adresin atanması işlemleri bu etiket kapsamında yapılmaktadır.

XSLT rekürsif yapıları (iç içe yapılar) çözmede oldukça etkindir. Bunlar için daha önce adı geçen çalıştırma ortamlarında özel algoritmalar kullanılmıştır. XSLT programının çalışmasında karşılaşılan karmaşıklıktan bir tanesi de alınan bir etikete hangi kuralın uygulanacağıdır. Bu karar aşamasında en uygun, en yakın (*best-fit*) olan kural bulunur.

[8] çalışmasında oldukça başarılı dönüşüm kuralları işlenmiştir. Uygulamanın durgun ortamda çalışması en büyük eksikliğidir. [8]’deki sistemin aktif çalışabilmesi için bir servis sağlayıcı (*web server*) geliştirilmesi gerektiği belirtilmiştir. Böyle bir servis sağlayıcı üzerinde ASP (*active server pages*), WMLScript gibi betik dilleri (*script language*) ile XSLT kuralları aktif ortamda gelen dokümanlara uygulanabilir.

Özet olarak XHTML/WML dönüşümünde XSLT kullanmanın faydaları aşağıda maddeler halinde belirtilmiştir [8]:

- XSLT yapısı hiyerarşiktir. Rekürsif yapıya uygun olarak kurallar zinciri belirlenebilir.
- Doku tanıma (*Pattern matching*) ile en yakın kural belirlenir.
- Şablonlar çeşitli tiplerde XML dokümanları için tanımlanabilir.
- Transformasyonlar oldukça kısa ve kolay anlaşılır bir yapıdadır.
- Rekürsif yapıların yanlışını bulmak (*debuging*) daha kolaydır.

3. İNTERNET VE KABLOSUZ İLETİŞİM

Haberleşme teknolojisi günümüzde en yaygın ve en hızlı gelişen teknolojilerden birisidir. İletişim kablolu ve kablosuz olmak üzere iki farklı ortamda yapılabilir. Bu iki ortamdaki iletimin kendilerine göre özellikleri vardır. Bu iki ortamın birbiriyle haberleşmesi de yine kendine özgü standartları beraberinde getirir. Bu bölümde bu iki ayrı haberleşme teknolojisinin kendi içindeki temel yapı taşlarından WWW (*world wide web*), HTTP (*hypertext transform protocol*), HTML (*hypertext markup language*) ele alınacaktır.

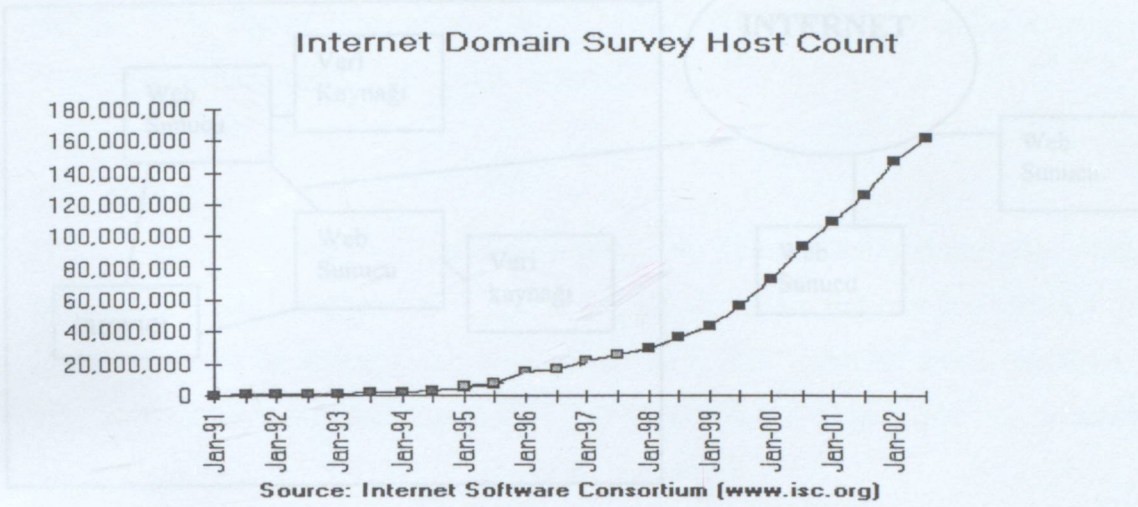
3.1 İnternet ve WWW

Günümüzde elektronik ortamda haberleşme iki önemli teknolojiye dayanmaktadır: İnternet ve WWW. Bu iki isim esasında birbirini tamamlamaktadır. İnternet fiziksel olarak dünya üzerindeki bütün bilgisayar ağlarının birbirine bağlanması ile oluşan bir altyapı; bu fiziksel altyapıdaki bütün ara bağlantılar, uç birimler ve bunların üzerinde geliştirilen her türlü standart ve protokolleri kapsar. WWW ise İnternet’de bulunan her türlü verinin erişimini ve görüntülenmesini sağlar.

Dünya üzerindeki farklı yapı ve özellikteki uç birimlerin birbiriyle haberleşmesi için izlenmesi gerekli olan bazı kurallar vardır. Protokol iki bilgisayarın haberleşmesi için gerekli olan metotlar zinciridir. İnternet değişik ölçekte ve özellikte ağlardan oluştuğuna göre her ağ için protokol yapısı farklı olabilir. Bununla beraber bütün ağların birbiri ile haberleşmesi için genel bir protokol yığını vardır. Bu protokol yığını TCP/IP (*Transmission Control Protocol / İnternet Protocol*)’dir.

İnternet’e ilk adım 1969’da çeşitli bilgisayar bilimleri ve askeri araştırma projelerini desteklemek için Amerikan Savunma Bakanlığı’nın ARPANET adında Paket Anahtarlama Ağ’ı geliştirmesiyle atıldı [1]. Daha sonraki yıllarda bu ağa üniversite ve araştırma kuruluşlarının değişik tipteki bilgisayarları da eklenince, ağ için bir protokol seti geliştirmek ihtiyacı ortaya çıktı. Bu amaçla 1973 yılında Stanford Üniversitesinde - daha sonra BBN’in ve “University College, London”ın da dahil olduğu - bir ağlar arası haberleşme (*internetworking*) projesi başlatıldı. Bu proje ile beraber daha kapsamlı standartlar ortaya atıldı. TCP (*Transmission Control Protocol*) bunlardan en önemlisi ve en kapsamlı olanıdır. 1983’te tüm ARPANET kullanıcıları İletim Kontrol Protokolü/İnternet Protokolü (*TCP/IP*) olarak bilinen yeni protokole geçiş yaptılar. ARPANET 1990’da kullanımdan kaldırıldı[1]. 20 yıllık bir geçmişi olan İnternet ve teknolojileri günümüzde en hızlı gelişen teknolojilerden birisidir. Şekil 3.1’deki grafik İnternet’te kayıtlı olan uç birimlerin (*host*) 1991’den günümüze artışı

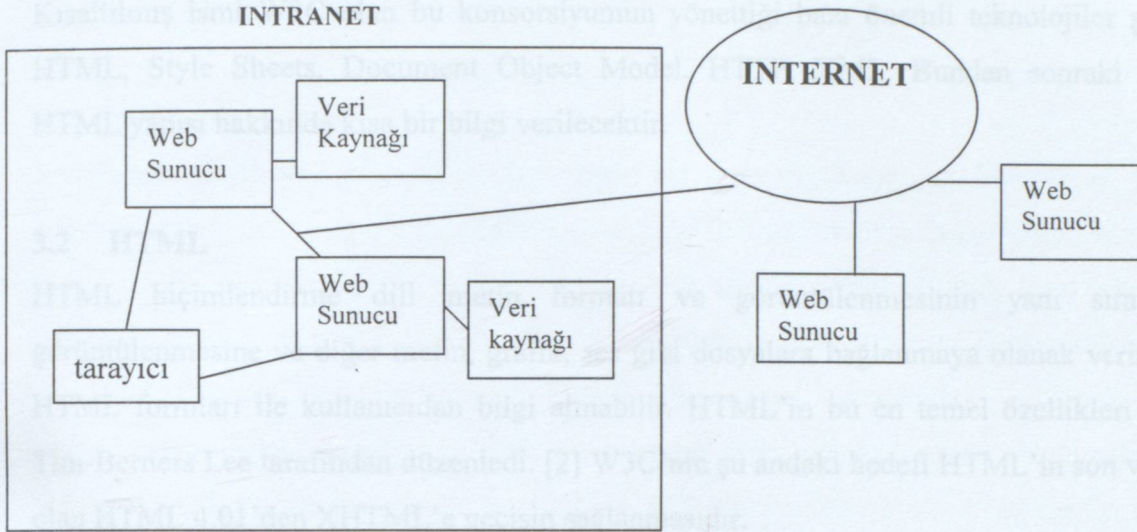
göstermektedir.



Şekil 3.1 İnternetteki uç birimlerin senelere göre artışı

TCP/IP genel bir protokol yığınıdır. Bu katmanlı protokol yapısındaki her katmanda farklı özellikte standartlar tanımlanmıştır. Bununla beraber İnternet'te çeşitli servisleri her tip bilgisayara sunmak hedeflendiği için protokol çeşitliliği oldukça fazladır. Terminal hizmetleri (TELNET protokolü), elektronik posta hizmetleri (SMTP protokolü), dosya gönderme ve alma protokolü (FTP protokolü) ilk akla gelen en temel servisler ve bu servislerle ilgili protokollerdir. Günümüzde İnternet geliştirilen protokoller ve sunulan servisler açısından seneler önce yapılan tahminlerden çok daha ileri noktalara gelmiştir. Gerçek zamanlı uygulamalar, çoğulortam (*multimedia*) uygulamaları, uzaktan eğitim, yüksek güvenlik gerektiren uygulamalar, veri tabanına hızlı erişimi gerektiren gerçek zamanlı uygulamalar, ticari uygulamalar, elektronik kütüphaneler ve daha başka bir çok halka açık servis veya özel uygulamalar İnternet üzerinde geliştirilmektedir.

WWW genel olarak hypertext ve çoklu ortam bilgi erişim sistemidir. WWW'in grafiksel bir görüntüsü Şekil 3.2'de sunulmuştur [1].



Şekil 3.2 WWW'in yapısal birimleri

WWW'in temel teknoloji yapı taşları kullanıcı/servis sağlayıcı modeli (client/server model), ağ protokolleri (HTTP vb), adresleme ve biçimlendirme dilleri gibi konuları kapsamaktadır[1].

Kullanıcı/servis sağlayıcı modeli haberleşme sistemlerinin temelini oluşturur. Servis sağlayıcı genel olarak tarayıcıdan (*browser*) gelen isteklere cevap verir, servis sağlayıcıdaki betiklerin (*scripts*) koşturulmasını sağlar ve çeşitli istatistiklerin tutulmasına olanak verir. Servis sağlayıcının temel özellikleri arasında çalıştığı platform, eşzamanlı iş parçacıklarını (*multithreaded*) desteklemesi ve güvenlik gibi konular vardır.

Kullanıcı ile servis sağlayıcı birbiriyle haberleşmeyi HTTP protokolü ile gerçekleştirir. Az karmaşıklıkta HTTP protokolünün şu an kullanılan versiyonu HTTP 1.1'dir. HTTP, bir uygulama protokolüdür. (TCP/IP katmanlı protokolünün en üst katmanı)

Internet'teki her uç birimin bir adresi vardır. Kullanıcı istediği bilgiye bu adres ile ulaşır. Internetteki her uç birime özel olan bu adrese IP (*Internet Protocol*) adresi denir. IPv4 (*IP version 4*) 32 bitlik adres bilgisinden oluşmaktadır.

WWW üzerinde biçimlendirme dili HTML'dir. HTML ile bilginin dokümanda yerleşimi ve bu sayfadan diğer metin, resim, ses, video gibi nesnelere bağlantısı belirlenir. Kullanıcıdaki tarayıcı (*browser*) ile servis sağlayıcıdan alınan HTML sayfası görüntülenir.

1994'te Tim-Berners Lee tarafından kurulan WWW Konsorsiyumunun amacı, [2] WWW'in en yüksek performansla kullanılmasına olanak veren standartların geliştirilmesini sağlamaktır.

Kısaltılmış ismi W3C olan bu konsorsiyumun yönettiği bazı önemli teknolojiler şunlardır: HTML, Style Sheets, Document Object Model, HTTP, XML. Bundan sonraki bölümde HTML yapısı hakkında kısa bir bilgi verilecektir.

3.2 HTML

HTML biçimlendirme dili metin formatı ve görüntülenmesinin yanı sıra, grafik görüntülenmesine ve diğer metin, grafik, ses gibi dosyalara bağlanmaya olanak verir. Ayrıca HTML formları ile kullanıcıdan bilgi alınabilir. HTML'in bu en temel özellikleri 1990'da Tim-Berners Lee tarafından düzenledi. [2] W3C'nin şu andaki hedefi HTML'in son versiyonu olan HTML 4.01'den XHTML'e geçişin sağlanmasıdır.

HTML dokümanlarını görüntüleyen Web tarayıcıları (*browser*) HTML dilinin gelişmesine paralel olarak günümüze kadar çeşitli aşamalardan geçmişlerdir. Bunlardan en önemlileri "Microsoft Internet Explorer" ve "Netscape Navigator"dur. Bu tarayıcılar HTML dilindeki olası hatalara karşı mümkün olduğunca duyarsızdır. Amaç ağ band genişliğini en verimli kullanarak, gelen bir HTML dokümanını görüntülemeye çalışmaktır. Bu esnek özellikten dolayı HTML sayfaları farklı tarayıcılarda farklı görüntülenebilir.

HTML yüz otuza yakın biçimlendirme elementinden oluşur. Bir elementin yapısı şu şekildedir[1]:

<başlangıç etiketi> *içerik* <sonlanma etiketi>

Her elementin kullanabileceği özellikler (*attributes*) vardır. Bu özelliklerin yapısı da

isim = "değer"

şeklinde dir.

Bu bilgiler ışığında basit bir HTML dokümanı Şekil 3.3'deki gibidir:

```
<html>
  <head>
    <title> Örnek bir HTML!!</title>
  </head>
  <body>
    <p id = "ilk_paragraph"> Merhaba HTML </p>
  </body>
</html>
```

Şekil 3.3 Basit bir HTML dokümanı

HTML dilinin mevcut etiketleri, özellikleri ve kullanımları [2]'de incelenmiştir.

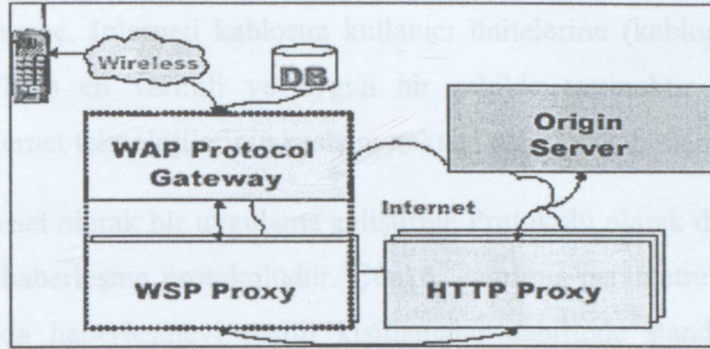
3.3 Kablosuz İletişim

Günümüz haberleşme teknolojilerinin diğer bir kolu da Kablosuz İletişimdir. Kablosuz ortamın kullanıcıya sunduğu hareketlilik (*mobility*) Internet ortamında kullanılan alışlagelmiş protokollerin çoğunu doğrudan kullanılmaz hale getirmektedir. Bugün için ortaya atılan kablosuz iletişim protokollerinin yüksek güvenilirlikte, karmaşıklığı az ve maliyeti düşük çözümler üretmesi beklenmektedir. Ortaya çıkan yeni teknolojiler ve farklı kullanıcı uygulamaları doğrultusunda kablosuz ortamda bir çok protokol vardır. Kablosuz iletişim kablosuz telefon haberleşmesiyle günümüzde etkinliğini arttırarak sürdürmektedir. Kablosuz telefonlar düşük enerjili radyo dalgalarını en yakın yerel antenlere iletirler. Bu antenler bağlantı isteğini ya başka bir yerel antene veya kablolu telefon haberleşme şebekesine yönlendirirler. Hücresel yapının kullanıldığı kablosuz telefon haberleşmesinde sistemin kurulacağı bir bölge hücre adı verilen daha küçük coğrafi bölgelere ayrılır (Onur, 2000). Birbirine komşu olan hücrelerin kullandığı frekanslar farklıdır. Böylece bir hücreden başka bir hücreye geçen hareketli kullanıcı daha önce bulunduğu hücredeki frekansını bırakıp yeni hücreden bir frekans almalıdır (*handoff*). Ancak birbirine uzak olan hücreler aynı frekansları kullanabilir. Frekans tekrar kullanılması (*frequency reuse technique*) adı verilen bu teknik ile hücresel kablosuz iletişimde daha çok kullanıcıya hizmet verilebilir. Bununla beraber TDMA (*Time Division Multiple Access*), GSM (*Global Standard for Mobile Communications*) ve CDMA (*Code Division Multiple Access*) gibi mevcut sayısal iletişim teknolojileri ile frekans kullanımında yüksek verimler elde edilmektedir [3].

WAP ilk olarak 1998'de WAP Forum tarafından geliştirildi. CDPD, CDMA, GSM, PDC, PHS, TDMA ve daha başka bir çok fiziksel iletişim ortamı için tasarlanan WAP protokolünün amacı Internet içeriğini kablosuz kullanıcıya aktarmaktır.

Bilindiği gibi Internet'teki veri standart HTML formatında iletilir. Bu yüzden WAP kullanıcısı ile Internet arasında olması gereken arayüzde (*wireless service provider*) HTML kodu WML'e dönüştürülerek kullanıcıya aktarılır. WAP Arayüzü WWW protokol yığını (HTTP, TCP/IP), WAP protokol yığına (WSP, WTP, WTLS, WDP) dönüştürür. Böyle bir protokol yığında yapılan dönüşümlerin her biri zaman alan, karmaşıklığı yüksek operasyonlardır. Böyle bir zaman kaybı direk olarak veri ulaşım zamanını olumsuz olarak etkiler. Örneğin Internet'in yaygın olan SSL şifrelemesinin WTLS'ye dönüşümü, HTML'in WML'e dönüşümü ve diğer katmanlardaki dönüşümler belirli karmaşıklıkta olan işlemlerdir.

WSP ile Internetteki hedef servis sağlayıcı (*origin server*) arasındaki bağlantı bir veya birden çok arayüz üzerinden gerçekleştirilir. Şekil 3.4’de bir cep telefonunun Internet üzerinden bir servis sağlayıcıya erişimi şematize edilmiştir [4].



Şekil 3.4 Cep telefonu Internet erişimi

Bu erişim bir veya birden çok arayüz üzerinden gerçekleşir. Böyle bir arayüz tasarımı ve uygulanması bu tezin konusudur.

4. KABLOSUZ ORTAMDA UYGULAMA GELİŞTİRME PROTOKOLÜ

WAP, kablosuz haberleşme teknolojilerine dayalı olarak geliştirilen uygulamaları standard bir yapıya getirmeyi hedefleyen bir protokol yığıdır.

Birincil öncelikli amaç, İnterneti kablosuz kullanıcı ünitelerine (kablosuz ağ cep telefonu, çağrı cihazları, PDA) en verimli ve uygun bir şekilde taşımaktır. WAP kablosuz ağ teknolojileri ile İnternet teknolojilerinin kesişim noktası olarak düşünülebilir.

WAP Protokolü genel olarak bir uygulama geliştirme Protokolü olarak düşünülebileceği gibi, aynı zamanda bir haberleşme protokolüdür. Çünkü, dağılmış parametrelili bir sistemde farklı platformlar arasında haberleşmeyi çeşitli kısıtlamalar dahilinde standart hale getiren bir protokoldür. Bu noktalar göz önüne alınarak, WAP'ın HTML ve HTTP protokolleri örnek alınarak band genişliği, hafıza ve görüntü kapasite sınırlamalarının olduğu ortamlar için tasarlandığı söylenebilir.

WAP Protokolü "WAP Forum" adı verilen ve Nokia, Ericsson, Phone.com ve Motorola şirketleri ile finanse edilen bir Forum tarafından düzenlenip, yönetilmektedir [6].

WAP teknolojileri, WWW teknolojilerinin temel yapı taşlarını içerisinde barındırmaktadır. Buna ek olarak bir WAP uygulaması mevcut WWW altyapısına erişebilir ve oradan servis alabilir. Böylece İnternet üzerindeki mevcut servis çeşitliliği kablosuz iletişim ortamına da taşınmış olacaktır. En azından hedeflenen budur. Bazı özel servisler (elektronik posta, kısa mesaj servisi gibi) kullanıcı isteğine bağlı olarak ayrı bir ücretlendirmeye tabi tutularak verilebilir. Kullanıcı-Servis sağlayıcı haberleşme yapısının (*client-server methodology*) yanı sıra, WWW teknolojilerine dayandırılan önemli diğer anahtar özellikler URL modeli, XML, kullanıcıdan veri aktarımı (*content-typing*), standart içerik yapıları (*content formats*) ve standart haberleşme protokolleri (*TCP-IP*) gibi İnternet teknolojilerinin olmazsa olmaz özellikleri WAP protokolleri için temel özelliklerdendir.

Buna göre WAP Forumun temel hedefleri aşağıdaki maddeler ile belirlenmiştir (Mann, 1999):

- İnternet ortamındaki verinin ve sunulan çeşitli servislerin kablosuz ortama (dar band genişlikli güvenli olmayan iletim ortamları) uygun hale getirilmesi
- Farklı kablosuz ağ teknolojileri üzerinde çalışabilecek standartlaşmış genel bir kablosuz katmanlı ağ Protokolü geliştirmek
- Kablosuz ortamdaki veri ve servislerin kablolu ortama aktarılması ve oradan verinin alınmasını sağlayan standartlar geliştirmek

- Oluşturulacak yeni hedef ve standartların mevcut teknolojiler ve standartlar ile birlikte işler olması ve ölçeklenebilir olması.

Internet ortamında geliştirilen teknolojilerin kablosuz ortamda kullanılamamasının nedenlerinin başında bu teknolojilerin geniş band genişlikli güvenilir iletim ortamları için tasarlanması, uç birimlerin (masa bilgisayarları ve daha büyük bilgisayarlar) yüksek işlemci gücüne ve hafıza kapasitesine sahip olması yatmaktadır. Bunların yanında cep telefonunu ele aldığımızda karşımıza çıkan temel kısıtlamalar işlemci gücünün düşük olması, hafıza kapasitesinin düşüklüğü, iletim ortamı band genişliğinin darlığı, görüntü kalite ve boyutunun sınırlı olması, güç kaynağı sınırlamaları gibi önemli kısıtlamalar Internet teknolojilerini doğrudan kullanmayı olanaksız hale getirmektedir. Bütün bu kısıtlamalar yanında kablosuz ortamda uç birimlerin sabit olmaması kablosuz ağ teknolojilerine ayrı bir boyut kazandırmaktadır. Geliştirilen WAP teknolojileri Internet teknolojileri ile kablosuz ortam standartlarını ve teknolojilerini buluşturmaya hedef almaktadır. Bu hedefin yanı sıra WAP ile kablosuz ortamda bazı servislerin (çağrı yönlendirme gibi) iyileştirilmesi sağlanabilir.

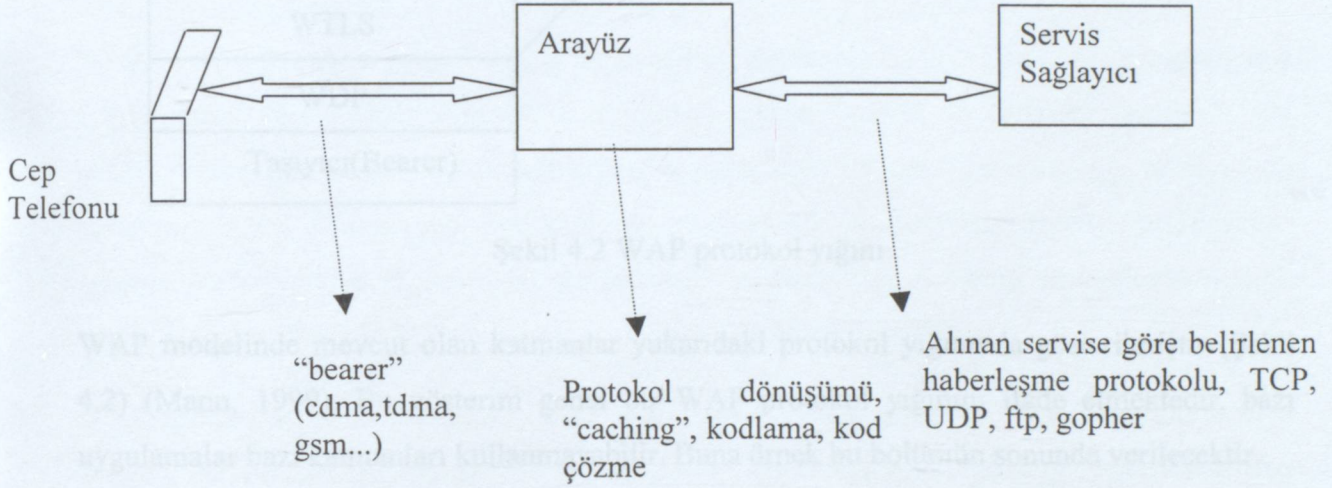
4.1 WAP Modeli

WAP programlama ortamı temel olarak WWW programlama ortamına benzerdir. Bu benzerlik WAP'ın Internet ortamı ile entegrasyonunda ve yeni standartların WAP modeline eklenmesinde büyük kolaylıklar getirecektir. Aşağıdaki şekilde Arayüz (*Gateway veya Proxy*) temelde kodlayıcı ve kod çözücü (*encoding-decoding*) işleminin olduğu bir arabirimdir. Bu tarz bir arabirimin istenilen özelliklerde tasarımı ve gerçekleştirilmesi bu tezin konusudur.

Genel anlamda bir Arayüz (*Proxy Server*) kullanıcıdan alınan istekleri ya servis sağlayıcıya iletir veya kendi ön belleğinde varsa isteğe bununla karşılık verir. Böylece Arayüz kullanıcı için bir servis sağlayıcı olabildiği gibi, isteği ilettiği servis sağlayıcı için de kullanıcı modunda çalışabilir. Bunlara ek olarak Arayüz istekleri filtreleyebilir ve bazı istatistiksel bilgiler de tutabilir.

Şekil 4.1'deki servis sağlayıcı istenilen verinin veya servisin bulunduğu uç birimdir. Cep telefonu tarafından istenen servis (cep telefonu yerine herhangi bir kablosuz ağ uç birimi olabilir) kablosuz ortamda bulunan haberleşme Protokolü ile (cdma, tdma, gsm gibi teknolojiler) istenilen servisten bağımsız olarak Arayüze, daha sonra buradan Servis Sağlayıcıya iletilir. Bu noktada Arayüz ile Servis Sağlayıcı arasında kullanılan protokol istenen servise bağlı olarak değişir (Mann, 1999). Kablosuz kullanıcı uç birimi ile ağ servis sağlayıcılarının arasındaki iletişim standartları WWW ortamında kullanılan standartlara

benzerdir. Yukarıda bahsedilen haberleşme protokolleri yanı sıra adresleme (*URL naming*), veri içeriğinin tipi (*content typing*), verinin formatı (*content format*) gibi standartlar WAP ortamına uygun olarak geliştirilmiştir. WAP ortamındaki bu standartlarda bulunan veri genel olarak bir WAP Arayüz ile WWW ortamına entegre olur. Bu Arayüzde WAP ortamındaki protokol WWW ortamındaki protokollere dönüşümü yanı sıra veri kodlama, veri miktarının azaltılması, sıkıştırılması ve istenen servise bağlı olarak diğer işlemlerden geçirilir (Mann, 1999). Bu tezde böyle bir arabirimde html dilinden, wml diline dönüşüm gerçekleştirilecektir.



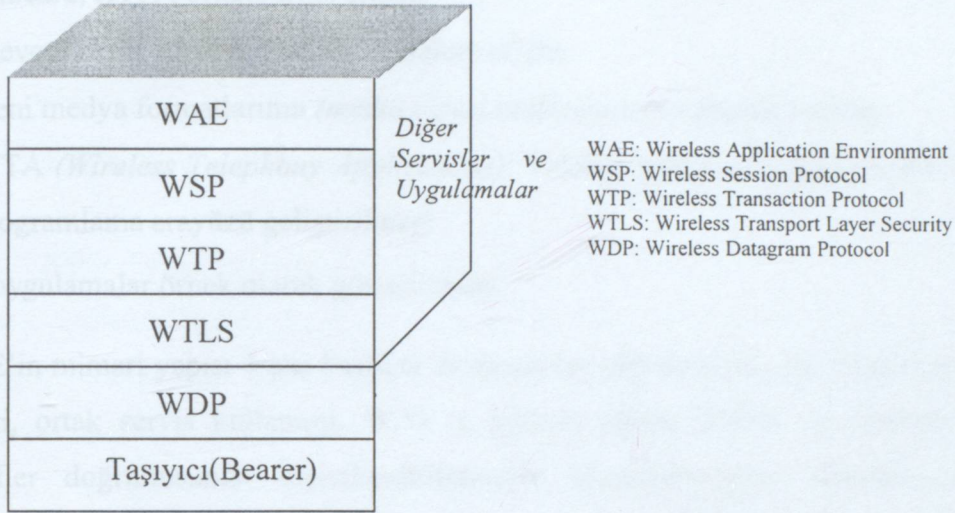
Şekil 4.1 Cep telefonu, arayüz, servis sağlayıcı ilişkisi

Mesela servis sağlayıcı Arayüze HTML dokümanı gönderirse bu kod wml koduna dönüştürülür ve daha sonra band genişliği yönünden daha verimli olması için “binary” olarak hareketli (*mobile*) cihaza iletilir.

Tipik bir WAP yapısı yukarıda belirlenen yapı birimlerinden oluşmaktadır. Bununla beraber Arayüz işlemleri servis sağlayıcıda da gerçekleştirilebilir. Bu durumda Arayüze gerek kalmayacaktır.

Yukarıda incelenen ağ yapısı, aralarında bir “HTTP proxy” yardımıyla haberleşen “web tarayıcı” ile “servis sağlayıcı”ya benzetilebilir. Böyle klasik bir ağda proxy, mesela alınan bir ftp listesini html dokümanına dönüştürme işlemini gerçekleştirir.

4.2 WAP Protokol Yığını



Şekil 4.2 WAP protokol yığını

WAP modelinde mevcut olan katmanlar yukarıdaki protokol yığnında gösterilmiştir (Şekil 4.2) (Mann, 1999). Bu gösterim genel bir WAP protokol yığnını ifade etmektedir, bazı uygulamalar bazı katmanları kullanmayabilir. Buna örnek bu bölümün sonunda verilecektir.

Diğer mevcut protokol yığnlarında olduğu gibi her katmandaki servis bir üst katman tarafından kullanılabilir. Böyle bir protokol yığnında hem katmanlar arası bilgi akışı, hem de diğer dış servislerin her bir katmandaki servise erişimi belirli ve tanımlanmış arayüzler ile sağlanır. Aşağıda her katmanın işlevi ve sunduğu servisler kısaca açıklanmıştır (Mann, 1999).

4.2.1 WAE (Wireless Application Environment)

OSI (*Open System Interface*) yapısında uygulama katmanına karşılık gelir. Uygulama katmanı adı verilen bu katmanda tanımlı standart ve protokoller WWW ve telefon teknolojilerini de kullanabilen WAP uygulamaları için tanımlanmıştır. Farklı kablosuz iletişim ortamlarına erişim bu katman ile sağlanır. Sınırlı band genişlikli iletim ortamları ve sınırlı hafıza, ekran, güç kaynağı ve işlemci gücüne sahip olan uç birimler için uygun uygulamalar geliştirmek WAE'de geliştirilen standartların en başta gelen hedefleridir. Bu katmandaki uygulamalar ve yeni hedefler arasında

- ön belleğe alma (*caching*) : Servis sağlayıcıya erişim sıklığını azaltarak, band genişliği verimini artırmaya hedefler
- uçtan uça haberleşmede güvenlik artırımı: WAP'ın mevcut olan güvenlik katmanı (WTLS) ile uyumlu çalışabilecek bir artırım

- Yeni teknolojinin gerektirdiği ortamlara erişim ve entegrasyonun sağlanması: Smartcard, Simcard, JAVA ortamlarına erişim
- Mevcut akıllı ağ ve bağlantılı devrelere erişim
- Yeni medya formatlarının (*media types*) kullanılmasına olanak vermesi
- WTA (*Wireless Telephony Applications*): Telefon haberleşmesine yönelik fonksiyonlar ve programlama arayüzü geliştirilmesi

gibi uygulamalar örnek olarak gösterilebilir.

WAE'in mimari yapısı 4 ana başlıkta incelenebilir: Ağ tasarımı, veri formatları, programlama dilleri, ortak servis kullanımı. WAE'in mimari yapısı WWW ve Internet yapısının yeni hedefler doğrultusunda biçimlendirilmesiyle oluşturulmuştur. Örneğin, ağ tasarımında adresleme (*URL Addressing*), yönlendirme (*routing*) gibi fonksiyonlar Internet ortamındaki çalışma prensipleri ve algoritmalar ile benzerlik göstermektedir. İletim de kablosuz ortama uygun hale getirilmiş HTTP'ye benzer protokoller ile gerçekleştirilir.

WAE modeli uç birimlerindeki (*WAE user agent*) yazılımlar veri içeriğinin kullanıcıya düzgün bir şekilde gösterilmesini sağlar. Mobil uç birimindeki tarayıcı (*micro-browser*) WML ve WMLScript dilleri ile sunulan veri içeriğini derleyerek kullanıcıya sunar. Örneğin, cep telefonu kullanıcısına erişim ufak ölçekli wml görüntüleyicisi ile gerçekleşir.

4.2.2 WSP (Wireless Session Protocol)

WSP, Oturum Katmanı, WAE katmanına 2 farklı session servisi sağlar. Birincisi, WTP (*Wireless Transaction Protocol*) katmanından devre bağlantılı (*connection-oriented*) özellikte sunulan servisi; diğeri WDP (*Wireless Datagram Protocol*) katmanından bağlantısız (*connectionless*) özellikte sunulan servisi WAE uygulama katmanına taşımaktır.

Bu seviyede temel *http1.1* fonksiyonları gerçekleştirilir. Oturum katmanında gerekli temel uygulamalar, mesela; uzun ömürlü oturumlar, veri içeriğinde uyum (*content-type negotiation*), oturma ara verme (*interrupt*), farklı güvenilirlikteki oturumların oluşturulması fonksiyonları bu seviyede gerçekleşen fonksiyonlara örnek olarak verilebilir.

4.2.3 WTP (Wireless Transaction Protocol)

WTP karşılıklı istek/cevap (*request/reply*) uygulamalarına olanak veren bir işlem (*transaction*) ortamı tanımlayan bir protokoldür. 3 farklı özellikte işleme olanak verilir: Güvenli olmayan tek yönlü istek (*unreliable one-way request*), Güvenli tek yönlü istek (*reliable one-way request*), Güvenli olmayan iki yönlü istek-cevap (*reliable two-way request*).

reply).

Daha çok hafif yüklü işlemlere (*lightweight transactions*) uygun olarak tasarlanmış uygulamaları içeren bu katman güvenlik katmanı veya direk olarak WDP katmanı üzerinde çalışır. WDP katmanından sunulan servisleri, işlem (*transaction*) ortamında daha güvenilir bir hale getiren fonksiyonlar bu katmanda yer alır.

4.2.4 WDP (Wireless Datagram Protocol)

Fiziksel katmanın hemen üzerinde bulunan WDP, WAP protokol yığınının adeta bir taşıma (transport) katmanıdır. Fakat WWW'den farklı olarak TCP Protokolü kullanılmaz. Bunun yerine verinin bir çok paketlere bölünmesini engelleyen UDP Protokolü kullanılır. Böylece kablosuz iletişime uygun olarak servis isteği doğrudan WAP Arayüz ile iletişim kurabilecektir. TCP'nin güvenilirlik amacı ile getirdiği yükten UDP kullanarak kurtulmuş olmak bu tip bir kablosuz iletişimde beklenen bir yoldur.

WDP, üst katmanların farklı fiziksel katmanlara ve taşıyıcılara erişiminde gerekli arayüz ve standartları sağlar.

4.2.5 WTLS (Wireless Transport Layer Security)

Güvenlik katmanı olup SSL (*Secure Sockets Layers*) ve/veya TLS (*Transport Layer Security*) protokolleri ile şekillenebilir. Dar band genişlikli iletim ortamlarında güvenlik standartlarını gerçekleştirir. Esas olarak veri bütünlüğü (*data integrity*), kişisel gizlilik (*privacy*), kimlik denetimi (*authentication*) işlemleri gerçekleştirilir. Bununla beraber WTLS katmanındaki fonksiyonlar ile Internet SSL şifrelemesi arasındaki geçiş kaçınılmaz olarak güvenlik sorunu yaratan bir operasyondur.

Üst katmanlar, verdiklere servisin ihtiyacına uygun olarak bu güvenlik katmanından gerekli olan servislere ulaşabilirler.

4.2.6 Taşıyıcı (Bearer)

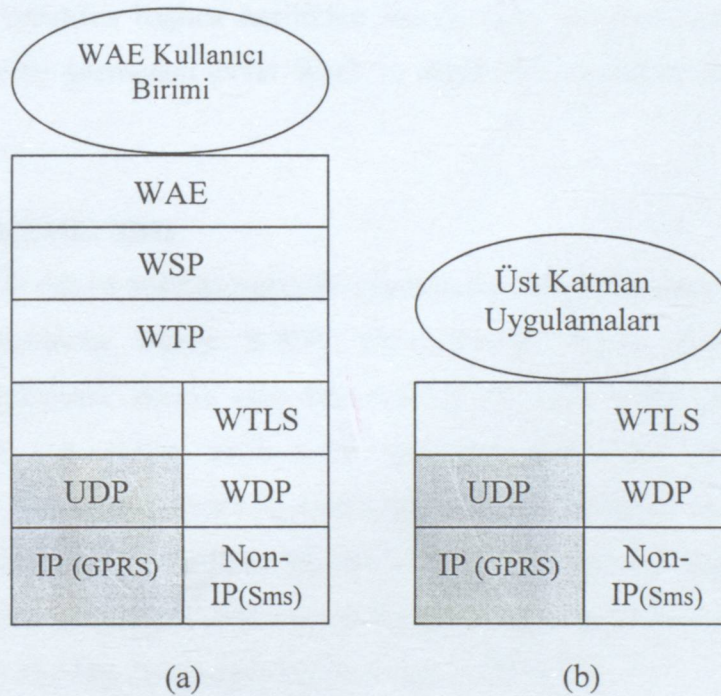
Bilginin fiziksel ortamda taşınmasına olanak veren en alt düzeydeki iletim, haberleşme protokollerini içerir. Yukarıda anlatılan protokoller farklı tipte taşıyıcı servisler (kısa mesaj, devre bağlaşmalı iletim paketi, paketlenmiş bilgi) için tasarlanmışlardır. Mevcut iletim ortamı protokolleri yanı sıra günümüzde yeni taşıyıcı standartları geliştirilmektedir. Verim (*throughput*), hata olasılığı (*error-rate*), gecikme (*delay*) farklı düzeyde servis kalitesi kriterleridir. GSM, CDMA, IS-136, PHS gibi teknolojiler bu katman protokollerine örnek

olarak verilebilir.

4.2.7 Diğer Servisler ve Uygulamalar

WAP yığını dışındaki diğer uygulama ve servisler şekilde görüldüğü gibi WAP üst katmanlarına direk olarak iyi-tanımlanmış arayüzler ile erişebilmektedir. Diğer uygulamalar katmanına örnek olarak elektronik posta, elektronik adres defteri, takvim, elektronik ticaret uygulamaları; diğer servislere örnek olarak da çeşitli ilan sayfalarının yayımlanması (*yellow pages broadcasting*) verilebilir.

4.2.8 Örnek Protokol Yığınları



Şekil 4.3 Örnek protokol yığınları

Şekil 4.3(a)'daki yığın tipik bir WAP uygulaması olarak ele alınabilir. WAE kullanıcısı (örneğin cep telefonu kullanıcısı) eğer WWW erişimi olacaksa UDP ve IP protokolleri ile servis alır; (bunlar WAP protokol yığını dışındadır) eğer kısa mesaj haberleşmesi (*sms, short message service*) gibi kablosuz ağ içerisindeki bir servis hedefleniyorsa WDP ile bu servise ulaşılır (Mann, 1999).

Şekil 4.3(a)'daki WAP uygulamasında sadece WDP (*Datagram Transport*) katmanı kullanılır. Böyle bir uygulama kablosuz ağ üzerinde uçtan-uça kontrol veya işlem (*transaction*) ile ilgili işaretleşmeler olabilir (Mann, 1999).

4.3 Kablosuz Ortamda Biçimlendirme Dili (Wireless Markup Language-WML)

WML, XML tabanlı bir biçimlendirme (*markup*) dilidir. Diğer bir ifadeyle, WML bir XML uygulamasıdır. WML dilinin en önemli özellikleri aşağıdaki gibi sıralanabilir (Mann, 1999):

- Metin (*text*) ve görüntü (*image*) desteği
- Kullanıcı bilgisi girilmesine olanak vermesi
- Dolaşmaya (*Navigation*) olanak veren aktif linkleri içermesi
- Çok uluslu kullanım (*Internationalization*)
- Kablosuz ortamın band genişliği kısıtlamalarına uygun bir tasarım ("deck of cards" tasarımı)

WML yukarıda belirtilen başlıca özellikleri tez ile ilgisi dahilinde incelenecektir. WML'in yapısal özelliklerine geçmeden evvel WML'in diğer biçimlendirme dilleri ile ilgisi aşağıda anlatılmıştır.

4.3.1 SGML, HTML, XML

WWW fikrinin ilk ortaya atıldığı zamanki uygulamalarının günümüzde çok değişik boyutlara taşındığı görülmektedir. Bütün WWW teknolojilerine dayalı olan bu büyük boyutlu uygulamaların(elektronik ticaret gibi) her yönden en ucuz yollar ile dağıtılıp işlenmesi kullanıcılar için (ve servis sağlayıcılar için de) güçlü bir veri işleme kapasitesi gerektirmektedir. Çünkü, bu fonksiyonellik sadece HTML'in başarı ile gerçekleştirdiği sabit formasyonda dokümanların dağıtılıp işlenmesi ve görüntülenmesi değildir. Dağıtılmış veri işleme (*Distributed Document Processing*) karmaşıklığına getirilmeye çalışılan bir çözüm XML (*eXtensible Markup Language*) biçimlendirme dilidir [7].

Bu noktada "XML denilen biçimlendirme dili ile, HTML ile gerçekleştirilemeyen hangi uygulamalar gerçekleştirilebilir?" sorusu sorulabilir [7]. Örneğin:

- Veri tabanı transferi (*database interchange*) adı verilen ve kullanıcının iki veya daha fazla farklı veri tabanı sistemleri arasında aracı olması problemi (daha sonra açıklanacağı üzere DTD denilen dosyaların kullanımı)
- Bilginin farklı kullanıcılara farklı sunulması problemi, kişiye özel profiller (*profiles on web sites*)
- Bilginin hepsinin sunulması yerine kullanıcı ihtiyacı ölçüsünde bilginin filitrelenmesi (*tailoring*) edilmesi
- Servis sağlayıcının işlem yükünün kullanıcılara dağıtılması

Öte yandan XML, HTML ve diğer doküman formasyonu dillerini kapsayan kümeye SGML (*Standard Genaralized Markup Language*) biçimlendirme dili adı verilir. SGML temel olarak bir dokümanın şekillenmesinde 3 anahtar kuralı baz almaktadır [1]. Bunlar:

- Kapsam (*Extensibility*): Mesela HTML’de tag ve özellikler (*attribute*) belirlidir ve HTML’in o versiyonu için değişmezler. XML ise kullanıcının yeni tag ve özellik listelerini tanımlanmasını ve dokümanların yapılarını tanımlamaya olanak verir.
- Yapı (*Structure*): Örneğin yapı itibariyle html derin (iç içe) yapılara olanak vermezken, XML’de dokümanın yapısal özellikleri ihtiyaca göre istenilen derinlikte tasarlanabilir.
- Gerçekleme (*Validation*): HTML dosyasının işlenmesinde büyük ölçüde hatalara duyarlı bir yaklaşım tasarlanmıştır. Sözdizimi (*syntax*) ve anlam (*semantic*) hataları olsa bile doküman görüntülenir. Oysa XML’de belirli sözdizimi ve yapısal kontroller vardır.

XML ve HTML biçimlendirme dillerinden başka yine SGML kapsamında başka biçimlendirme dilleri vardır. (Örneğin: Adobe PDF, MS RTF, MS Word). Fakat XML, SGML’in yukarıda belirtilen anahtar özelliklerini koruyan WWW’in karmaşık uygulamaları için tasarlanmış bir biçimlendirme dilidir [1].

SGML uygulamalarında DTD (*document type definition*) adı verilen dosya ile oluşturulan markup dilinin yapısı, sözdizimi kuralları, elemanları, elemanların anlamları ve elemanların özellikleri belirlenir.

Bir XML uygulaması olarak önceki paragraflarda da bahsedilen veri tabanı transferi (*database interchange*) uygulaması için eğer XML kullanılacaksa öncelikle ilgili DTD dosyaları tanımlanması gerekir [1]. DTD dosyaları ilgili, biçimlendirme dilinin gramerini ve kullanılan elemanları tanımlamaktadır. SGML’in 3 anahtar özelliğine olanak veren bu yaklaşım ile çeşitli endüstri sektörleri kendi içlerinde ortak yapı ve elemanları belirleyebilirler. Mesela sağlık sektörü kendi sistemleri arasındaki dokümanlarında kendi eleman ve yapısal özellikleri kullanır ve bunları kendi DTD’si içerisinde düzenlerken, havacılık sektörünün de kendi DTD’si olabilir.

Bu bağlamda XML tek başına gerçek manada bir biçimlendirme dili değildir. Fakat yeni biçimlendirme dilleri tanımlamaya olanak veren bir sistemdir.

SGML, HTML, XML ve DTD kavramları hakkındaki bu kısa bilgilerden sonra şimdi WML markup dilinin elemanları ve yapısı hakkında bilgi verebiliriz.

4.3.2 WML Yapısı ve Elemanları

WML bir XML uygulamasıdır. Demek ki, WML de kendine ait elemanları ve bir yapısı olan XML dilidir, ve bu yapıyı tanımlayan ilgili bir DTD olacaktır.

Bir mobil telefonun (veya bir kablosuz haberleşme cihazının) WAP uyumlu olmasıyla, cihazın mini tarayıcı (*microbrowser*) adı verilen yazılıma sahip olduğu anlaşılır. Bu yazılım ile mobil telefonu WML DTD dosyasında bulunan bütün varlıkları (*entity*) işleyebilir (Mann, 1999).

Bütün WML dosyaları işte bu DTD dosyasına işaret eden 2 satırlık prolog ile başlar [6].

```
<xml version='1.0'>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
```

Bütün XML dokümanlarında doküman elemanı denilen bir eleman diğer bütün elemanları DTD'de belirlenen kurallar çerçevesinde kapsar. WML için bu eleman <wml>dir. Diğer elemanlar belirli kurallara göre <wml> ile </wml> arasında yer alır.

Her elemanın belirli özellik seti (*attribute set*) vardır. Mesela aşağıdaki örnekte <card> elemanı "id" ve "title" özellikleri ile tanımlanmıştır.

```
<!-- This is the Login Card -->
<card id="LoginCard" title="Login">
  Lütfen isminizi seçiniz!
</card>
```

WML yapısal düzenindeki en karakteristik özellik "card" denilen birimlerin bir araya gelerek "deck" adı verilen üniteleri oluşturmasıdır. "Deck", mobil cihaz ile WAP Arayüz arasındaki en küçük kod ünitesidir. "Card" ise kullanıcı ile mobil telefon cihazı arasındaki kod parçasıdır. Diğer bir ifade ile "card" bir ekrana sığması beklenen kullanıcı etkileşimli kod parçasıdır.

Böylece her "card" için Arayüz'e ulaşım yükü olmayacaktır. Mobil, bir bağlantıda aldığı "deck" ile ("deck" html'de "document" elemanına karşılık gelebilir) o "deck" içindeki bütün kartlara erişebilir. Bu yönüyle "deck of cards" tasarımı, band genişliği kısıtlamalarına uygun bir yaklaşımdır (Mann, 1999).

WML elemanları 2 ana grup altında incelenebilir:

- "Deck /Card" Elemanları: *wml, card, template, head, access, meta*
- "Event" Elemanları: *do, ontimer, onenterforward, onenterbackward, onpick, onevent,*

postfield

Event elemanları da kendi içlerinde aşağıdaki gibi gruplanabilir:

Tasks: *go, prev, refresh, noop*

Variables: *setvar*

User Inputs: *input, select, option, optgroup, fieldset*

anchors, images, and timers: *a, anchor, img, timer*

Text Formatting: *br, p, table, tr, td*

Bu raporda yukarıdaki elemanları birer birer incelemek yerine yeri geldiğinde özelliklerinden bahsedilecektir.

Bütün elemanlar yapısal olarak aşağıdaki sözdizimi (*syntax*) yapısındadır:

```
<element ismi element attributes >
  content
</element ismi>
```

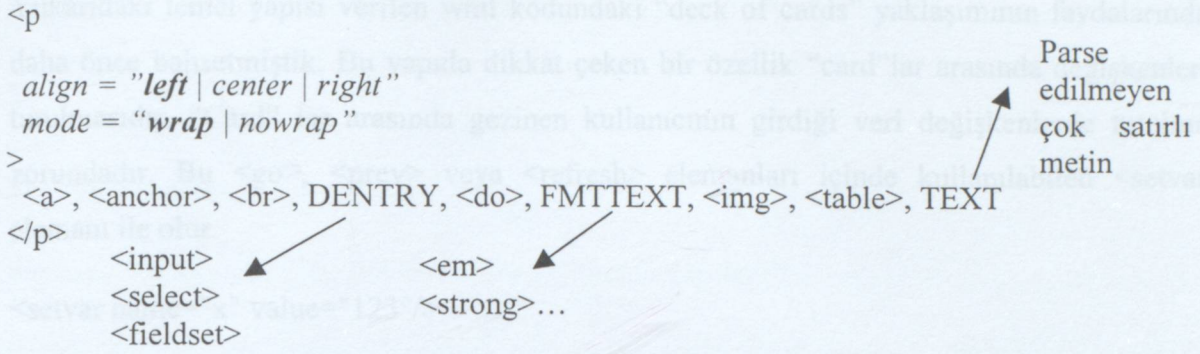
Bütün elemanların servis sağlayıcı tarafından derlenmek üzere 3 tane özellikleri vardır. Bunlar *xml:lang, id, class* özellikleridir.

WML bağlam yönetimine (*context management*) olanak verir. O andaki bağlam (*context*) en azından mevcut bütün değişkenler ve değerleri, ve belirli bir geçmişe kadar olan URL değerleri gibi değerleri tutar. Yapı daha gelişmiş bağlam yönetimi algoritmalarına açıktır. Şunu da belirtmek gerekir ki, herhangi bir wap sitesine erişimde mutlak (*absolute*) URL adresi kullanılır.

Diğer bazı kurallar şöyledir (Mann, 1999):

- WML XML karakter setini kullanır, bu da UNICODE 2.0 karakter setidir.
- Açıklamalar (*comment*) ve dokümantasyonlar `<!-- comment -->` şeklindedir.
- Elemanlar, özellikler ve değerleri için küçük harfler kullanılır.
- Kart içindeki bütün içerik `<p>` paragraf elemanının içerisinde yer alır.

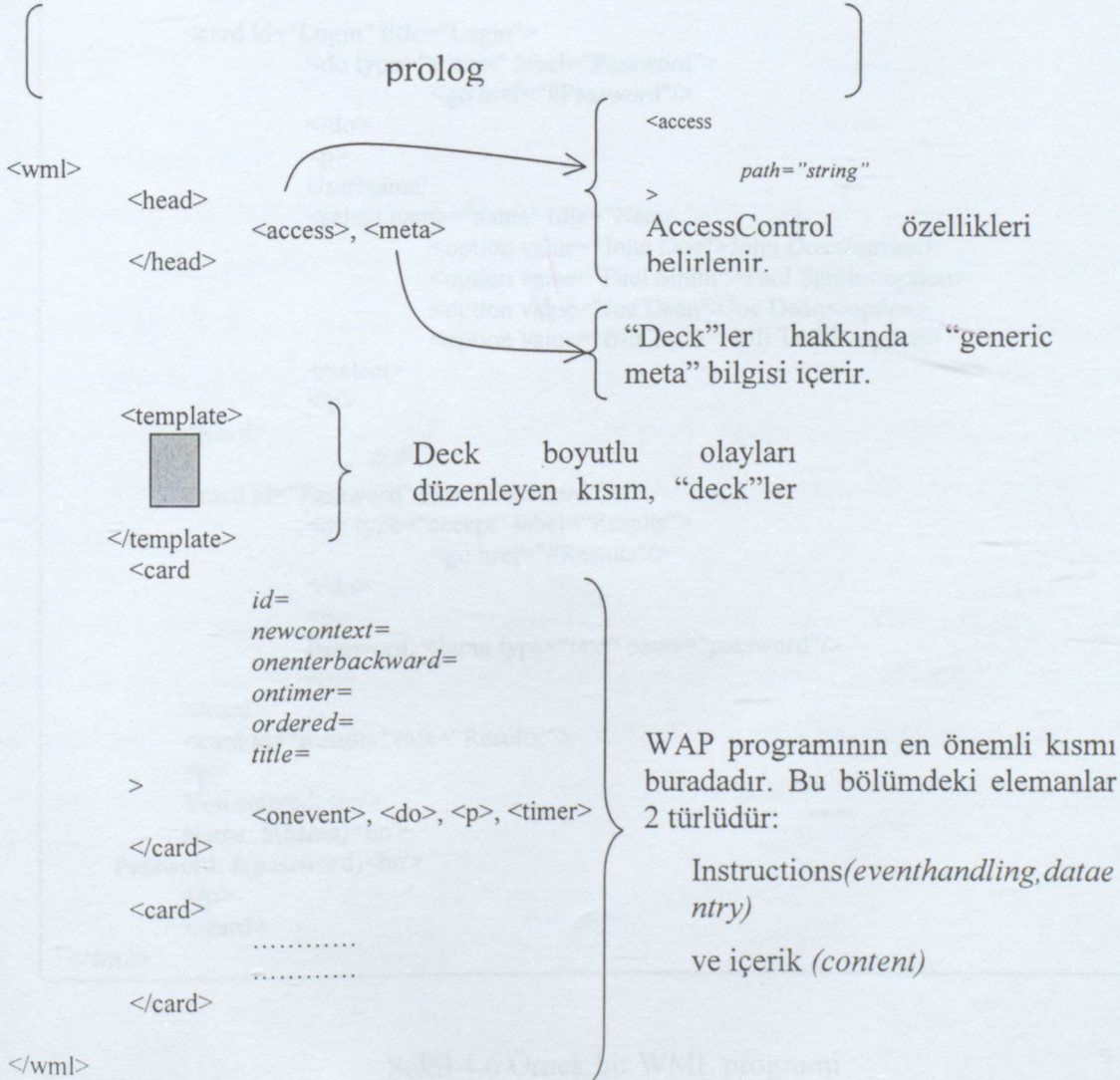
Şimdi örnek olarak `<p>` elemanın yapısını aşağıdaki gibi şematize edebiliriz: (Şekil 4.4)



Şekil 4.4 <p> elemanının yapısı

Yukarıda, elemanın yapısı (içerisinde bulundurabileceği diğer elemanlar) ve alabileceği özellikler listesi gösterilmiştir. Ayrıca her özelliğin ilk değeri **bold** olarak yazılmıştır. Şimdi diğer bütün elemanların bu özellikleri DTD dosyasına bakarak bulunabilir.

Şekil 4.5’de tipik bir wml kodunun ana hatları açıklanmaya çalışılmıştır.



Şekil 4.5 Basit bir WML program yapısı

Yukarıdaki temel yapısı verilen wml kodundaki “deck of cards” yaklaşımının faydalarından daha önce bahsetmiştik. Bu yapıda dikkat çeken bir özellik “card”lar arasında değişkenlerin tutulmasıdır. “Card” lar arasında gezinen kullanıcının girdiği veri değişkenlerde tutulmak zorundadır. Bu <go>, <prev> veya <refresh> elemanları içinde kullanılabilen <setvar> elemanı ile olur.

```
<setvar name="x" value="123"/>
```

Mesela yukarıdaki eleman ile x değişkeni tanımlanmış ve 123 değerini almıştır.

Örnek bir WML Programı [3]:

```
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

  <card id="Login" title="Login">
    <do type="accept" label="Password">
      <go href="#Password"/>
    </do>
    <p>
      UserName:
      <select name="name" title="Name:">
        <option value="John Doe">John Doe</option>
        <option value="Paul Smith">Paul Smith</option>
        <option value="Joe Dean">Joe Dean</option>
        <option value="Bill Todd">Bill Todd</option>
      </select>
    </p>
  </card>

  <card id="Password" title="Password:">
    <do type="accept" label="Results">
      <go href="#Results"/>
    </do>
    <p>
      Password: <input type="text" name="password"/>
    </p>
  </card>

  <card id="Results" title="Results:">
    <p>
      You entered:<br/>
      Name: $(name)<br/>
      Password: $(password)<br/>
    </p>
  </card>

</wml>
```

Şekil 4.6 Örnek bir WML programı

Yukarıdaki örnek programda (Şekil 4.6) kullanıcı bir listeden kullanıcı ismini (*username*)

seiyor, daha sonra bir parola (*password*) giriyor, son olarak program bize yapılan *username* ve *password* seimlerini gsteriyor. Yapılan bu 3 iřlem iin 3 ayrı “card” tanımlanmış: Login, Password, ve Results. İlk iki kard da

```
<do type="accept" ... >
</do>
```

yapısıyla kullanıcıda giriş bekleyen “event” elemanları ierir.

HTML’de “anchor” yapısıyla özdeş olarak ařağıdaki <go> elemanı, bulunulan “card”dan “Results” “card”ına ulaşmayı sağlar.

```
<go href="#Results"/>
```

Son olarak Results kartında <input> elemanı ile belirlenen deęişkenlere (*name* ve *password*) ulaşım ařağıdaki söz dizimi ile gereklenmiştir. Burda <setvar> elemanına ihtiyaç olmamıştır.

```
$(name)
$(password)
```

4.3.3 WMLScript

WMLScript WML diline programlama lojięi getirmek maksadıyla tasarlanmış Javascript’in WAP’teki karřılıęıdır. WMLScript özel olarak ECMAScript diline dayandırılmıştır. ECMAScript üzerine yapılan deęişiklikler band genişlięi ve hafıza kısıtlamaları göz önüne alarak yapılmıştır. WMLScript, WML gibi WML Arayüzde “bytecode” olarak derlenerek WAP uyumlu mobil cihaza iletilir. WMLScript sadece kullanıcı tarafındaki işlemlere (*client-side transaction*) olanak verir. Hedef band genişliğinden kazanç ve servis sağlayıcı yükünü hafifletmektir (Mann, 1999).

Kullanıcı tarafındaki işlemlere (*client-side transaction*) ek olarak, servis sağlayıcı tarafındaki hareketlere (*Server-side transactions*) olanak veren ve gerek manada Internet’te verilen ok deęişik servisleri (Internet üzerinden mesaj aktarımı, enterprise bilgi yığınlarına erişim, ve elektronik ticaret gibi) almaya olanak veren WML programları, WAP ve bunlar gibi protokoller kullanılarak gereklenebilir.

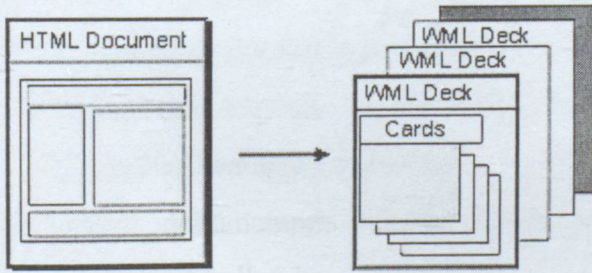
5. HTML/WML GEÇİŞİNİ SAĞLAYAN SİSTEMİN TASARIMI

Internet'teki bilgilerin ve kaynakların kablosuz ortama taşınması düşüncesiyle ortaya atılan standart ve protokoller WAP teknolojileri adı altında daha önceki bölümlerde incelendi.

Bir HTML dokümanını, WAP ortamında kullanılmak üzere WML dokümanına çevirmek basit bir etiket (*tag*) dönüşümü değildir. Bu bölümde böyle bir dönüşümde karşılaşılan temel sorunlar, bu sorunlara getirilen çözümler ve bu dönüşümü yapan bir sistemin tasarımı açıklanacaktır.

Basit bir etiket dönüşümü gibi görünen bu problemde temel sorun HTML gibi düzenli olmayan bir formattan bilginin mümkün olduğunca kayıpsız çıkarılması ve WML gibi yeni bir formata uygun hale getirilmesidir. HTML düzenli bir formatta olmamasının yanı sıra etiket sayısı WML'in etiket sayısının yaklaşık 3 katıdır. Bu da gösteriyor ki birebir bir dönüşüm imkanı yoktur. Demek ki ortaya atılan sistemlerde hem bazı önemsiz etiketler (belki içerisindeki bilgi ile beraber) atlanacak, hem de bazı etiketler birden çok etikete eşlenecektir [8].

Diğer bir konuda HTML dokümanının büyüklüğüdür. Önceki bölümlerde belirtildiği gibi WAP kullanıcısı kapasitesi her yönden kısıtlıdır. Bu yüzden büyük HTML dosyaları belki birden çok küçük WML dosyalarına dönüştürülmesi kaçınılmaz olacaktır.(Şekil 5.1) Bu tasarlanan sistemin karmaşıklığını önemli oranda artıracaktır (Kaasinen vd., 2000).



Şekil 5.1 HTML dosyasından birden çok WML dosyası oluşturmak

Diğer bir problem bazı önemli yapısal etiketlerin WML'de olmamasıdır. Örneğin, *frameset* ve *frame* etiketi yapısal olarak önemli bir etikettir. Fakat bu etiketin yerini tutan bir yapı WML'de yoktur. Bu noktalarda genel yaklaşım her karşılaşılan *frame* için yeni bir *card* tanımlayarak ayrı bir WML dokümanına bağlantı (*link*) tanımlamaktır. Bunun yanında *h1*, *h2*, ... (*headings*), *img* (*images*), *script* (*scripts*), *table*, *tr*, *td* (*tables*) gibi yapısal ve önemli veri içeren bazı etiketlerde karşılaşılan problemler, tasarımın diğer aşamalarında ele

alınacaktır [8].

Bu temel sorunlar böyle bir dönüşümde karşılaşılan problemlerden bazılarıdır. Aşağıda HTML dili ile WML dilinin arasındaki farklar daha ayrıntılı olarak ele alınacaktır.

5.1 HTML ile WML Dili Arasındaki Farklar

Daha önceki bölümlerde belirtildiği gibi WML, kablosuz uç birimler için (cep telefonu, PDA) tasarlanmış XML tabanlı bir biçimlendirme dilidir. WML biçimlendirme dili 30 etiketli, hatasız sözdizimli bir HTML olarak düşünülebilir. Fakat buradaki etiketler HTML'de kullanılmayan, kablosuz ortam için tanımlanmış yeni etiketlerdir.

Öte yandan HTML biçimlendirme dilinde 120 kadar etiket, sözdizimi kurallarına uymadan rahat bir yapıda kullanılır. HTML'in bu şekilde kullanılmasının Internet ve WWW ortamlarında avantajı çoktur. Uç birimlerdeki sayfa tarayıcısı HTML dokümanlarındaki bilgiyi mümkün olduğunca gösterir. XML teknolojisi HTML'in bu düzensiz yapısına belirli bir düzen getirmektedir.

HTML'in kablosuz ortamda kullanılamamasının temel nedenleri şunlardır [8]:

- HTML'in mevcut etiketleri ile sınırlı derinlikte iç içe geçmiş karmaşık yapılar oluşturmak mümkündür. Bu yapıların düşük kapasiteli PDAlar ile işlenmesi ve görüntülenmesi olanaksızdır.
- HTML sözdizimi hatasız değildir. PDAlar ancak az karmaşıklıkla hatasız sözdizimli dokümanları görüntüleme kapasitesine sahiptirler.
- HTML özel Unicode, ASCII, ISO2002, UTF karakterleri görüntüleyebilir. WML'de UTF 8 karakter kodlaması kullanılır.
- HTML dokümanında etiketler küçük yada büyük karakter olabilir. WML'de her zaman küçük harf kullanılır.
- WML sadece belirli kurallar çerçevesinde iç içe girmiş yapıları görüntüleyebilir.
- PDA uç birimlerinin görüntü boyutu oldukça küçüktür.

HTML'de yüksek düzeyde olabilen iç içe yapıların WML'e belirli oranda aktarılması sistemin karmaşıklığını belirli oranda artırır. Bunun yanında girişin yapısal olarak eksik olması ve/veya sözdizimi kurallarına uymaması da sistemin karmaşıklığını artırır. Bu nedendir ki bu sistemlerde ortak yaklaşım gelen HTML dosyasını ilk olarak XHTML'e dönüştürmektir. Böylece sisteme giren dosya XML standartlarında sözdizimi kurallarına uygun olacak ve düzenli bir doküman alan sistem karmaşıklık yönünden iyileşecektir. Karmaşıklığı arttıran ilk

nedeni ele alırsak; iç içe yapılar sistemin bu yapıları çözebilme yeteneğine bağlı olarak belirli oranda veri kaybı ile çözülebilirler.

SGML bütün biçimlendirme dillerinin çıkış noktasıdır. HTML bir SGML uygulamasıdır. XML de SGML'in bir alt kümesidir. WML ve XHTML, bir XML uygulaması olarak düşünülebilir [8]. Bu sınıflandırmadan da anlaşılacağı gibi iki XML uygulamaları arasındaki dönüşüm sistem için en uygun olan dönüşümdür.

Aşağıdaki tablo (Çizelge 5.1) HTML ile WML etiketleri ve bu etiketlere ait olan özellikler ve bu etiketlerin hangi etiketleri içerebileceğini göstermektedir. Tablodaki etiketler hem HTML hem de WML'de olan etiketlerdir. Bu tabloda bulunmayan HTML de olup WML'de olmayan çok sayıda etiket mevcuttur. Aynı şekilde WML'de olup da XHTML'de olmayan etiketler de Çizelge 5.1'de gösterilmemiştir.

Çizelge 5.1 Bazı HTML ve WML etiketleri ve özellikleri

ETİKET	HTML	XHTML	WML
	Özellikler	Özellikler	Özellikler
a	class, id, style, title, charset, type, name, href, hreflang, rev, rel, accesskey, shape, coords, tabindex, onfocus, onblur	accesskey, charset, href, hreflang, rel, rev, tabindex, title, type	href, title, id, class, xml:lang
b	class, id, style, title		id, class, xml:lang
br	class, id, style, title,	clear	id, class, xml:lang
em	class, id, style, title		id, class, xml:lang
head			id, class
i	class, id, style, title		id, class, xml:lang
img	class, id, style, title, src, alt, longdesc, height, width, usemap, ismap	alt, height, longdesc, src, style, title, width	alt, src, localsrc, vspace, hspace, align, height, width, id, class, xml:lang
input	class, id, style, title, type, name, value, checked, disabled, readonly, size, maxlength, src, alt, usemap, tabindex, accesskey, onfocus, onblur, onselect, onchange, accept	accesskey, checked, maxlength, name, wml:name, size, src, style, tabindex, title, type, value	name, type, value, format, emptyok, size, maxlength, tabindex, title, id, class, xml:lang
p	class, id, style, title	align, mode, wml:mode	align, mode, id, class, xml:lang
select	class, id, style, title, name,	Style, title	title, name, value,

	size, multiple, disabled, tabindex, onfocus, onblur, onchange		iname, ivalue, multiple, tabindex, id, class, xml:lang
strong	class, id, style, title		id, class, xml:lang
table	class, id, style, title, summary, width, border, frame, rules, cellspacing, cellpadding	align, style, summary, title, width	title, align, columns, id, class, xml:lang
td	class, id, style, title, abbr, axis, headers, scope, rowspan, colspan	abbr, align, axis, colspan, headers, rowspan, scope, style, title, valign	id, class, xml:lang
tr	class, id, style, title	style, title, valign	id, class
u	class, id, style, title		id, class, xml:lang

Tabloda XHTML biçimlendirme dilinin özellikleri ile WML'in özellikleri sunulmuştur. Tablodan da görüleceği gibi HTML'in mevcut kullanılabilecek özelliklerinin sayısı diğer biçimlendirme dillerindekine göre daha fazladır. XHTML ve WML, XML uyumlu oldukları için yapıları tabloda görülen özellikler dışında başka özellik kullanmazlar. XHTML ile WML özelliklerine baktığımızda bunlar arasında bir dönüşümde ancak zorunlu kullanımı ortak olan özellikler dönüşümde aktarılır. Dönüşüm ile ilgili tablolar daha sonra tekrar ele alınacaktır.

Kullanılan karakter kodları (*character encodings*) HTML ve WML'de farklıdır. Bu farklar belirtilmeden evvel karakter kodlarının ne olduğu aşağıda anlatılmıştır.

5.1.1 Karakter Kodları

Belirli bir sırada karakter listesine "karakter seti" denir. Bir karakter setindeki karakterlerin bitler ile kodlanması karakter kodlamasıdır. Kod çözme ise bu bit dizilerinin okunabilir karakterlere dönüştürülmesidir. Aşağıdaki tabloda (Çizelge 5.2) örnek olarak bazı karakter setleri ve o sete ait karakter kodlama isimleri verilmiştir.

Çizelge 5.2 Bazı karakter setleri ve ilgili kodlama standardı

KARAKTER SETİ	KODLAMA
UNICODE (Çok Dilli)	UTF-8, UTF-16, MIME, UTF-7
EACC (Çok Dilli)	4-byte EACC, 3-byte EACC
Big5 (Çince)	Big5, MIME

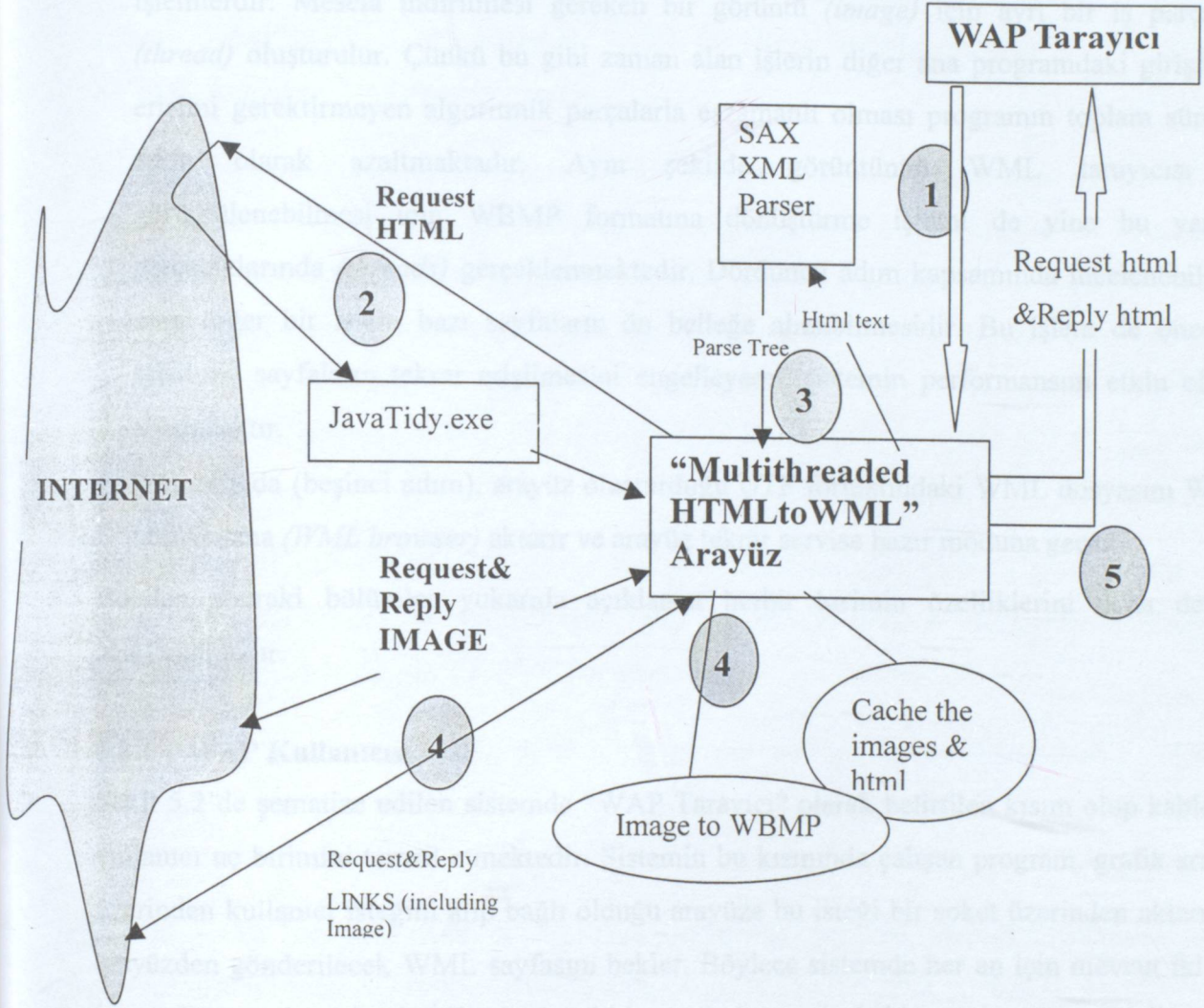
Örneğin 16 bit kodlamalı bir Unicode [UCS-2 ISO 10646] seti $2^{16} = 65536$ farklı karakteri kodlama kapasitesine sahiptir. Bu yüzden Unicode standardı dünyada kullanılan en yaygın, büyük dilleri de içine alan bir karakter setidir. Uluslararası Karakter Seti (*Universal Character Set*) betik dilleri, programlama dilleri ve çeşitli platformlar üzerinde herhangi bir dilden her bir karakter için ona ait bir kod tanımlar. Unicode standardı 7 bit karakter uzunluklu ASCII üzerine bina edilmiştir. 7 bitli ASCII'nin uluslararası bir standart için yetersiz kalması üzerine Unicode karakter seti yaygın olarak kullanılmaktadır [9].

Unicode standardı Apple, HP, IBM, Microsoft, Oracle, SAP, Sun, Sybase, Unisys ve diğer büyük endüstri liderleri tarafından benimsenen bir standarttır. XML, JAVA, ECMAScript (JavaScript), LDAP, WML gibi son yıllarda ortaya atılan teknolojiler ve yine yaygın olarak kullanılan İşletim Sistemleri ve tarayıcılar Unicode standardını kullanmaktadır.

UTF (*Unicode Transformation Formats*) kodları mevcut ASCII kodlarını koruyarak $2^8 = 256$ dan daha çok olan karakter setlerindeki karakterleri kodlar [9].

WML bir XML uygulaması olduğu içindir ki WML UTF-8 kodlamasını kullanır. Öte yandan HTML UTF kodlarını gösterebildiği gibi Latin1 (&#__; şeklinde ifade edilen karakterler), ASCII, ISO2002 gibi diğer kodlamaları da gösterir. Bu yüzden HTML/WML dönüşüm sisteminde HTML'den gelen bu özel karakterler uygun olan en yakın karaktere dönüştürülmelidir. Bu gibi özel karakterlerin dönüşümleri sistemde ayrı bir dosyada tanımlanmıştır.

5.2 Sistemin Tasarımı ve Temel Yapı Birimleri



Şekil 5.2 HTMLtoWML sisteminin temel yapı birimleri

Yukarıdaki şekilde (Şekil 5.2) bu tezde tasarlanan sistemin temel yapı birimleri gösterilmiştir.

- Birinci adımda, WAP Kullanıcısı (*WAP Client*) istenilen URL'i servise hazır olarak bekleyen Arayüze gönderir.
- İkinci adımda Arayüz (Şekil 5.2'de "*Multithreaded HTMLtoWML*" olarak adlandırılan birim) istenilen sayfanın URL'ini aldıktan sonra kurduğu socket bağlantısı ile Internetteki sayfanın HTML kodunu alır. Fakat bu kodu alırken HTML kodu bir temizlenme aşamasından geçer. Sonuçta istenen URL'den gelen sayfa XHTML dokümanı olup, ayrıştırılmaya uygun hale getirilmiştir.

- Üçüncü adımda ise XHTML sayfası kelimelerine ayrıştırılmak üzere SAX HTML ayrıştırıcıya (Şekil 5.2’de *SAX HTML Parser* olarak adlandırılan birim) gönderilir.
- Dördüncü adımda ifade edilen işlemler arayüzde yapılan dönüşüm sırasında eşzamanlı olan işlemlerdir. Mesela indirilmesi gereken bir görüntü (*image*) için ayrı bir iş parçacığı (*thread*) oluşturulur. Çünkü bu gibi zaman alan işlerin diğer ana programdaki giriş/çıkış erişimi gerektirmeyen algoritmik parçalarla eşzamanlı olması programın toplam süresini etkin olarak azaltmaktadır. Aynı şekilde görüntünün WML tarayıcısı ile görüntülenebilmesi için WBMP formatına dönüştürme işlemi de yine bu yan iş parçacıklarında (*threads*) gerçekleştirilmektedir. Dördüncü adım kapsamında incelenebilecek olan diğer bir işlem bazı sayfaların ön belleğe alınabilmesidir. Bu işlem de önceden erişilmiş sayfaların tekrar erişilmesini engelleyerek sistemin performansını etkin olarak arttıracaktır.
- Son adımda (beşinci adım), arayüz oluşturduğu UTF formatındaki WML dosyasını WML tarayıcısına (*WML browser*) aktarır ve arayüz tekrar servise hazır moduna geçer.

Bundan sonraki bölümler yukarıda açıklanan her bir birimin özelliklerini daha detaylı açıklamaktadır.

5.2.1 WAP Kullanıcısı

Şekil 5.2’de şematize edilen sistemde “WAP Tarayıcı” olarak belirtilen kısım olup kablosuz kullanıcı uç birimini temsil etmektedir. Sistemin bu kısmında çalışan program, grafik arayüz üzerinden kullanıcı isteğini alıp bağlı olduğu arayüze bu isteği bir soket üzerinden aktarır ve arayüzden gönderilecek WML sayfasını bekler. Böylece sistemde her an için mevcut iki ana iş parçacığı çalışmaktadır. Bunlardan biri arayüzde servis bekleyen iş parçacığı diğeri de WAP tarayıcının çalıştığı iş parçacığıdır. WAP Tarayıcı WML formatında bilgiyi aldıktan sonra bunu görüntüler. Sistemin bu kısmı JAVA 1.1 programlama dili ile yazılmış bir *applet* uygulamasıdır (Seçilir, 2000). Tarayıcı sadece metin bilgisi görüntülemektedir. Bu bir miktar veri kaybına neden olabilmektedir. Fakat bir kablosuz uç birim ekranın boyutları düşünüldüğünde HTML dosyalarındaki büyük görüntülerin filitrelenmesi oldukça doğru bir yaklaşımdır. Öte yandan link içeren küçük görüntülerin (ufak butonlar gibi) filitrelenmesi daha ciddi bir veri kaybına neden olmaktadır.

Grafiksel ortamda kullanıcı ile etkileşimde kullanıcıdan URL isteği almanın yanı sıra bir önceki sayfaya gitme, o andaki sayfada aşağı/yukarı hareket edebilme, ve bulunulan sayfadaki linklere erişebilme gibi özellikler vardır. Kullanıcının seçtiği linkin URL adresi daha önce

olduğu gibi arayüze aktarılır ve karşılık gelen WML sayfası beklenir.

5.2.2 HTML/XHTML Dönüşümü (TIDY ve Java TIDY)

Arayüze gelen HTML isteği ağa aktarılır ve istenilen HTML sayfasının kodu alınır. HTML kodunu doğrudan WML dönüştürmek yerine ilk olarak kurallara uygun bir yapıda olan XHTML formatına dönüştürmek gerekir.

HTML biçimlendirme dilini hatasız kullanmak neredeyse olanaksızdır. Bilgilerin sunulmasında biçimlendirme dili kullanmak gereklidir ve aynı zamanda bu dili kullanırken her kuralı takip etmek oldukça zordur. Bunun için HTML sayfalarını otomatik olarak düzenleyen hataları gösteren veya temizleyen bir programa ihtiyaç duyulmuştur. Böyle bir program ile HTML dosyası temizlenmiş HTML'e, diğer bir ifade ile XML'e dönüştürülmüş olarak XHTML dokümanı adını alır. İşte bu dönüşümü yapan program HTML Tidy isimli programdır.

HTML Tidy ilk olarak Dave Raggett tarafından geliştirilmiş olup şu anda W3C (*world wide web consortium*) konsorsiyumunda bir grup tarafından kontrol edilmektedir. Program çalıştırma dosyası ve kaynak kodu [12]'den elde edilebilir.

HTML Tidy programı HTML dokümanını analiz ettikten sonra düzeltebildiği hataları düzeltir, uyarı olarak çıkışa aktarır; düzeltemediklerini olduğu gibi bırakarak hata olarak çıkışa aktarır.

Çizelge 5.3'de bazı örneklerde HTML Tidy'nin bazı HTML kod parçacıklarını temizlemesi gösterilmiştir (Morrison, 2000): (tabloda Tidy çıktısı XHTML olarak gösterilmiştir)

Çizelge 5.3 HTML Tidy örnekleri

HTML	XHTML
<h1> bilgi <h2> başlık </h3>	bilgi başlık
<p> paragraf kalın<i> yan yazı kalın </i>	paragraf kalın <i> yan yazı </i>kalın
bilgi<i> yan yazı	bilgi<i> yan yazı</i>
<i>yan yazı<h1> bilgi </h1></i> <p> paragraf kalın <p> kalın	<h1> başlık<i> yan yazı </i></h1> <p> paragraf kalın <p> kalın
bilgi<hr> bilgi bilgi<hr>bilgi	<hr> bilgi bilgi <hr>

<code> link <a></code>	<code><h1> bilgi </h1></code>
<code><body></code>	<code> link </code>
<code> ilk eleman</code>	<code><body></code>
<code> ikinci eleman</code>	<code></code>
	<code> ilk eleman </code>
	<code> ikinci eleman</code>
	<code></code>

HTMLTidy programı kullanıcı isteğine göre konfigure edilebilir. XHTML çıktısının yapısı (*layout*) kullanıcı isteğine göre ayarlanabilir. Mesela “auto-indenting” ile daha kolay okunabilir XHTML dokümanları elde edilebilir.

HTML Tidy karakter kodlaması olarak US ASCII, ISO Latin 1, UTF-8 ve ISO2022 kodlarını kullanır. Konfigurasyon ile kullanıcı gerekli kodlamayı belirler. HTML Tidy çalışırken *config.txt* isimli dosyadan gerekli konfigurasyon bilgilerini okuyarak XHTML dokümanını oluşturur.

5.2.2.1 Java TIDY

Java TIDY (JTidy), HTML Tidy’nin JAVA’da geliştirilmiş halidir. JTidy, HTML Tidy gibi hatalı ve düzensiz HTML dokümanını düzenleme programıdır. JTidy’nin Tidy özelliği yanı sıra, DOM tipi ayrıştırıcı özelliği de vardır [12].

Bu tezde tasarlanan sistemde JTidy, sadece Tidy özelliği ile kullanılmıştır. Erişilmek istenen URL bilgisi, WAP kullanıcılarından alındıktan sonra sistem Internet’ten istenen dokümanı XHTML’e dönüşmüş olarak alır [12].

5.2.3 Ayırıştırma

XML dokümanının ayırıştırma ağacının düzgün olması ve bu yüzden başka bir ağaca dönüştürmede kullanılabilmesi XML’in belirli kurallar içerisinde kalmasındandır. XML dilinin ortaya atılmasının temel nedenlerinden birisi de Internet’in ve WWW’in diğer ortamlarla entegrasyonunun sağlanabilmesidir. XML’in diğer bir kullanımı büyük çapta verilerin belirli bir düzende saklanabilmesi, verinin sınıflandırılmasının kolaylaşması, bunun sonucu olarak erişimin hızlı olabilmesi gibi çok yönlü avantajları vardır.

Sistemimizin, bu aşamasında XHTML bir XML dokümanı olarak oluşturulmuştur. Elimizdeki bu XML dokümanının kelimelerine ayrıştırılması bu bölümde incelenecektir.

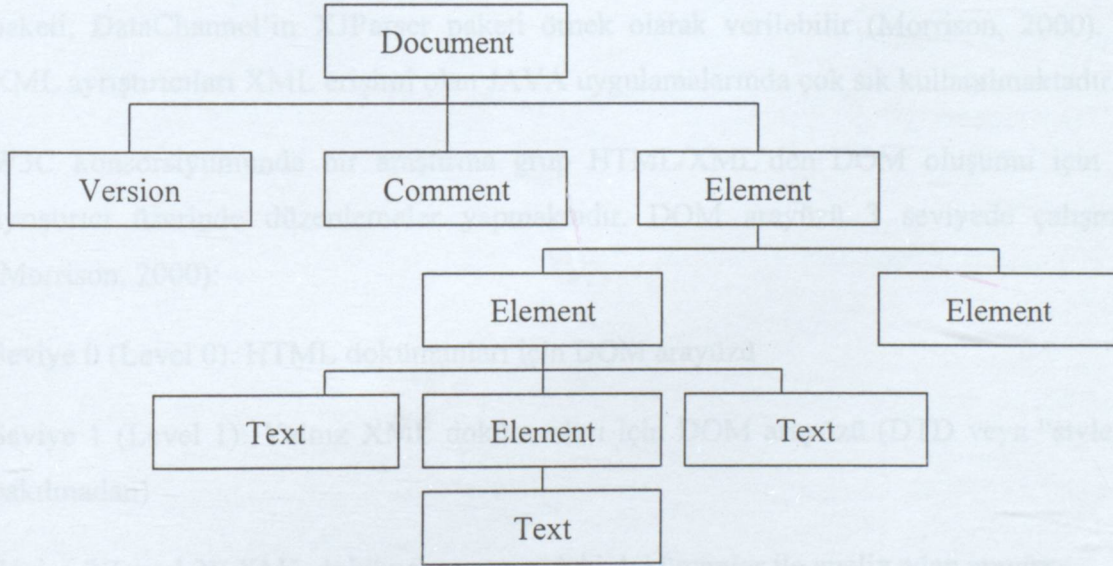
Ayrıştırma ağacı (*parse tree*) bir veri saklama yapısıdır. Bir dosyada belirli bir formatta

bulunan bilgi, birçok temel veri nesnelerini içinde taşır. Bu nesnelerin ve her nesnenin diğer nesnelerle ilişkisini bir ağaç üzerinde kolaylıkla gösterebiliriz.

Madem XHTML bir XML dokümanı ve XML de belirli bir biçimde saklanıyor, o zaman bir XHTML dokümanı bir ayrıştırma ağacında saklanabilir. Bu ağaç üzerinde gerekli işlemler kolaylıkla, hızlı bir şekilde yapılabilir.

Şekil 5.3'deki örnekte basit bir XML dokümanı ve hemen altında ilgili ayrıştırma ağacı gösterilmiştir (Morrison, 2000):

```
<?xml version= "1.0"?>
<!--ayrıştırma ağacı -->
<xdoc>
  <selam> merhaba <kalın> ayrıştırma </kalın> ağacı </selam>
  <denge type= "yok"/>
</xdoc>
```



Şekil 5.3 Örnek bir ayrıştırma ağacı

Şekil 5.3'deki ayrıştırma ağacı incelendiğinde her nesnenin birbiri ile alakası kolaylıkla görülmektedir. Yukarıdaki ağaçta 10 tane düğüm vardır. Mesela "Document" nesnesi dışında diğer bütün nesnelerin ebeveyni (*parent*) vardır. Ağaçtaki kutularda nesnenin hangi tip bir nesne olduğu yazılmıştır. Ağacın kökü "Document" nesnesidir. Ve bu nesnenin 3 tane çocuğu vardır. Bunlardan ilki XML'in versiyonunu belirten `<?xml version= "1.0"?>`, ikincisi not olarak düşülen `<!--ayrıştırma ağacı -->`, üçüncüsü ise `<xdoc>` nesnesidir. Ağaçta `<xdoc>` nesnesi birinci seviyede element olarak görülmektedir. Ağacın oluşumu bu şekilde analiz yapılarak incelenebilir.

Bir ayrıştırma ağacı XML dokümanını yukarıdaki gibi ayrıştırıp her düğümdeki nesneye ait özellikleri o düğüm yapısında tutmaktadır. Her düğümün tipi, ismi, değeri, ebeveynin ismi, çocuk düğümlere sahip olup olmadığı, çocuklarının listesi gibi özellikler düğümün içerisinde tutulur.

JAVA ortamında mevcut XML ayrıştırma programlarına genel olarak 2 tür arayüz vardır. Bunlar SAX (*Simple API for XML*) ve DOM (*Document Object Model*)dur (Morrison, 2000). SAX kullanımı daha az kaynak gerektiren ve olaya dayalı(*event-driven*) bir arayüzdür. DOM ise XML dokümanını bir ayrıştırma ağacında saklar. Bu ayrıştırma ağacı ile veriye erişim çok hızlı hale gelir. Böyle bir erişim olmadığı zaman XML sadece veriyi biçimlendirme ve saklama için kullanılmış olur.

Bazı önemli SAX ayrıştırıcılar şunlardır: JavaProject X, IBM XML4J (Morrison, 2000). DOM ayrıştırma ağacını oluşturan paketlere ise IBM'in XML4Jv2 paketi, Oracle'ın XML paketi, DataChannel'in XJParser paketi örnek olarak verilebilir (Morrison, 2000). Bu tip XML ayrıştırıcıları XML erişimi olan JAVA uygulamalarında çok sık kullanılmaktadır.

W3C konsorsiyumunda bir araştırma grup HTML/XML'den DOM oluşumu için gerekli ayrıştırıcı üzerinde düzenlemeler yapmaktadır. DOM arayüzü 3 seviyede çalışmaktadır (Morrison, 2000):

Seviye 0 (Level 0): HTML dokümanları için DOM arayüzü

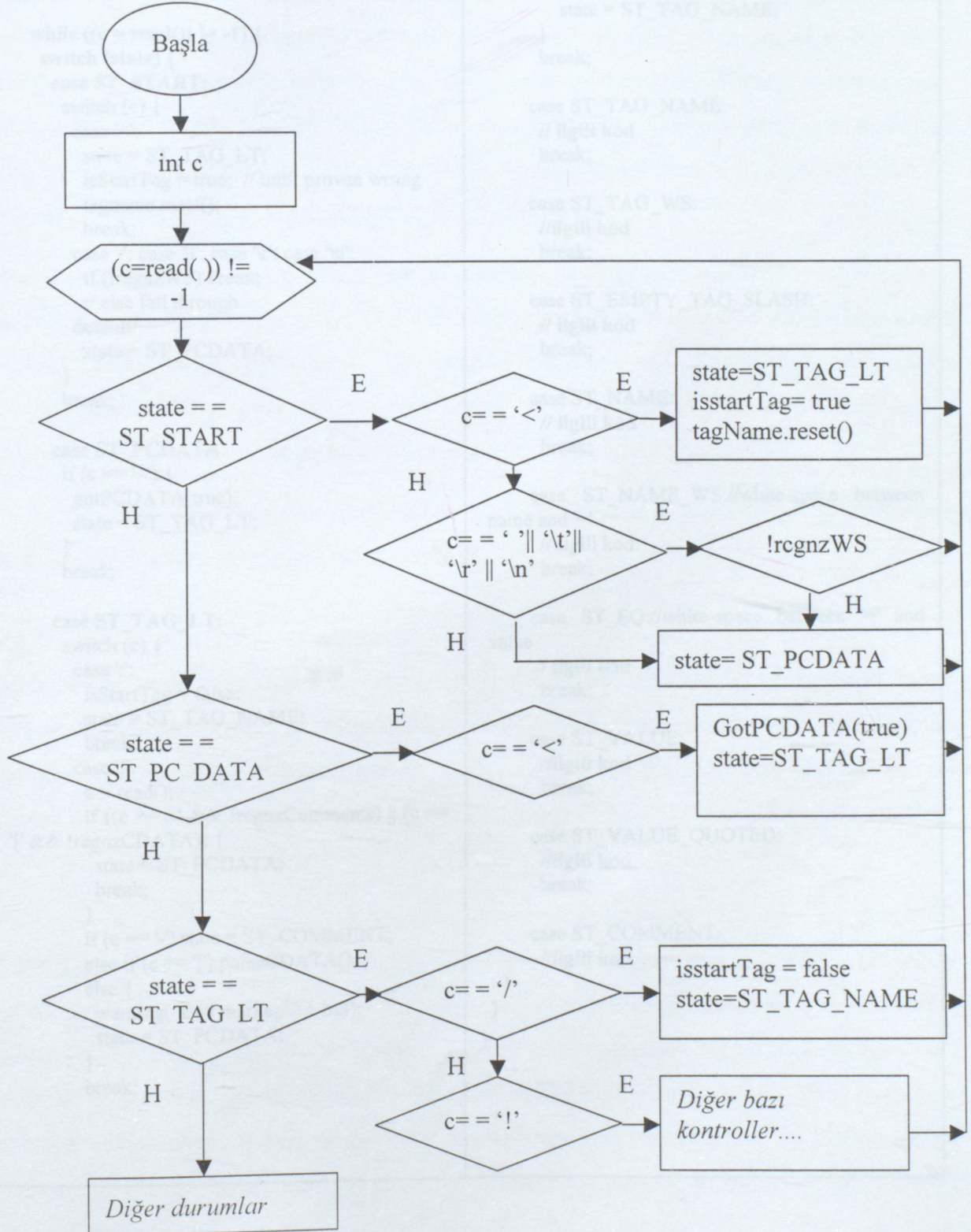
Seviye 1 (Level 1): Yalnız XML dokümanları için DOM arayüzü (DTD veya "stylesheet"e bakılmadan)

Seviye 2(Level 2): XML dokümanını yanındaki dokümanlar ile analiz eden arayüz

DOM ayrıştırıcı, HTML/XML dokümanına yukarıdaki seviyelerle erişimi çeşitli arayüzler ile yapar. Bu arayüzlerdeki fonksiyonlar ile veri üzerinde işlemler gerçekleştirilir. En çok kullanılan arayüzler *DOMImplementation*, *DocumentFragment*, *Document*, *Node*, *NodeList*, *Element*, *Text*, *Attr*'dir.

DOM, ağaç yapısında bir nesne modeli sunarken SAX herhangi bir veri yapısında ayrıştırma yapmaktadır. Bu veri yapısı ağaç olabilir veya herhangi başka bir yapıda ayrıştırma bilgisi saklanabilir. O yüzden SAX çok daha geniş uygulamalara olanak verir. Aynı zamanda SAX'ın veriye daha düşük seviyeli erişimi, erişimin daha hızlı olmasına olanak verir ve performansı artırır (Morrison, 2000).

Bu tez çalışmasında tasarlanan sistemde HTML Tidy çıkışında elde edilen XHTML dokümanı ayrıştırılması bir tür SAX ayrıştırıcı ile gerçekleştirilir. Şekil 5.4'deki akış diyagramı ve hemen arkasından gelen ilgili kod parçacığı sistemin ayrıştırma kısmının bir parçasıdır. Kod parçacığında görüldüğü gibi *Tokenizer* isimli nesne SAX arayüzünü kullanan bir ayrıştırıcıdır.




```

public class Tokenizer implements
org.xml.sax.Parser {
.....
// değişkenlerin tanımlanması ve ilk değerlerin
atanması
.....
public void tokenize() throws Exception {
    int c;

    while ((c = read()) != -1) {
        switch (state) {
            case ST_START:
                switch (c) {
                    case '<':
                        state = ST_TAG_LT;
                        isStartTag = true; // until proven wrong
                        tagname.reset();
                        break;
                    case ' ': case '\t': case '\r': case '\n':
                        if (!rcgnzWS) break;
                        // else fall through
                    default:
                        state = ST_PCDATA;
                }
                break;

            case ST_PCDATA:
                if (c == '<') {
                    gotPCDATA(true);
                    state = ST_TAG_LT;
                }
                break;

            case ST_TAG_LT:
                switch (c) {
                    case '/':
                        isStartTag = false;
                        state = ST_TAG_NAME;
                        break;
                    case '!':
                        c = read();
                        if ((c == '-' && !rcgnzComments) || (c ==
[' ' && !rcgnzCDATA])) {
                            state = ST_PCDATA;
                            break;
                        }
                        if (c == '-') state = ST_COMMENT;
                        else if (c == '[') parseCDATA();
                        else {
                            warning("Bad markup " + buf);
                            state = ST_PCDATA;
                        }
                }
                break;

```

```

            case '?':
                parsePI();
                break;
            case ' ': case '\t': case '\r': case '\n':
                state = ST_TAG_WS;
                break;
            default:
                tagname.write(c);
                state = ST_TAG_NAME;
        }
        break;

        case ST_TAG_NAME:
            // ilgili kod
            break;

        case ST_TAG_WS:
            // ilgili kod
            break;

        case ST_EMPTY_TAG_SLASH:
            // ilgili kod
            break;

        case ST_NAME:
            // ilgili kod
            break;

        case ST_NAME_WS://white-space between
name and '='
            // ilgili kod
            break;

        case ST_EQ://white-space between '=' and
value
            // ilgili kod
            break;

        case ST_VALUE:
            // ilgili kod
            break;

        case ST_VALUE_QUOTED:
            // ilgili kod
            break;

        case ST_COMMENT:
            // ilgili kod
        }
    }
}

```

Şekil 5.4 “Tokenizer” nesnesinin ayrıştırma ağacını oluşturması

Şekil 5.4'da iskelet olarak verilen ayrıştırıcı bir Sonlu Durum Makinası, FSA (*Finite State Machine*) gibi çalışmaktadır. Okunan her bir harf için program ya yeni bir duruma geçer ya da bulunduğu durumu muhafaza eder.

Akış diyagramı daha detaylı incelenirse okunan her 'c' karakteri o andaki durum için değerlendirilir. Eğer doküman en başındaysa (durum= ST_START), c= '<' okununcaya kadar bekler. Bu arada başka karakterler geliyorsa o zaman ST_PCDATA (*parsed character data*) durumunda c= '<' oluncaya kadar bekler. Ne zaman ki '<' karakteri okunur, o zamana kadar okunan karakterler *gotPCDATA* fonksiyonu ile düzenlenip ağaca eklenir. (Bazı durumlarda bu bilgilerin ağaçta bulunması sorun çıkartabilir o yüzden eklenmemesi gereken durumlar olabilir.) Bundan sonraki yeni durumumuz ST_TAG_LT'dir.

Bundan sonra gelecek ilk karakter '/' ise bu okunacak etiketin kapanma tarafında olduğumuz anlamına gelir. (</p> gibi) Aksi takdirde gelecek karakterler, '>' karakterine kadar etiketin açılış tarafını oluşturur. Bu etiketin oluşumu, ST_TAGNAME durumunda gerçekleştirilir. Daha sonra etiket uygun olarak ağaca yerleştirilir. Etiket ve içerisindeki bilgiler benzer şekilde durumlar arasında uygun geçişler ile ağaca yerleştirilir.

Ayrıştırma ağacı *Tokenizer.java* isimli dosyada oluşturulmaktadır. Daha detaylı inceleme için bu dosyanın dokümantasyonuna bakılabilir.

5.2.4 "XHTMLtoWML" Dönüşümü

Bu bölüm bu tezde gerçekleştirilen sistemin beynidir. Sistemin bu kısmına giriş, düzenlenmiş ve yanlışlardan arındırılmış XHTML dokümanından oluşan ayrıştırma ağacıdır. Bu ayrıştırma ağacında rekürsif olarak dolaşarak gerekli dönüşümler yapılır. Bu dönüşüme geçmeden evvel buraya kadar sistemin çalışma akış diyagramı göstermek faydalı olacaktır. (Bkz. Ek1 Sistemin genel akış diyagramı)

Ek 1'deki akış diyagramında servis sağlayıcının çalışması şematize edilmiştir. İlk olarak bir soket açılır. Servis sağlayıcı bu soketten servis vermeye başlar. Sistem bir anda bir isteğe servis vermektedir. Bu anlamda eş zamanlı (*multithreaded*) değildir. Fakat bir servis sırasında birden çok iş parçacığı (*thread*) eş zamanlı çalıştığı için "eş zamanlı HTMLtoWML Arayüzü" (*multithreaded HTMLtoWML Proxy*) denilmiştir. Sisteme eklenen bu özellik ile çalışma hızı oldukça arttırılmıştır. Bunun nedeni her oluşturulan iş parçacığının yeni bir giriş/çıkış gerektiren iş olmasıdır. Bunlar ayrı bir iş parçası olmazsa programın sonlanma hızı oldukça düşecektir.

Ek 1'deki akış şemasında görüldüğü gibi bir servis tamamlandıktan sonra sistem "Servis sağlayıcı istek bekliyor" durumuna geçiyor. Bu geçiş ya başarılı bir WML dosyası oluşturularak olur, ya da hata dosyasına hata bilgisi yazılarak olur.

Servis sağlayıcı isteği WAP kullanıcısından alır. Bu istek ya bir URL'dir, Internet erişimi gerektir; veya yerel bir HTML doküman isteğidir. Bu kontrol yapıldıktan sonra Java TIDY konfigürasyonları yapılır. Bunlar şu şekildedir. (Şekil 5.5)

```
Tidy tidy = new Tidy(); // tidy nesnesinin oluşması
tidy.setXHTML(true); // Tidy çıkışının XHTML olması
tidy.setDocType("omit"); // giriş dökümanı 1 tane, herhangi bir DTD dökümanı yok

// Eğer URL ise, bağlantı kur
tidyIn = new BufferedInputStream(url.openStream());
// Yerel dosya ise
tidyIn = new BufferedInputStream(new FileInputStream(str));

// tidy programını çalıştır
tidy.parse(tidyIn, f);
```

Şekil 5.5 Java Tidy konfigürasyonları

Bu aşamada Tidy konfigürasyonları sistemi daha basitleştirecek hale getirilebilir. Bu gibi ince ayarlar tezin sonraki aşamalarında incelenebilir

Yukarıdaki kod parçacıklarından anlaşılacağı gibi tidyIn ile gelen HTML kodu, **f** dosyası ile işaret edilen XHTML dokümanına dönüştürülür.

Bundan sonraki aşamada **f** dosyasındaki XHTML, ayrıştırma ağacına çevrilir. Bu da aşağıdaki kod satırları ile olur.

```
Parser parser = new Parser(); // parser nesnesinin oluşturulması
Document doc = parser.parse(f); // XHTML'in ayrıştırma ağacının oluşturulması
```

Program bu aşamada daha önce incelenen ayrıştırma akış diyagramı ile ayrıştırma ağacını oluşturur. Eğer ağaç kökünün çocukları varsa, program devam edecek ve WML'e dönüşüm gerçekleşecektir. Aksi halde program hata dosyası oluşturacaktır, çünkü herhangi bir ağaç oluşmamıştır.

Son olarak, akış diyagramında görüleceği gibi, XHTML ağacı WML ağacına dönüşümü ve uygun olarak dosyaya yazılması aşağıdaki komutlar ile gerçekleştirilir. (Şekil 5.6)


```

Htmletowml outputWML = new Htmletowml(false); // Htmletowml nesnesinin
                                                    oluşturulması
outputWML.startDocument(); // nesnenin özel fonksiyonları
outputWML.traverse(doc); // WML ayrıştırma ağacının oluşturulması
outputWML.endDocument(); // nesnenin özel fonksiyonları
outputWML.print(out); // WML ayrıştırma ağacının düzgün olarak dosyaya yazılması

```

Şekil 5.6 XHTML ağacının WML ağacına dönüşümünü sağlayan komutlar

Ayrıştırma işlemi sırasında olduğu gibi, dönüşüm işlemi de herhangi bir hata durumunda program hata dosyası ile sonlanır. Normal şartlarda başarılı olarak WML çıktısı verir. Yukarıdaki kod parçasığında etiket dönüşümünü outputWML nesnesindeki **outputWML.traverse(doc)** fonksiyonu yapar. Bu fonksiyon daha detaylı olarak ayrı bir akış diyagramı ile incelenecektir.

5.2.4.1 Örnek bir Uygulama

Aşağıda basit bir HTML dokümanının yukarıdaki akış diyagramları ile geçirdiği evreler örnek olarak gösterilmiştir.(Şekil 5.7 – Şekil 5.10)

1. ADIM: HTML dokümanı:

```

<html>
  <head>
    <title>My Document</title>
  </head>
  <body bgcolor="#ff0000" link="#000000" vlink="#00003f" alink="#ffffff">
    <center>
      <table width="100%" border="0" cellspacing="0" cellpadding="0"
        height="100%">
        <tr>
          <td width="33%">- <a href="ornek1.htm">ORNEK1.HTM...</a>
          <td width="33%">
            <center>- <a href="ornek2.htm"> ORNEK2.HTM ...</a>- <a
              href="ornek1.htm"> ORNEK1.htm</a></center>
          </td>
          <td width="883">
            <p align="center">
              <small>
                <font face="Arial">
                  First Consultant UK Ltd
                </font>
              </small>
            </p>
          </td>
        </tr>
      </center>
    </body>
  </html>

```

Şekil 5.7 Örnek bir HTML dokümanı

Şekil 5.7'deki HTML dosyasında bazı etiketler doğru kapatılmamıştır. 2.Adımda bu HTML'den Tidy işlemi sonunda oluşan XHTML dokümanı görülmektedir.

2.ADIM: XHTML dokümanı:

```
<html>
<head>
<meta name="generator" content="HTML Tidy, see www.w3.org" />
<title>My Document</title>
</head>
<body bgcolor="#ff0000" link="#000000" vlink="#00003f"
alink="#ffffff">
<center>
<table width="100%" border="0" cellspacing="0" cellpadding="0"
height="100%">
<tr>
<td width="33%">- <a href="ornek1.htm">ORNEK1.HTM...</a> </td>
<td width="33%">
<center>- <a href="ornek2.htm">ORNEK2.HTM ...</a> - <a
href="ornek1.htm">ORNEK1.htm</a></center>
</td>
<td width="883">
<p align="center"><small><font face="Arial">First Consultant UK
Ltd</font></small></p>
</td>
</tr>
</table>
</center>
</body>
</html>
```

Şekil 5.8 TIDY işlemi sonunda oluşan XHTML dokümanı

Bu adımda yukarıdaki XHTML oluşurken aşağıdaki TIDY raporu görülmektedir.

```
* ***** *
Tidy (vers 4th August 2000) Parsing "InputStream"
line 10 column 1 - Warning: <table> lacks "summary" attribute
line 28 column 1 - Warning: missing </table> before </center>

InputStream: Document content looks like HTML 3.2
2 warnings/errors were found!
* ***** *
```

3.Adımda **Document doc = parser.parse(f)** komutu ile ayrıştırma ağacı oluşturulur.

Şekil 5.7'deki HTML dosyasında bazı etiketler doğru kapatılmamıştır. 2.Adımda bu HTML'den Tidy işlemi sonunda oluşan XHTML dokümanı görülmektedir.

2.ADIM: XHTML dokümanı:

```
<html>
<head>
<meta name="generator" content="HTML Tidy, see www.w3.org" />
<title>My Document</title>
</head>
<body bgcolor="#ff0000" link="#000000" vlink="#00003f"
alink="#ffffff">
<center>
<table width="100%" border="0" cellspacing="0" cellpadding="0"
height="100%">
<tr>
<td width="33%">- <a href="ornek1.htm">ORNEK1.HTM...</a> </td>
<td width="33%">
<center>- <a href="ornek2.htm">ORNEK2.HTM ...</a> - <a
href="ornek1.htm">ORNEK1.htm</a></center>
</td>
<td width="883">
<p align="center"><small><font face="Arial">First Consultant UK
Ltd</font></small></p>
</td>
</tr>
</table>
</center>
</body>
</html>
```

Şekil 5.8 TIDY işlemi sonunda oluşan XHTML dokümanı

Bu adımda yukarıdaki XHTML oluşurken aşağıdaki TIDY raporu görülmektedir.

```
* *****
Tidy (vers 4th August 2000) Parsing "InputStream"
line 10 column 1 - Warning: <table> lacks "summary" attribute
line 28 column 1 - Warning: missing </table> before </center>

InputStream: Document content looks like HTML 3.2
2 warnings/errors were found!
* *****
```

3.Adımda **Document doc = parser.parse(f)** komutu ile ayrıştırma ağacı oluşturulur.

3.ADIM: XHTML Ayırıştırma ağacının Oluşması:

```
<html >
  <head >
    <meta name=generator content=HTML Tidy, see www.w3.org />
    <title >
      My Document
    <body bgcolor=#ff0000 link=#000000 vlink=#00003f alink=#ffffff>
    <center >
      <table width=100% border=0 cellpadding=0 cellspacing=0 height=100%>
        <tr >
          <td width=33%>
            -
            <a href=ornek1.htm>
              ORNEK1.HTM...
          <td width=33%>
            <center >
              -
              <a href=ornek2.htm>
                ORNEK2.HTM ...
              -
              <a href=ornek1.htm>
                ORNEK1.htm
          <td width=883>
            <p align=center>
              <small >
                <font face=Arial>
                  First Consultant UKLtd
```

Şekil 5.9 Ayırıştırma sonunda oluşan XHTML ağacı

Yukarıda oluşan ayırıştırma ağacında (Şekil 5.9) etiketlerin sadece açılma tarafları görülmektedir. Kapatma etiketleri ekrana yazılmamaktadır. Ağaçta görünen her satır bir düğüme karşılık gelmektedir.

Ağaçtaki düğümler aşağıdaki tiplerde olabilir:

- Document
- Element
- Attribute
- PI
- Comment
- Text

Ağaçta her düğümün ismi, varsa özelliği (*Attribute*), metin (*text*) ise metnin kendisi yazılmaktadır. *Document* ise herhangi bir ağacın kökündeki düğümdür.

Son adım olarak yukarıdaki ayırıştırma ağacı WML ağacına ve dokümanına aşağıdaki gibi

dönüştürülür.

4. ADIM: WML Ayrıştırma Ağacı ve dokümanı:

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE wml SYSTEM "C:\PROGRAM
FILES\3TL\WBUILDER PROFESSIONAL\DTD\wml.dtd">
<wml>
  <card id="Results" title="My Document">
    <p>
      <table columns="3">
        <tr>
          <td>
            -
            <a href="ornek1.wml">
              ORNEK1.HTM...
            </a>
          </td>
          <td>
            -
            <a href="ornek2.wml">
              ORNEK2.HTM ...
            </a>
          </td>
          <td>
            -
            <a href="ornek1.wml">
              ORNEK1.htm
            </a>
          </td>
        </tr>
      </table>
      First Consultant UKLtd
    </p>
  </card>
</wml>
```

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE wml SYSTEM "C:\PROGRAM
FILES\3TL\WBUILDER PROFESSIONAL\DTD\wml.dtd"><wml><card id="Results" title="My
Document"><p><table columns="3"><tr><td><a
href="ornek1.wml">ORNEK1.HTM...</a></td><td><a href="ornek2.wml">ORNEK2.HTM ...-
</a><a href="ornek1.wml">ORNEK1.htm</a></td><td>First Consultant
UKLtd</td></tr></table></p></card></wml>
```

Şekil 5.10 Dönüşüm sonunda oluşan WML ağacı ve dokümanı

Oluşan ağaç ve dokümanda görüldüğü gibi (Şekil 5.10), etiketler ve özellikleri WML biçimlendirme dilinin kurallarına uygun olarak XHTML'den farklıdır. Bu dönüşümün akış diyagramı ve dönüşüm kuralları bir sonraki bölümde incelenecektir.

Yukarıdaki basit örnekte programın çalışması sırasında önemli ara adımlarda oluşan ağaç ya da dokümanlar gösterilmiştir. Programın normal çalışmasında WAP kullanıcısı istediği dokümanın adresinin WML Tarayıcısında (*browser*) belirtir ve program sonlanınca da WML dokümanını tarayıcıda görür.

5.2.5 Dönüşüm Akış Şeması ve Kuralları

Bir önceki bölümde belirtildiği gibi XHTML ağacı WML ağacına

outputWML.traverse(doc)

rekürsif fonksiyonu ile dönüştürülür. Bu Htmltowml nesnesinin bu fonksiyonu sistemde

Htmltowml.java dosyasında bulunmaktadır. Bu java dokümanına giriş, XHTML ağacının kökü olan **doc** değişkenidir. Bu nesnedeki ilk değer atamaları Ek 2'deki gibidir. Rekürsif yapıda olan “public void **traverse**(org.w3c.dom.Node node)” fonksiyonun akış şeması da Ek 3'de gösterilmiştir. Bu fonksiyon ilk olarak **traverse(doc);** komutu ile servis sağlayıcı tarafından çağırılır.

Ek 3'de genel olarak şematize edilmiş **traverse(node)** fonksiyonunda 3 ayrı mod vardır. Birinci mod sistemin *skipMode*'da çalışma durumudur. Sistemin *skipMode*'da olması tablo içindeki tablonun yeni bir WML oluşturmaya demektir. Bu modda iken karşılaşılan etiketler yeni WML dokümanının bilgileridir. Tablonun <td> etiketinde karşılaşılan <table> etiketi yerine yeni oluşturulan WML dokümanına link yazılır. Şekil 5.11'de bu modda XHTML'in WML'e dönüşümü küçük bir örnek ile görülmektedir.

örnek2.xhtml <pre> <table border="0" cellpadding="0" width="600" cellspacing="4"> <tr> <td bgcolor="#acb2cc" valign="top"> <table border="0" width="200" cellpadding="2" cellspacing="0"> <tr> <td colspan="1"> &nbsp; Volcano Hazards </td> </tr> </table> </td> </tr> </table> </pre>
örnek2.wml (örnek2.html'de kalın yazılan kod yerine aşağıda kalın yazılan kod üretilmiştir) <pre> <table columns = "1"> <tr> <td> Daha fazla bilgi için linki tıklayınız </td> </tr> </table> </pre>
table1.wml (iç tablo yerine aşağıdaki wml dokümanı oluşturulmuştur) <pre> <table columns = "1"> <tr> <td> Volcano Hazards </td> </tr> </table> </pre>

Şekil 5.11 İç içe tabloların çözülmesi örneği(skip mod)

İkinci çalışma modu her bir elementi tanıyan ve gerekli olan WML kodunu üreten bölümdür. Bu dönüşümlerin kaynak kodu için htmltowml.java dosyasına bakılabilir. Sistemde hangi XHTML etiketinin ve özelliklerinin hangi WML etiket ve özelliklerine dönüştürüldüğü genel bir tabloda görülmektedir. (Çizelge 5.4) Bu tabloda daha önceki tabloya (Çizelge 5.1) benzer

olarak HTML, XHTML ve bunlara karşılık gelen etiketler görülmektedir. Sistemde WML etiketlerinin zorunlu olan özellikler kalın (**bold**), diğerleri normal özellikte gösterilmiştir.

Tabloda gri kalın kolon dönüşümü göstermektedir. Tablonun sağ tarafı dönüştürülen WML etiketlerini gösterir. Bazı değişik durumlar için tablonun altında ek olarak bilgi verilmiştir. Tabloda WML kolonunun en sağındaki kolon, karşı gelen etiketin içinde hangi etiketin olabileceğini gösterir. Mesela etiketi herhangi etiket veya metin içermez, diğer yandan <p> etiketi tablodaki bütün etiketleri içine alabilir.

Çizelge 5.4 Arayüzde uygulanan dönüşüm kuralları

ETİKET	HTML	XHTML	WML		
	Özellikler	Özellikler	ETİKET	Özellikler	İçeriği
a	class, id, style, title, charset, type, name, href, hreflang, rev, rel, accesskey, shape, coords, tabindex, onfocus, onblur	accesskey, charset, href, hreflang, rel, rev, tabindex, title, type	a	href , title, id	metin, br, img
b	class, id, style, title		b		metin, img, table, a
i	class, id, style, title		i		metin, img, table, a

img	class, id, style, title, src, alt, longdesc, height, width, usemap, ismap	alt, height, longdesc, src, style, title, width	img	alt, src, align, height, width,	
p	class, id, style, title	align, mode	p	align	metin, a, b, i, img, font, table, td, tr, u, small, frameset, frame, h1, h2, h3
font	class, id, style, title	align, mode	font	align	metin
table	class, id, style, title, summary, width, border, frame, rules, cellspacing, cellpadding	align, style, summary, title, width	table	columns	tr
td	class, id, style, title, abbr, axis, headers, scope, rowspan, colspan	abbr, align, axis, colspan, headers, rowspan, scope, style, title, valign	td		metin, img, a
tr	class, id, style, title	style, title, valign	tr		td
u	class, id, style, title		u		metin, img, a, table

small	class, id, style, title		small		metin, img, a, table
frame set	class, id, style, title, rows, cols		*		
frame	class, id, style, longdesc, name, longdesc, name, src, frameborder, marginwidth, marginheight, noresize, scrolling		a*	href	
h1, h2, h3	class, id, style, title	Align	**em		metin

* WML <frameset> etiketini desteklemiyor. Bunun yerine bu bilgiyi belirten metin görüntülenir. <frameset> etiketi içindeki her <frame> etiketi için de bir link ile başka bir wml dokümanına erişim sağlanır.

** WML <h1> <h2> <h3> etiketlerini desteklemediği için bunların yerine ile baskılı görüntü gerçekleştirilmiştir.

Tablo HTML, XHTML ve WML biçimlendirme dillerinin DTD (*Data Type Definition*) dosyaları incelenerek oluşturulmuştur. Aynı zamanda tablo sadece sistemin içerdiği etiketleri göstermektedir.

Sistem üzerinde daha ileri bir çalışmada DTD dosyalarından faydalanarak sistemin çalışması genişletilebilir.

Sistemin üçüncü çalışma modu 3 numaralı kutuda yapılan işlemlerdir. Burada okunan herhangi bir metin WML ağacında o andaki etikete çocuk düğüm (*child node*) olarak eklenir.

Bundan sonraki bölümde sistemin skipMode'da, etiket dönüşümünde ve metin ekleme aşamasında karşılaşılan zorluklar ve bu problemlerin çözümü için geliştirilen basit algoritmalar anlatılmıştır. Bu kısım yukarıda genel akış şeması verilen rekürsif fonksiyonun (**traverse (node) fonksiyonu**), 1, 2 ve 3 numara ile çalışma modlarını ifade eden kutuların içindeki algoritmalar ve genel akış diyagramlarıdır.

5.2.6 Uygulamada Kullanılan Temel Algoritmalar

Bu bölümde etiket değişiminde kullanılan algoritmalar sunulacaktır. İncelemeye ilk olarak birinci moddan başlayacağız. Çünkü sistemin çalışma zamanını en çok kaplayan kod bu bölümde bulunuyor. Akış diyagramında kullanılan değişkenler ve fonksiyonlar aşağıdaki gibidir.

Değişkenler:

Bu aşamada giriş bilgisi sadece **“child”** değişkenidir. **“child”** değişkeni XHTML ağacından okunan düğümdür. Diğer önemli bir değişken **“current”** değişkenidir. **“current”** işaretçisi programın çalışma esnasında WML ağacında gösterdiği düğümdür. Bununla beraber programın çalışmasında kontrol amaçlı kullanılan diğer değişkenler ve ilk değerleri şunlardır:

```
following = null
followingTable = null
followingA = null
parentofP = null
parentofA = null
skipTo = null
inP = false
inA = false
numberofTD = 0
skipMode = false
```

Diğer global değişkenler (**current** gibi) daha önceden belirtilmiş olup, `htmltowml.java` dosyasından daha detaylı olarak bakılabilir.

Fonksiyonlar:

traverse(node x) fonksiyonu dönüşümü sağlayan ana fonksiyonumuz olup rekürsif olarak çağırılmaktadır. Kökü **x** düğümü olan ağacın bütün düğümlerini dolaşır.

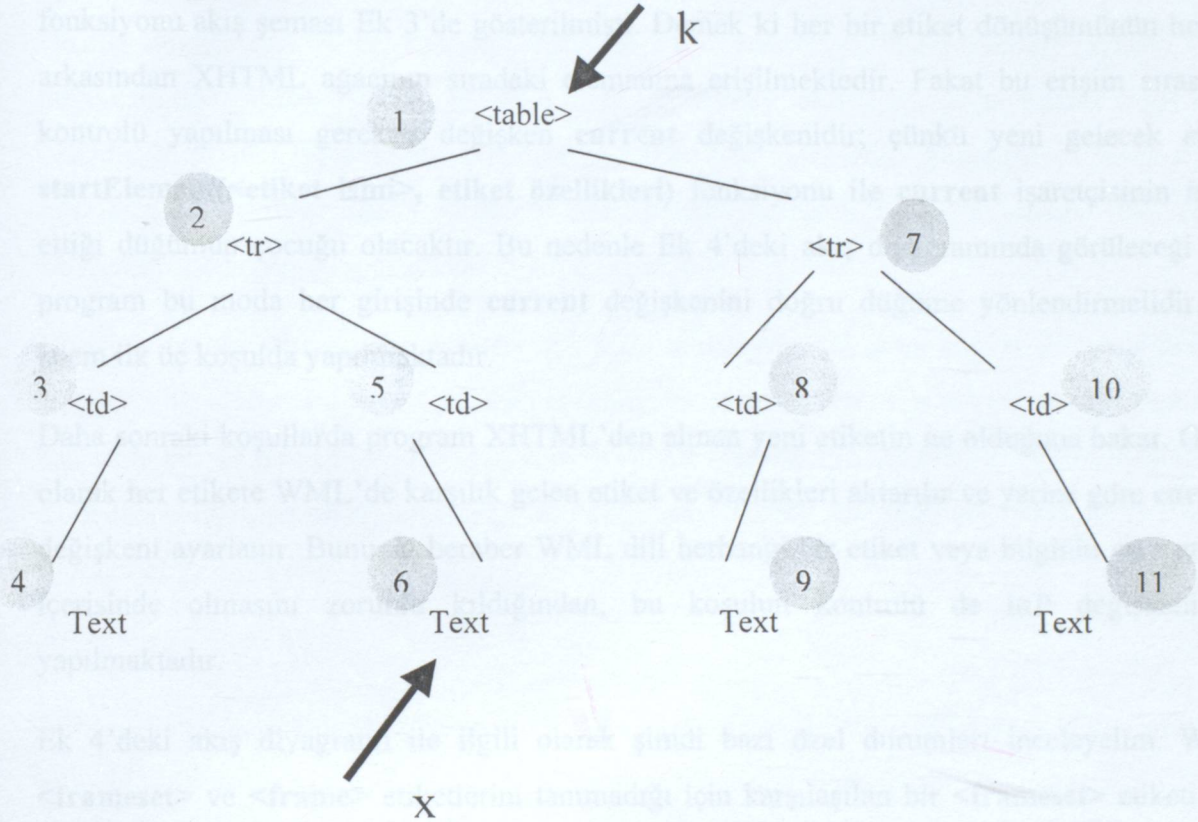
startElement(<etiket ismi >, etiket özellikleri) fonksiyonu eklenecek etiketi düğüm (*node*) değişkenine çevirerek **current** işaretçisinin işaret ettiği düğümün sıradaki çocuğu olarak ağaca ekler.

node.parentNode() fonksiyonunu hangi düğüm(*node*) çağırıyorsa o düğümün ebeveyn düğümüne ulaşılır.

getFollowing(node x) fonksiyonu **traverse** fonksiyonunda **x** düğümünden bir sonra gelen elemanı belirler. Örneğin, ağaçtaki düğümleri bir dizide şu şekilde sakladığımızı düşünelim:

Ağacın kökünden başlayarak önce düğümün kendisi hemen sonra soldan sağa çocuklar olmak üzere bütün elemanları diziye yerleştiririm. Bu fonksiyon ile dizideki **x** düğümünün hemen sağındaki elemana ulaşılır.

Örnek bir ağaç:



Yukarıdaki ağaçta **x** işaretçisi 6 numaralı düğüme, **k** işaretçisi de 1 numaralı düğüme işaret etmektedir. Buna göre aşağıdaki fonksiyonlar ve işaret ettikleri yeni düğümler şöyledir:

getFollowing(x) 7 numaralı düğümü gönderir.

x.getParentNode() 5 numaralı düğümü gönderir.

startElement("tr", wmlattrs) <tr> etiketi için düğüm oluşturur. **current** işaretçisinin işaret ettiği düğümün sıradaki çocuğu olarak ekler.

traverse(k) fonksiyonu ağacı belirtilen düğüm sırasına göre dolaşır.

Bu bilgiler ışığında sistemin birinci modunun akış diyagramı genel olarak Ek 4'deki gibidir. Diğer bazı fonksiyonlar ve işlemler kısaltma amacıyla sembolik olarak gösterilmiş veya metin olarak anlatılmıştır: dokümantasyon için `htmltowml.java` dosyasına bakılabilir. (Mesela "gerekli özelliklerin aktarılması" notu daha önce Çizelge 5.4 tablosunda belirtilen kurallara

göre WML etiketindeki zorunlu ve bazı özelliklerin belirlenmesi işlemleridir.)

BİRİNCİ MOD (Etiket Dönüşüm Modu):

Ek 4’de ana hatları ile verilen akış diyagramında her koşuldan sonra çağırılan **traverse(child)** fonksiyonu akış şeması Ek 3’de gösterilmişti. Demek ki her bir etiket dönüşümünün hemen arkasından XHTML ağacının sıradaki elemanına erişilmektedir. Fakat bu erişim sırasında kontrolü yapılması gereken değişken **current** değişkenidir; çünkü yeni gelecek etiket **startElement(<etiket ismi>, etiket özellikleri)** fonksiyonu ile **current** işaretçisinin işaret ettiği düğümün çocuğu olacaktır. Bu nedenle Ek 4’deki akış diyagramında görüleceği gibi program bu moda her girişinde **current** değişkenini doğru düğüme yönlendirmelidir. Bu işlem ilk üç koşulda yapılmaktadır.

Daha sonraki koşullarda program XHTML’den alınan yeni etiketin ne olduğuna bakar. Genel olarak her etikete WML’de karşılık gelen etiket ve özellikleri aktarılır ve yerine göre **current** değişkeni ayarlanır. Bununla beraber WML dili herhangi bir etiket veya bilginin **<p>** etiketi içerisinde olmasını zorunlu kıldığından, bu koşulun kontrolü de **inP** değişkeni ile yapılmaktadır.

Ek 4’deki akış diyagramı ile ilgili olarak şimdi bazı özel durumları inceleyelim. WML **<frameset>** ve **<frame>** etiketlerini tanımadığı için karşılaşılan bir **<frameset>** etiketi için "Frames! Bir frame seçin!" bilgisi ekrana basılır ve hemen altına karşılaşılan **<frame>** etiketlerinin her biri için bir link belirlenir. Böylece kullanıcı **<frameset>** yapısını ilk ekranda göremez fakat linkler ile bu **<frame>** etiketleri içerisindeki bilgilere ulaşabilir.

<a> etiketi programda çözülmesi en çok satır kaplayan kısımdır. Bunun nedeni **href** özelliğindeki URL bilgisinin doğru olarak belirlenmesidir. Bu noktadaki temel sorun **href** ile alınan URL’in hangi biçimde olduğudur.

URL yapısı [13]’de tanımlanmıştır. Buna göre örneğin

<http://www.ktp.yildiz.edu.tr:80/ytuyer3#H> adresi için

<scheme>: http

<host> : www.ktp.yildiz.edu.tr

<port>: 80

<path>: ytuyer3

<fragment>: H

bilgileri elde edilir. Sistemde okunan **href** bilgisinden elde edilen bu bilgilere göre bu URL

adresine yeni bir link belirlenir. Mesela alınan **href** sadece “**path**” bilgisi içeriyorsa, yeni açılacak olan **socket**, ana sayfanın URL’i “**host**” olmak üzere, bu “**host**”a “**path**”in eklenmesiyle oluşturulan URL üzerinden kurulur. **<a>** ve **** etiketlerindeki bu gibi benzer kontroller daha detaylı olarak `htmltowml.java` dosyasından incelenebilir.

Birinci modda son olarak **<table>**, **<tr>** ve **<td>** etiketlerini inceleyeceğiz. Alınan **child** değişkeni eğer **<table>** ise ve sistem o anda bir **<table>** etiketi içindeyse sistem *skipModda* çalışmaya başlar. Eğer böyle bir durum yoksa yeni bir tablo oluşturulur. Sistemin bu aşamasındaki önemli nokta **<table>** elemanındaki zorunlu olan “**columns**” özelliğinin belirlenmesidir. Bunun için şöyle bir yol izlenmiştir:

Karşılaşılan her **<tr>** etiketi için bu etiketin çocuklarının sayısına bakılır ve **numberofTD** değişkenine yazılır. Böylece tablodaki herhangi bir satırdaki en fazla sayıda kolon belirlenmiş olur. Sistem ne zaman ki tablonun sonuna ulaşır (bunun için de **followingTable** değişkeni kullanılmıştır), o zaman **numberofTD** değişkeni **<table>** elementinin “**columns**” özelliği olarak belirlenmiş olur.

İKİNCİ MOD (SkipMode):

İçile tablo olması halinde iç kısımdaki tablo için ayrı bir kart (*card*) yazılır ve bu yeni karta ana tabloda bulunduğu hücreden link ile ulaşılır. Bundan sonra linkin işaret ettiği WML dosyası yazılır. Bu doküman iç kısımda bulunan tablonun içeriği olacaktır.

Sistemin bu moddaki akış diyagramları genel olarak Ek 5 ve Ek 6’daki gibidir. İlk akış diyagramı sistemin *skipModda* girişini gösterir.(Bkz. Ek 5) İkinci akış şeması ise **traverse(node)** fonksiyonunda *skipMod* koşulundan sonraki komutları ifade eder.(Bkz. Ek 6)

Sistem *skipModda* geçerken bu moddan ne zaman çıkacağını **skipTo** işaretçisi ile belirler. “Skip Modda Çalışma” akış diyagramında (Ek 6) görüldüğü gibi **child=skipTo** koşulu sağlandığında *skipMode* sona erer ve ana sayfa kaldığı yerden devam eder. Bu koşul sağlanıncaya kadar **<table>** (iç kısımdaki **<table>**) köklü ağaç üzerinde **traverse(child)** fonksiyonu çağırılır. Böylece okunan bilgi yeni dosyaya (`tableFile.html`) yazılmış olur.

`tableFile.html` dosyası yazıldıktan sonra bu dosyanın WML formatına dönüşümü için program ayrı bir iş parçası (*thread*) olarak tekrar çağırılır. Bu eşzamanlı çalışma hem programın sonlanma zamanını azaltacaktır hem de programın hatalı sonlanma olasılığını düşürecektir. Çünkü eğer iç tabloda bir hata varsa (herhangi bir nedenden dolayı ayrıştırılamıyorsa) sadece bu bilgi kaybolacaktır. Ana sayfamız doğru biçimlendirilmiş olarak görüntülenecektir.

ÜÇÜNCÜ MOD (Metin Yazma Modu):

Sistem bu modda etiketlerin metin (*Text*) tipindeki çocuk düşümlerini etiketlere ekler. Diğer bir ifade ile elementlerin içersindeki metin bilgilerinin aktarılması bu modda sağlanır. Bu modun akış diyagramı da Ek 7’de gösterilmiştir.

Sistem “metin yazma moduna” girdiğinde okunan metin XHTML’deki <title> etiketinin bilgisi ise bu bilgi WML’de <card> etiketinin özelliği olarak aktarılır.

Diğer bir koşulda **p** (**child**’ın ebeveyni) işaretçisinin işaret ettiği düğüm bir *Element* olmak üzere, programın o anda bir <p> etiketinde olup olmadığına bakılır. **cleanData(metin)** fonksiyonu metin içersinde WML ile tanınmayan bazı karakterleri (<, > gibi) temizler.

Bu moddan çıkmadan önce son olarak **current** değişkeninde gerekli düzeltmeler yapılır. Diyagramda görülen koşullarda (<i> <u> <small>) metin yazıldıktan sonra **current** işaretçisi bu etiketlerden bir üst elemana (ebeveyne) geçer.

5.2.7 Sistemin Test Edilmesi ve Sistemin Sorunları

Sistem 20’ye yakın yerel HTML dosyaları üzerinde (Internet’e erişim gerektirmeden) ve 15’e yakın URL üzerinde denenmiştir. Yerel dosyalar karmaşıklığı az olmakla beraber sistemin içerdiği elemanları mümkün olduğunca değişik kombinasyonlarda örnekleyerek oluşturulmuştur. Sistemin (etiket dönüşüm kısmı) ilk iskeletinden sonra gelişmesi ve her test ile daha kapsamlı ve doğru çalıştığı gözlenmiştir. Internet erişimi gerektiren testlerde sistemin kilitlenmeden bir wml dosyası oluşturma oranı yüksektir. HTML’in düzensiz yapısı, her ne kadar XHTML’e dönüştürülüyorsa da, bu oranın daha yukarılara çıkmasına engel olmaktadır. Veri kaybında belli bir artış ile bu oran yükseltilebilir.

Sistemde yaygın olarak 2 tür hata ortaya çıkmaktadır:

- HTML’in XHTML’e dönüştürülemeyecek kadar düzensiz olması. Bunun kontrolü Ek 1’deki akış diyagramında görülmektedir. Bu durumda *hata.wml* dosyasına WML görüntülenemeyeceği yazılır ve çıkışa gönderilir.
- Herhangi bir nedenden dolayı URL bağlantısı kurulamaması. Böyle bir durum için bir zaman ölçere (*timer*) gerek kalmadan *JAVA*’nın *exception* nesneleri ile bu hatalar yakalanır ve *hata.wml* dosyasına gerekli bilgiler yazılır.

Bazı büyük dosyalar için sisteme bir zaman kontrolü eklemek gerekebilir. Belki büyük HTML dosyaları ve büyük görüntü dosyaları için belli bir zamandan sonra program bu

dosyalardan vazgeçip kaldığı yerden devam etmelidir. Bu soruna başka bir çözüm olarak, görüntü veya büyük HTML sayfalarını indirmeden tespit etmek ve belli büyüklükte sayfalar ile karşılaşınca kullanıcıya baştan böyle bir bilgi vermek olabilir. Fakat yapılan denemeler sonucunda indirecek olan dosyanın büyüklüğü ya bilinmiyor veya bu bilgiye erişmek ayrı bir bağlantı (*connection*) gerektirdiği için yine ilk durumda olduğu gibi programın sonlanma zamanı uzuyor.

Sistemin bir diğer sorunu bilginin kaybolması sorunudur. Bunun temel nedeni sistemin şu andaki kapsamıdır. Sistemdeki etiket dönüşümü direk olarak Çizelge 5.4'deki dönüşüm kuralları ile belirlenir. WML bu tabloda görünen elemanlara ek olarak <access>, <do>, <fieldset>, <go>, <head>, <input>, <meta>, <option>, <select>, <timer> gibi önemli etiketleri de kullanmaktadır. Bu tezde geliştirilen sistem sadece kullanıcının HTML isteğine cevap vermektedir. Sistemin daha sonraki olası aşamalarında bu sıralanan etiketler de kullanarak kullanıcıdan bilginin alınması, form doldurma, güvenli erişim gibi fonksiyonlar üzerinde çalışılabilir.

Farklı servis sağlamadaki böyle bir kısıtlamanın yanısıra veri kaybının temel nedeni programın etiket dönüşümü aşamasında yakaladığı etiketlerin sadece en önemli etiketler olmasıdır. XHTML ağacından gelen etiketlerden üzerinde işlem yapılanlar aşağıda sıralanmıştır:

<p>, , <table>, <tr>, <td>, <h1>, <h2>, <h3>, <i>, , <u>, <small>, <a>, , <frameset>, <frame>

Bu listenin bir XHTML dökümanından WML dökümanı oluşturmak için en çok gerekli olan etiketler olduğu düşünülmüştür. En çok kullanılan bu XHTML etiketleri ile veri kaybını en aza indirmek mümkün olacaktır. Yine de HTML'in mevcut etiket sayısı yüzün üzerinde olduğu düşünülürse sözkonusu veri kaybı kaçınılmaz olacaktır.

Bununla beraber sistemin karmaşıklığını arttırmadan veri kaybını azaltmanın bir yolu XHTML etiketleri ile WML etiketleri arasında 1'e 1'lik bir ilişki yerine n'e 1'lik bir ilişkinin yolunu bulmaktır. HTML'de benzer etiketler belirlenerek bunlar bir WML etiketine karşılık getirilebilir. Böyle bir sıkıştırma bu tezde tasarlanan sistemde <h1>,...<h6> etiketleri için yapılmıştır. Bu XHTML etiketlerinin hepsi etiketine dönüştürülmüştür.

Sistemin en çok zaman harcadığı kısım HTML → XHTML dönüşümünün olduğu kısımdır. Sisteme **import org.w3c.tidy.*;** ile dahil edilen Java Tidy HTML → XHTML dönüştürücü.

dönüştürme işlemi için gerekli konfigürasyonlardan sonra istenilen URL yerine bağlantı kurar ve dosyayı indirip XHTML'e dönüştürür. Java TIDY böyle bir uygulama için ortaya atılan belki tek programdır. XHTML dosyası oluşturma gibi ara bir basamak olmadığını düşünürsek sistemin hızını belli oranda arttırabiliriz. Fakat bu sefer sistemin hata dosyası oluşturma oranı yükselecektir. Tasarımdaki bu gibi değişiklikler ve yenilikler gelecekte sistem üzerinde yapılması muhtemel olan çalışmalar olarak görülebilir.

6. SONUÇ

Bu tezde kablosuz kullanıcı uç birimlerinin HTML sayfalarını görüntülemesine olanak veren sistemin tasarımı ve uygulanması gerçekleştirilmiştir. JAVA programlama dili kullanarak geliştirilen program mevcut diğer programlar (WML tarayıcı, XHTML ayrıştırıcı) ile birleştirilerek basit bir WAP Arayüz tasarlanmıştır. Böylece kablosuz kullanıcıyı temsil eden WML tarayıcı özerinden URL isteği yapılır ve bu istek arayüze aktarılır. Arayüzde düzensiz yapıdaki HTML, kurallı yapıdaki XHTML dönüştürüldükten sonra, XHTML ağacı oluşturulur. Bu ağaç WML ağacına dönüştürülür ve oluşan WML formatındaki bilgi WML tarayıcısına aktarılır. Sistem birçok farklı yapıda HTML sayfası üzerinde denenmiştir. Genel olarak küçük HTML sayfalarında başarı oranı oldukça yüksektir. İçile tablo bulunduran HTML sayfalarında da başarı oranı yüksektir. Bununla beraber bazı iç tablolar veri kaybına maruz kalmaktadır. Büyük boyutlu HTML sayfaları doğru olarak WML dokümanına dönüştürülse bile WML tarayıcı büyük dosyaları görüntülemeye kısıtlamalar getirmektedir.

Tezde genel olarak sistemin tasarımı ve uygulanması aşamaları ve karşılaşılan sorunlar yanı sıra, sistemin eksiklikleri incelenmiştir. Sistemdeki bu eksiklikler sistemin daha sonraki gelişme aşamalarına ışık tutmaktadır.

[5] <http://www.project-horse.net/html/int-ov/ovsazi.html>

[6] <http://www.wapforum.org/wml/index.htm>

[7] <http://sunsite.unc.edu/pub/information/standards/xml/why/xmlapps.htm> (Borak, J., (1997), XML, Java, and the future of the Web, Sun Microsystems).

[8] <http://www.rtf.com/wap/articles/html/wml/default.asp> (Paul Howard "Converting HTML to WML")

[9] <http://www.cs.toronto.edu/~korpela/papers/html2wml.html> (Gültekin Korpela, "A tutorial on Character Code Issues")

[10] <http://www.cs.lal.ca/~doug99/> (Doug Korpela "A Report on WAP/WML")

[11] <http://www.w3.org/TR/xslt/> (XSL Transformations (XSLT), Version 1.0 W3C Recommendation November 1999)

[12] <http://nhs.sourceforge.net/> (HTML Tidy Library Project)

[13] <http://www.ietf.org/rfc/rfc2396.txt> ("Uniform Resource Identifiers" RFC2396, Uniform Resource Identifiers (URI): Generic Syntax, amended by RFC 2732, Format for Embedding IPv6 Addresses in URIs, 1998)

KAYNAKLAR

Onur, E.,(2000), Traffic Modeling and Optimization in Cellular Networks, MS Thesis, BÜ Fen Bilimleri Enstitüsü

Mann, S., (1999), Programming Applications with WAP: The Complete Developer's Guide, John Wiley & Sons

Kaasinen, E., Aaltonen, M., Kolari, J., Melakoski, S. ve Laakko, T., (May 2000), "Two Approaches to Bringing Internet Services to WAP Devices", WWW-9 conference. Amsterdam.

Morrison, M., et al., (2000), "XML Unleashed", Sams Publishing

Seçilir, D., (2002), WML Browser, Senior Project, YTU Computer Sciences and Eng. Department

INTERNET KAYNAKLARI:

[1] <http://www-scf.usc.edu/~csci571/> (Ellis Horowitz, 1999-2000 CS571 Course Notes)

[2] <http://www.w3.org> (W3C Konsorsiyumu)

[3] www.wirelessdevnet.com/channels/wap/training

[4] <http://www.columbia.edu/~ir94/wireless.html#Intro> (Irene Aldridge, Prof. M. Lerner, (Summer 2000), "Analysis of Existing Wireless Communication Protocols", Semester Project, Columbia University)

[5] <http://www.project-house.net/html/intsoz/intsozi.html>

[6] <http://www.wapforum.org/what/index.htm>

[7] <http://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm> (Bosak, J., (1997), XML, Java, and the future of the Web, Sun Microsystems)

[8] <http://www.topxml.com/wap/articles/htmlwml/default.asp> (Paul Howard "Converting HTML to WML")

[9] <http://www.cs.tut.fi/~jkorpela/chars.html#basics> (Jukka Korpela, "A tutorial on Character Code Issues")

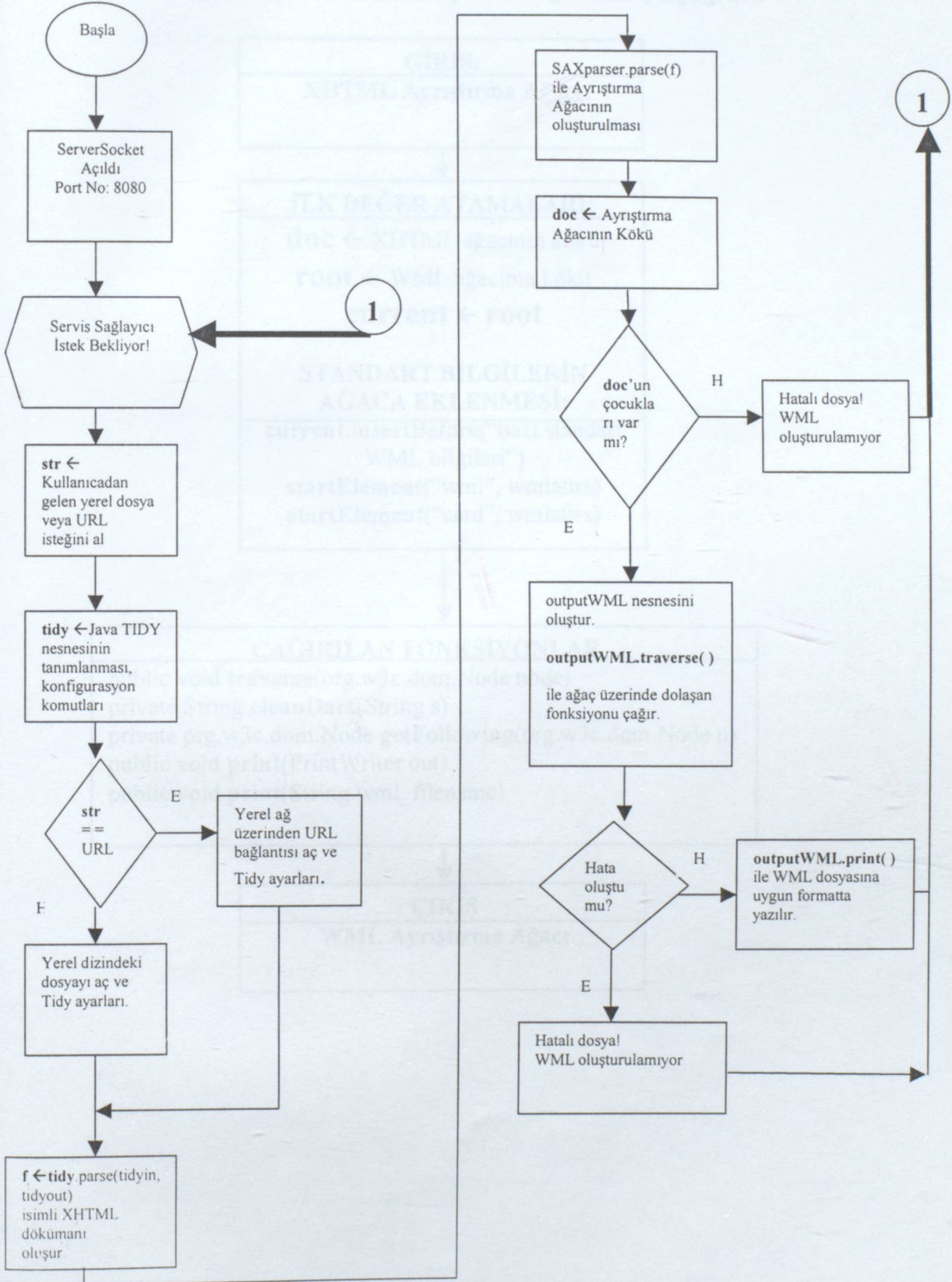
[10] <http://www.cis.ksu.edu/~deep/690/> (Deep Kapadia "A Report on WAP/WML")

[11] <http://www.w3.org/TR/xslt/> (XSL Transformations (XSLT) Version 1.0 W3C Recommendation November 1999)

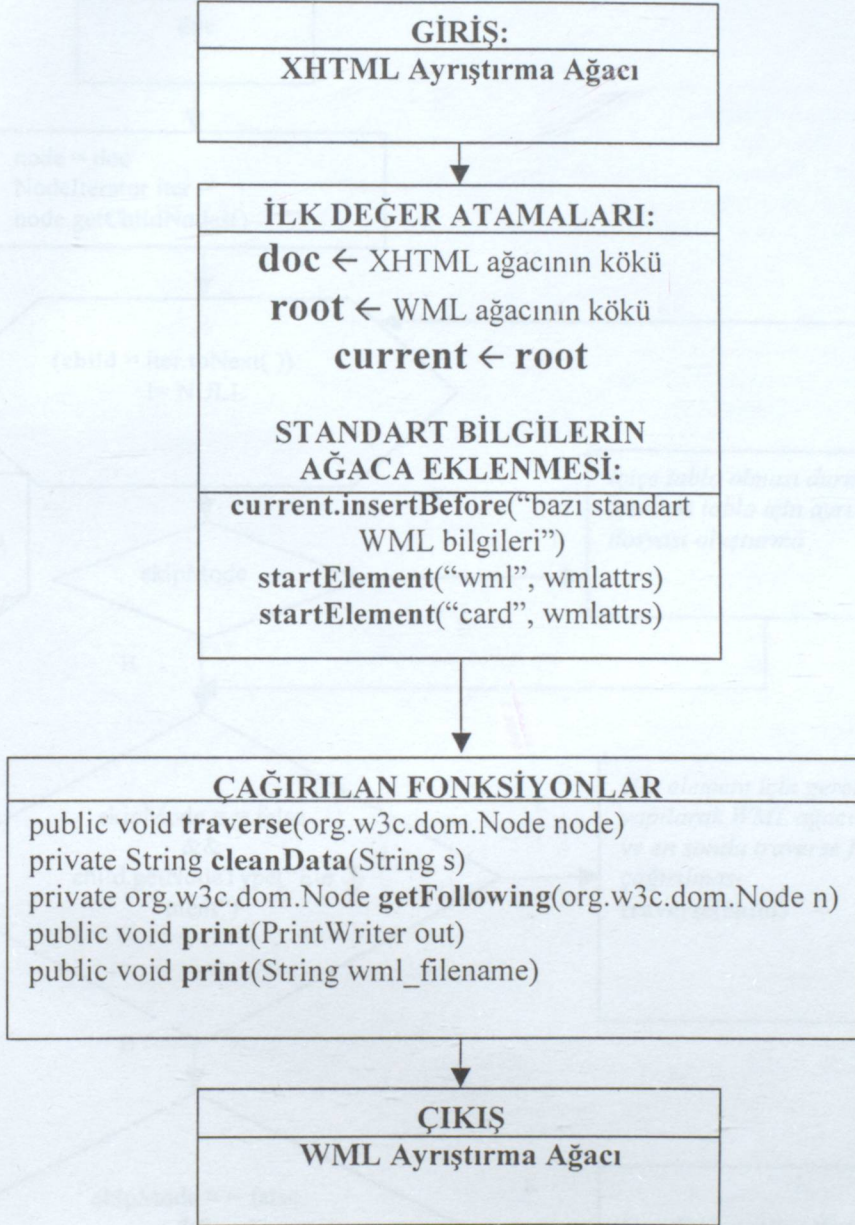
[12] <http://tidy.sourceforge.net> (HTML Tidy Library Project)

[13] <http://www.ietf.org/rfc/rfc2396.txt> ("Uniform Resource Identifiers" RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs.)

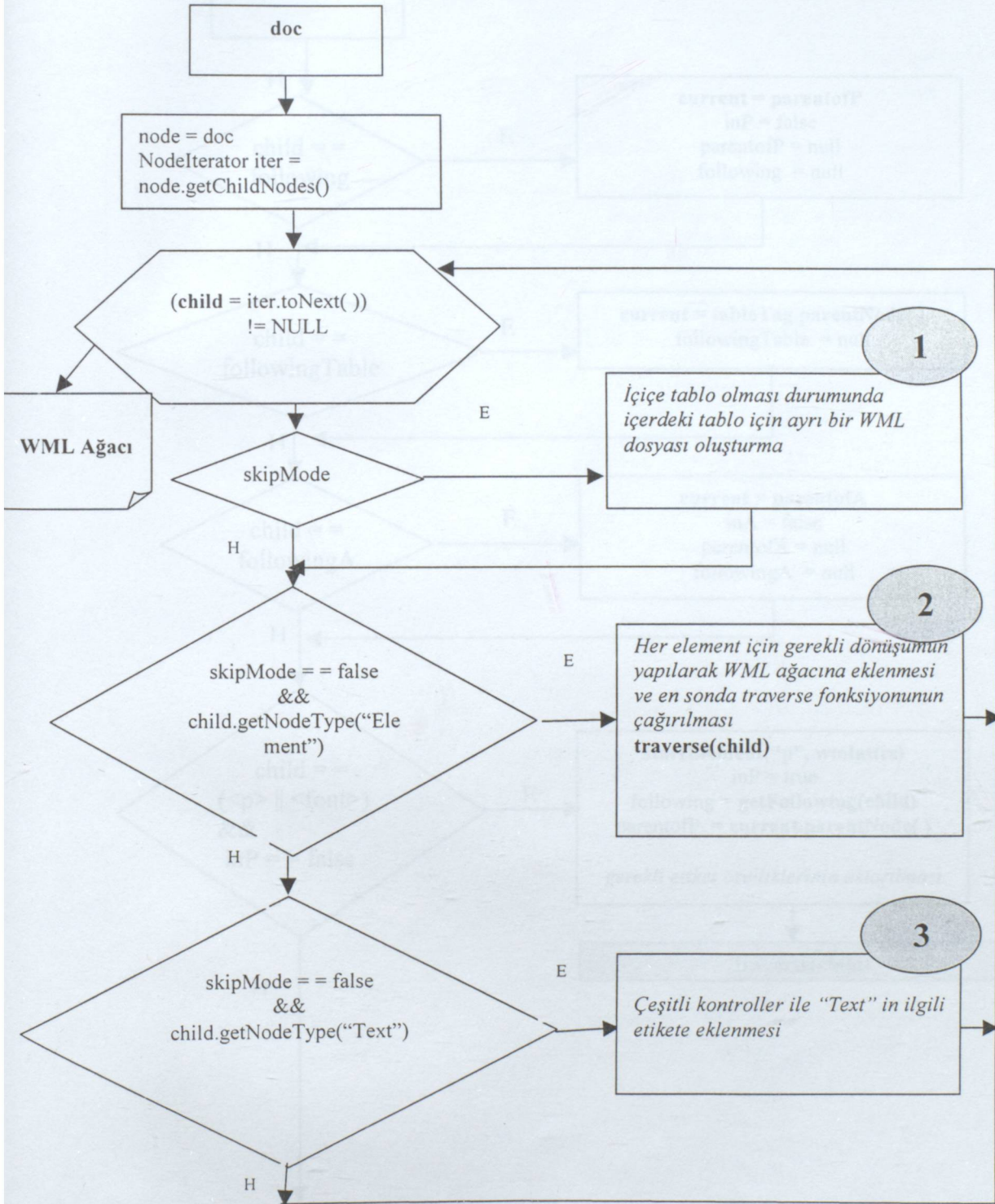
Ek 1 Sistemin genel akış diyagramı



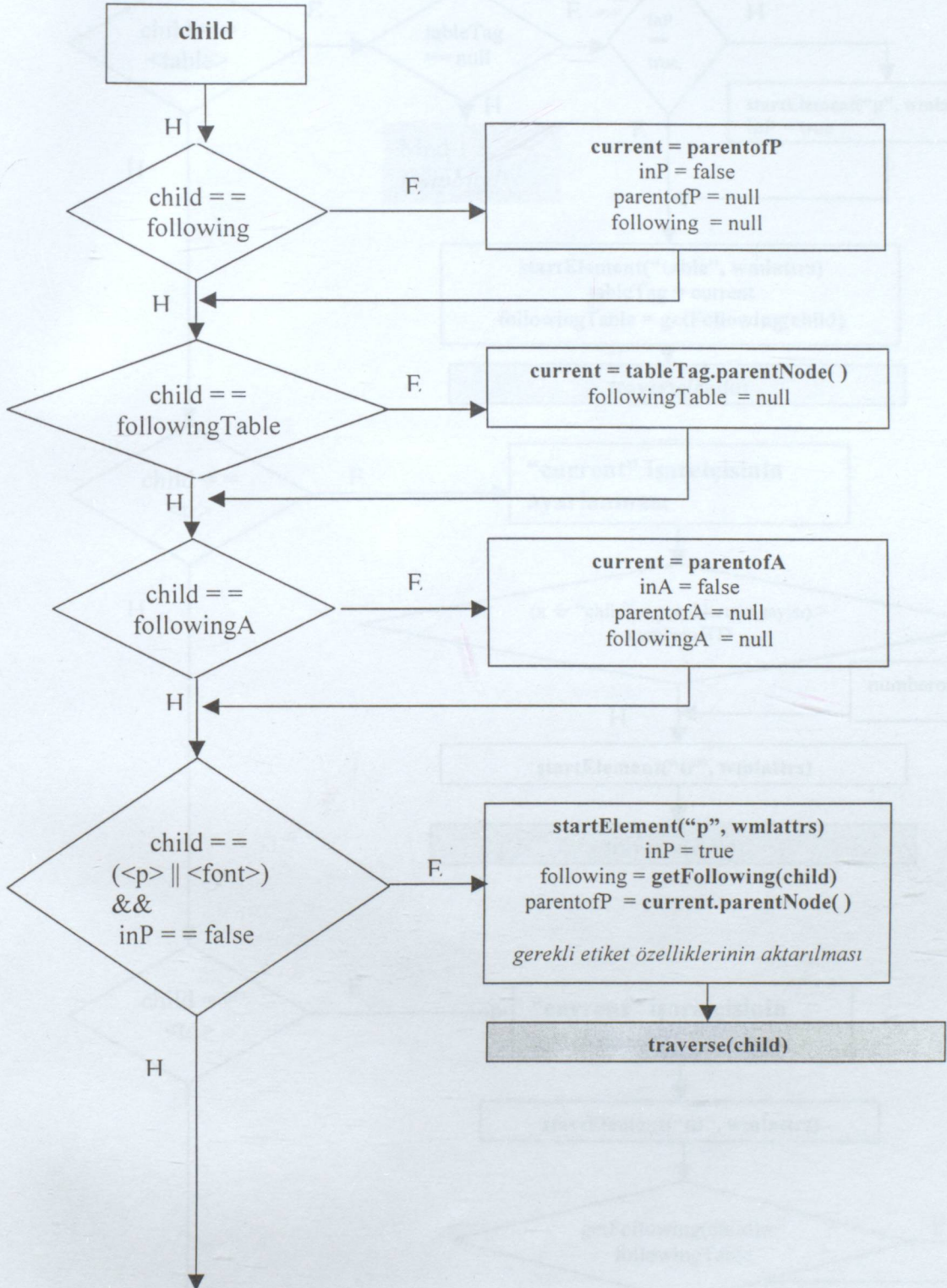
Ek 2 XHTML/WML arayüzünün genel akış diyagramı

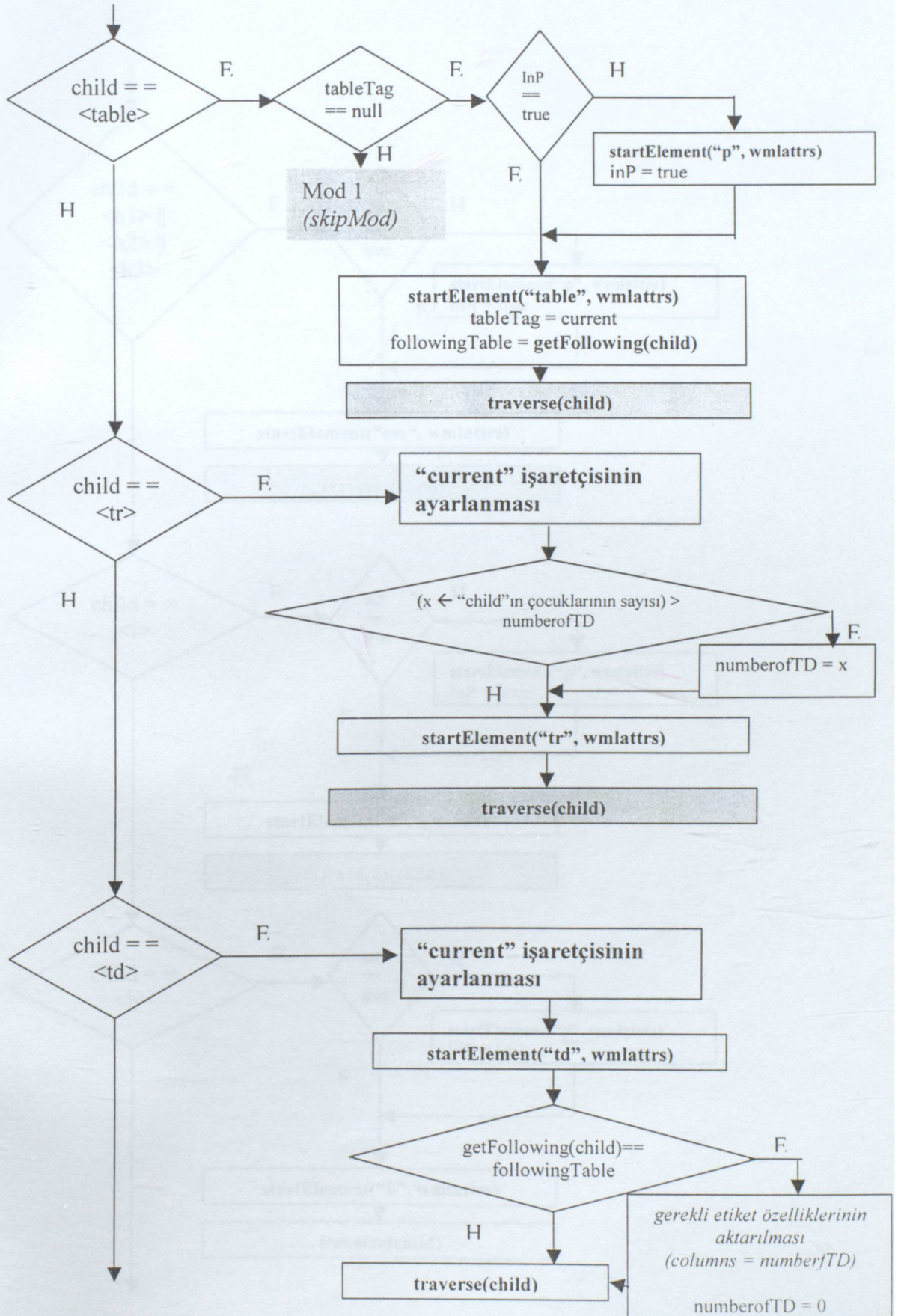


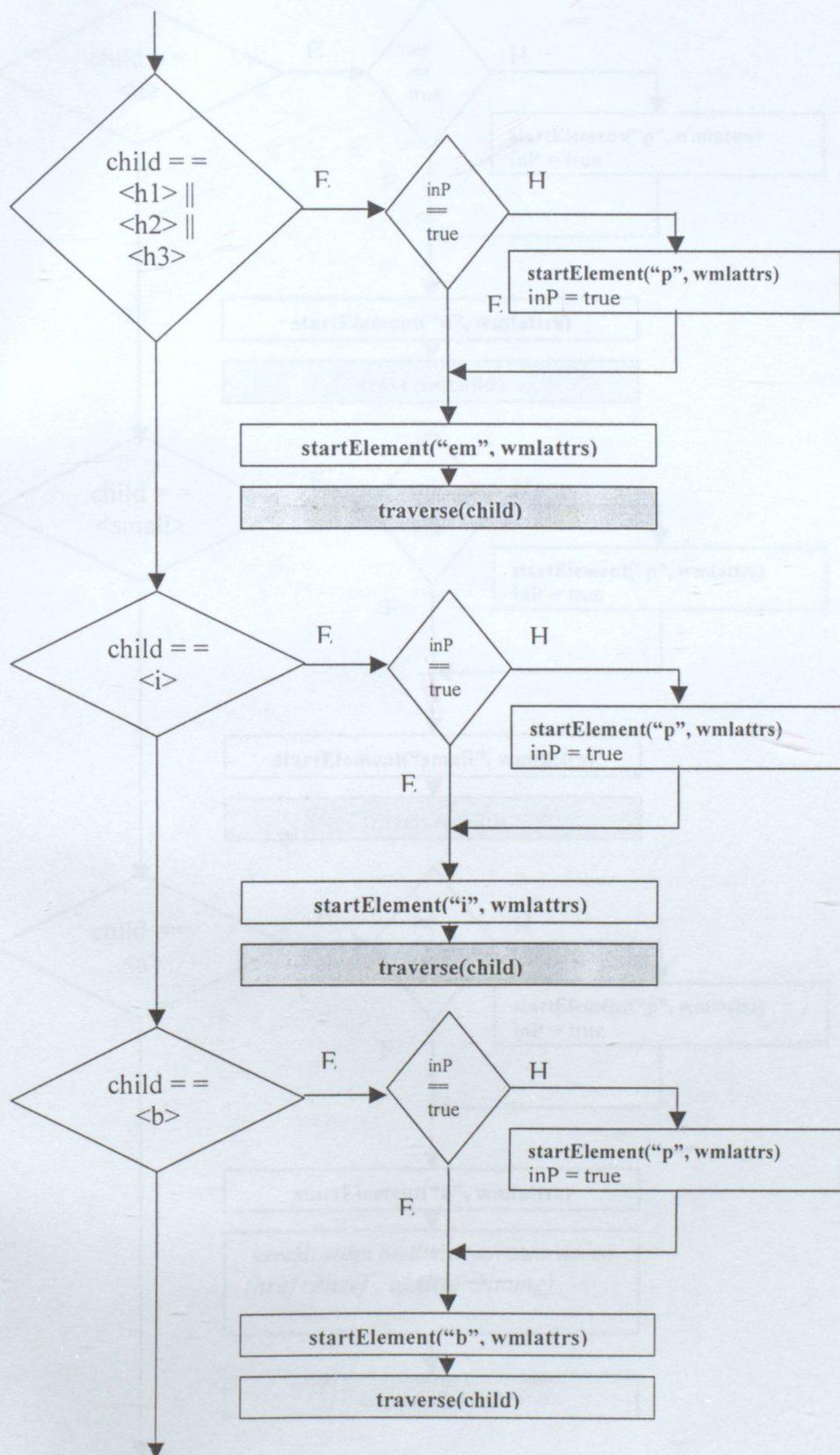
Ek 3 traverse(org.w3c.dom.Node node) rekürsif fonksiyonunun akış diyagramı

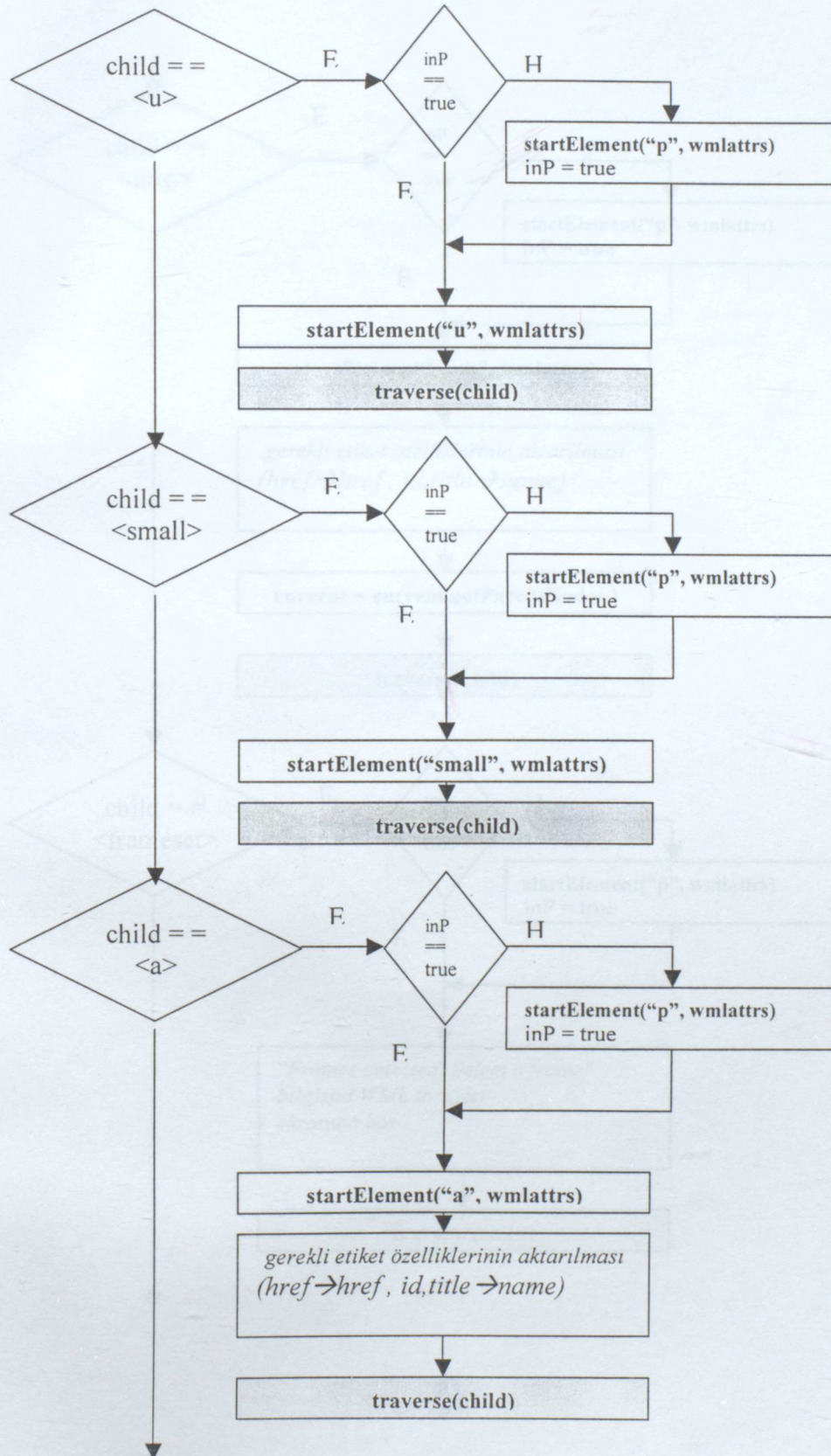


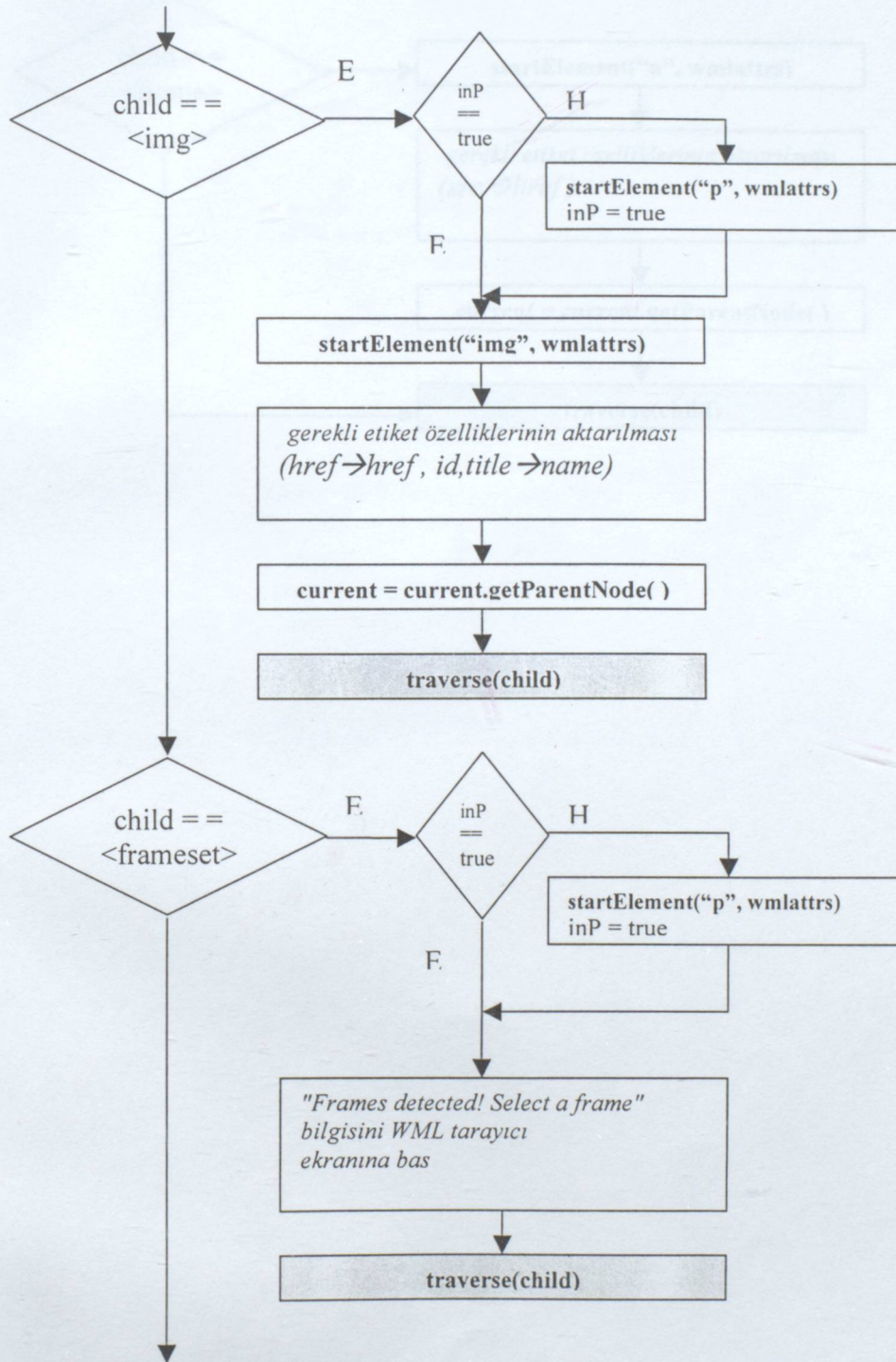
Ek 4 Etiket dönüşüm modü akış diyagramı



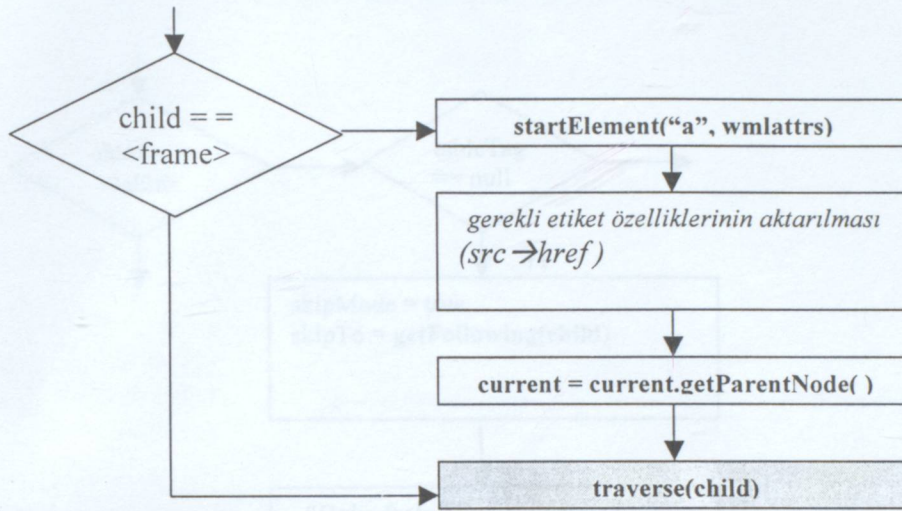




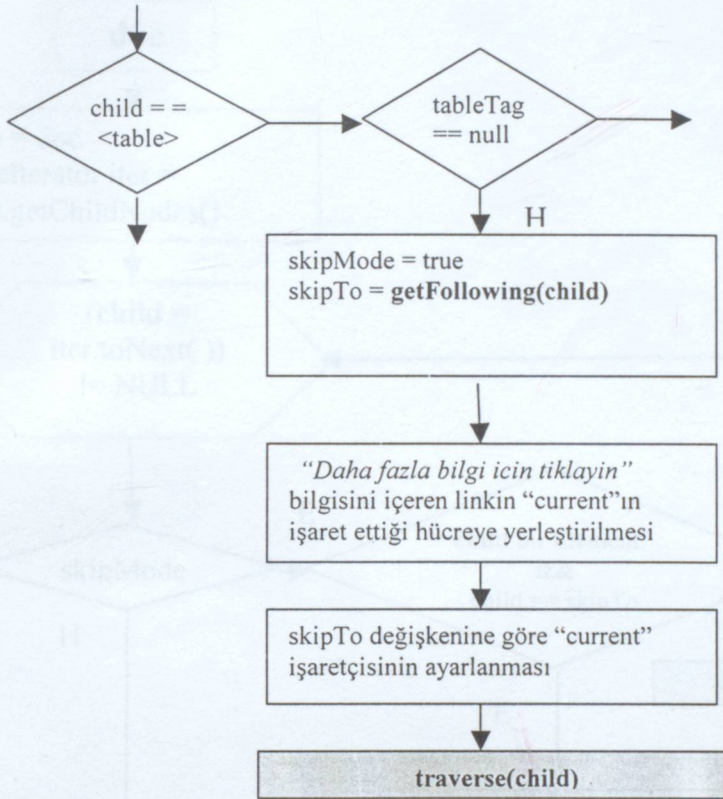




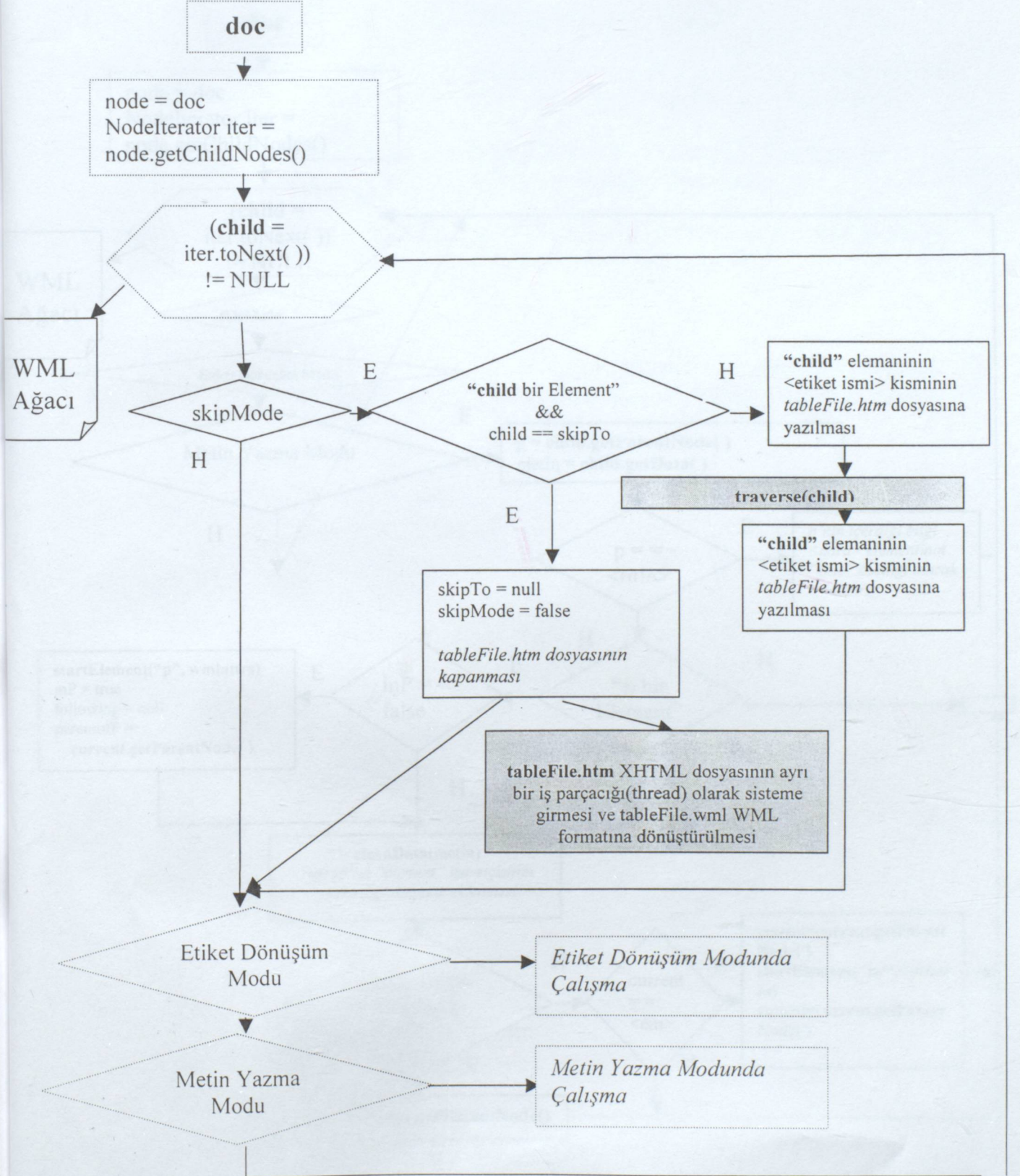
Şekil 3 İkinci moda giriş akış diyagramı



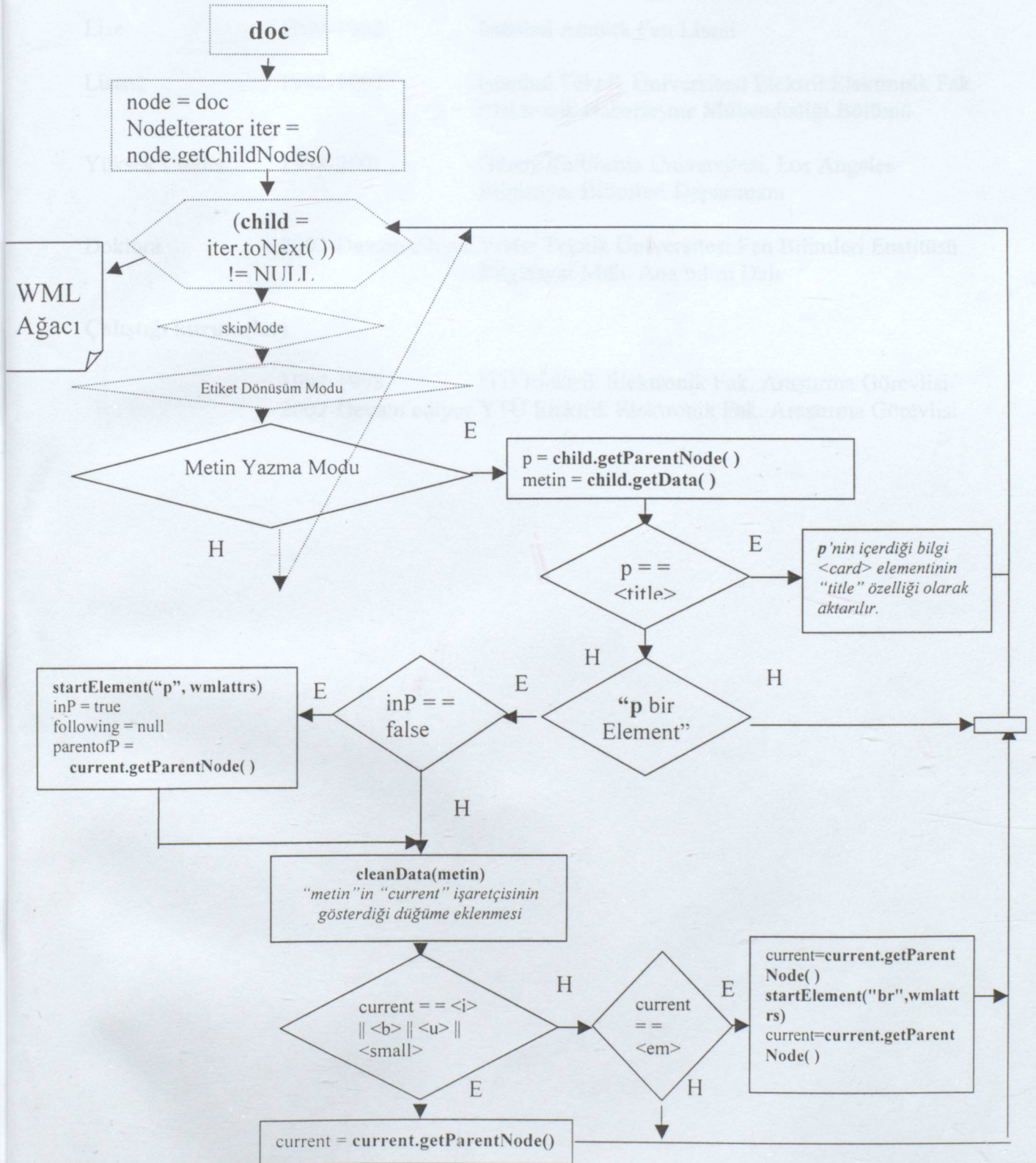
Ek 5 İkinci moda giriş akış diyagramı



Ek 6 İkinci modda çalışma akış diyagramı



Ek 7 Metin yazma modu akış diyagramı



ÖZGEÇMİŞ

Doğum tarihi	18.12.1975	
Doğum yeri	Bursa	
Lise	1989-1992	İstanbul Atatürk Fen Lisesi
Lisans	1992-1997	İstanbul Teknik Üniversitesi ElektrikElektronik Fak. Elektronik Haberleşme Mühendisliği Bölümü
Yüksek Lisans	1998-2001	Güney Kalifornia Üniversitesi, Los Angeles Bilgisayar Bilimleri Departmanı
Doktora	2002-Devam ediyor	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Müh. Ana bilim Dalı

Çalıştığı kurum(lar)

1997-1998	İTÜ Elektrik Elektronik Fak. Araştırma Görevlisi
2002-Devam ediyor	YTÜ Elektrik Elektronik Fak. Araştırma Görevlisi

