İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BİLİŞİM HUKUKUNDA KAYNAK KOD İNTİHALİ

Zeki ÖZEN

Enformatik Anabilim Dalı

Enformatik Programı

Danışman

Doç.Dr. Sevinç GÜLSEÇEN

Ocak, 2012

İSTANBUL

İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BİLİŞİM HUKUKUNDA KAYNAK KOD İNTİHALİ

Zeki ÖZEN

Enformatik Anabilim Dalı

Enformatik Programı

Danışman

Doç.Dr. Sevinç GÜLSEÇEN

Ocak, 2012

İSTANBUL

Bu çalışma 11/01/2012 tarihinde aşağıdaki jüri tarafından Enformatik Anabilim Dalı Enformatik programında Yüksek Lisans Tezi olarak kabul edilmiştir.

Tez Jürisi

Doç.Dr. Sevinç GÜLSEÇEN (Danışman)
İstanbul Üniversitesi
Fen Fakültesi

Doç. Dr. Zuhal TANRIKULU
Boğaziçi Üniversitesi
Uygulamalı Bilimler Yüksek Okulu

Doç. Dr. Alp BARAY
İstanbul Üniversitesi
Mühendislik Fakültesi

Yrd. Doç. Dr. Zerrin AYVAZ REİS
İstanbul Üniversitesi
Hasan Ali Yücel Eğitim Fakültesi

Yrd. Doç. Dr. Hakan SATMAN
İstanbul Üniversitesi
İktisat Fakültesi

Bu çalışma İstanbul Üniversitesi Bilimsel Araştırma Projeleri Yürütücü Sekreterliğinin 13472 numaralı projesi ile desteklenmiştir.

ÖNSÖZ

Lisans ve yüksek lisans öğrenimim sırasında ve tez çalışmalarım boyunca gösterdiği her türlü destek ve yardımından ötürü değerli hocam ve danışmanım Doç. Dr. Sevinç GÜLSEÇEN'e en içten dileklerimle teşekkür ederim.

Tezde yadsınamaz yardımları olan ve Enformatik çalışma hayatım boyunca her zaman desteğini hissettiğim çok sevdiğim müşfik hocam Yrd. Doç. Dr. Zerrin AYVAZ REİS'e de müteşekkirim.

Tezimle adeta ikinci bir danışman gibi ilgilenen ve tezde emeği olan Dr. Çiğdem EROL'a teşekkürü bir borç biliyorum.

Tezimin hukuk kısmında yardımını ve desteğini esirgemeyen İstanbul Üniversitesi Hukuk Fakültesi Medeni Hukuk Anabilim Dalı öğretim üyesi değerli hocam Doç. Dr. Mustafa AKSU'ya, hukukî literatürle ilgili her türlü doküman isteğime ve sorularıma cevap veren Av. Ceyda AKAYDIN'a ve yine hukuk kısmıyla ilgili kaynak bulmamda yardım eden Yrd. Doç. Dr. Mete TEVETOĞLU'na yardımlarından ötürü minnettarım.

Bu tez çalışması boyunca yardımları ve özellikle hoşgörülerinden ötürü çalışma ve oda arkadaşlarım Araş. Gör. Elif KARTAL KARATAŞ ve Araş. Gör. Fatma Önay KOÇOĞLU'na teşekkür ediyorum. Ve yine yardımlarından ötürü tüm Enformatik bölümü personeline başta Selim Bey olmak üzere çok teşekkür ediyorum.

Tez çalışmalarım sırasında kendileriyle yeteri kadar ilgilenemeyişimi anlayışla karşılayan aileme de her şey için teşekkür ediyorum. İyi ki varlar.

Bu çalışmamı, geçirmiş olduğu hastalıktan dolayı ziyadesiyle üzüldüğüm babama ithaf ediyorum.

Aralık, 2011 **Zeki ÖZEN**

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

SEMBOL LİSTESİ

AB : Avrupa Birliği
ABD : Amerika Birleşik Devletleri
ASP : Active Server Pages
AST : Abstract Syntax Tree
BK : Borçlar Kanunu
BSA : Business Software Alliance
BSD : Berkeley Software Distribution
BT : Bilişim Teknolojisi
COBOL : Completely Obsolete Business Oriented Language
CSV : Comma Separated Values
DTÖ : Dünya Ticaret Örgütü
ENIAC : Electronic Numerical Integrator And Computer
FORTRAN : Formula Translation
FSEK : Fikir ve Sanat Eserleri Kanunu
GNU : Gnu Is Not Unix
GPL : Gnu Public License
GUI : Graphical User Interface
HTML : Hyper Text Markup Language
IDC : International Data Corporation
IDE : Integrated Development Environment
IT : Information Technology
JDK : Java Development Kit
JIT : Just-In-time Compilation
JRE : Java Runtime Environment
JSP : Java Server Pages
JVM : Java Virtual Machine
KHK : Kanun Hükmünde Kararname
m. : Madde
PDG : Program Dependency Graph
PHP : Personal Home Page
TCK : Türk Ceza Kanunu
TRIPS : Trade-Related Aspects Of İntellectual Property Rights
v. : Versiyon
VB : Visual Basic
WIPO : World Intellectual Property Organisation
WTO : World Trade Organization
XML : Extensible Markup Language

ÖZET

BİLİŞİM HUKUKUNDA KAYNAK KOD İNTİHALİ

Bu çalışmada yazılım sektörüne ait hukukî problemlerden biri olan kaynak kod intihalinin ne olduğu ve kaynak kod intihalinin hukuksal açıdan durumu araştırılmış, kaynak kod intihalini ve kaynak kod benzerliğini tespit etmek amacıyla geliştirilmiş teknikler ve programlar incelenmiştir.

Bu tez çalışması teknik ve hukuk kısım olarak ikiye ayrılabilir. Hukuk kısmında eser kavramı, bilgisayar programı ve bilgisayar programı unsurlarından olan kaynak kodun eser niteliği üzerinde durulmuş, intihal durumunda eser sahibinin hukuksal haklarının neler olduğu Fikir ve Sanat Eserleri Kanunu etrafında açıklanmaya çalışılmıştır.

Çalışmanın teknik kısmında kaynak kod, bilgisayar programı ve yazılım kavramları açıklanmaya çalışılmış ve kaynak kodun bilgisayar programına dönüşme yöntemleri incelenmiştir. Devamında intihalin ve kaynak kod intihalinin üzerinde durulmuş, kaynak kod benzerliği ve klon kod kavramları açıklanmaya çalışılmış, kaynak kod intihalini tespit etmek amacıyla geliştirilen teknikler ve yazılımlar incelenmiştir.

Çalışmada mahkemelere intikal etmiş birkaç kaynak kod intihali davası incelenmiş ve kaynak kod intihalini tespit eden yazılımların bu davalardaki önemine işaret edilmiştir.

Çalışmada kaynak kod intihali ve klon kod tespiti yapan dokuz yazılımın kullanımı örneklerle açıklanmaya çalışılmıştır. Çalışma, bu araçların ayrıntılı kullanımını içermesi bakımından kaynak kod intihali davalarına bakan mahkeme üyeleri ve intihali veya benzerliği tespit etmekle görevlendirilen bilirkişiler için bu anlamda büyük önem taşımaktadır.

SUMMARY

SOFTWARE SOURCE CODE PLAGIARISM IN INFORMATICS LAW

In this study, the nature of the source code plagiarism which is one of the legal problems in software business as well as the legal aspect of source code plagiarism was researched and the techniques and programs that had been developed for the purpose of detecting the source code plagiarism and source code similarity was also observed.

This study may be divided into two categories as technical and legal. In the legal portion, it is focused on product concept, computer programming and the product qualification of the source code which is one of the components of software. The rights of the legal owner of the product in case of plagiarism are also tried to be explained within the framework of Law on Intellectual and Artistic Works.

In the technical section of the study, the concepts of source code, computer programming and software and the methods of converting source code into computer programming was tried to be explained. Consequently the nature of the source code, source code plagiarism, source code similarity and clone code concepts were explained and the techniques and software that had been developed for the purpose of detecting the source code plagiarism and source code similarity was observed.

In the study, some legal cases pertaining to source code plagiarism that had been taken to the law court were studied and it was pointed out to the importance of software detecting the source code plagiarism in these cases.

In the study, the usage of nine software tools that detect source code plagiarism and clone code was tried to be explained with examples. Since this study contains the detailed usage of these tools, it is very important for the members of law court who overlook the source code plagiarism cases as well as for the experts who have been designated to detect plagiarism or the similarity.

1. GİRİŞ

Bilişim teknolojisinin (BT) gelişmesine paralel olarak bilgisayar fiyatlarının ucuzlaması ve bilgisayarın yaşamın her alanında kullanılır olmasıyla birlikte bilgisayara iş yapabilme kabiliyeti katan yazılımların da çeşitliliği ve yaygınlığı artmıştır. Artık hemen hemen her alanda bilgisayar yazılımları kullanılmakta ve üretilmektedir. Bilgisayar ve türevi akıllı cihazların yaygınlaşmasıyla donanım sektörüne paralel olarak yazılım sektörü de büyümektedir.

Yazılım sektörünün gelişmesi bir takım hukukî problemleri de beraberinde getirmiştir. Bilgisayarın tarihinin henüz yüz yılı bulmamış olmasından ötürü bilgisayar ve bilişimle ilgili suçlar ülkelerin geleneksel hukuk mevzuatları içinde kısa zaman öncesine kadar yer almıyordu. Zaman içerisinde başta gelişmiş ülkeler hem konunun önemi ve hem de zorunluluktan ötürü yasalarında bilgisayar programlarına yönelik tanımlamalar ve kanunlar çıkarmışlardır.

Bilgisayar programlarının korunmasına yönelik temelde iki farklı bakış açısı vardır. Bunlardan ilki bilgisayar programlarını bir fikir ürünü olarak kabul edip korumayı fikrî mülkiyet kanunları etrafında sağlamak, bir diğeri de bilgisayar programlarını patentlenebilir ürünler arasında kabul edip korumayı patent kanunları ile sağlamaktır. Avrupa Birliğinin üye ülkelere gönderdiği ve tezin devamında Avrupa Konseyi Direktifi ile anılacak olan 14 Mayıs 1991 tarihli 91/250/EEC numaralı "Bilgisayar Programlarının Hukukî Korunmasına İlişkin Yönergesi" ile bilgisayar programlarının üye ülkeler nezdinde fikir ve sanat eserleri hukuku kapsamında ve Bern Sözleşmesinin kastettiği anlamda edebi eser olarak korunması esas almıştır (Aksu, 2006; EU, 2011).

Her ne kadar ülkemizde bilgisayar programları için ayrı bir kanun düzenlenmiş olmasa da bilgisayar programları ülkemizde de bir fikir ürünü değerlendirilmekte ve bilgisayar programlarına yönelik koruma tezin devamında "Kanun" şeklinde de zikredilecek olan Fikir ve Sanat Eserleri Kanunu (FSEK) ile sağlanmaktadır. Bunu göz önünde

bulundurarak tezde bilgisayar programları üzerindeki haklar FSEK çerçevesinde incelenecektir.

FSEK ile bilgisayar programlarına sağlanan koruma diğer eser türlerinden farksızdır. Aynı bağlamda bilgisayar programlarının eser sahiplerine verilen haklar da diğer eser sahiplerine verilen haklarla aynıdır. Bununla birlikte bilgisayar programlarının unsurları, bunlara sağlanan koruma ve bilgisayar programlarının tersine mühendislik yöntemleri ile ayrıştırılması gibi bilgisayar programlarına has konular diğer eser türlerinden farklı olarak kanunda ayrıca ele alınmış ve açıklanmıştır.

Tezin ilerleyen bölümlerinde irdeleneceği üzere kaynak kod, FSEK ile koruma altına alınan bilgisayar programı unsurlarından biridir. Tezin adında fikrî mülkiyet hukuku yerine Bilişim Hukuku terimi kullanmamızın sebebi şu anda olmasa da gelecekte bilgisayar programları ve bilişimle ilgili diğer hak ve tanımlamaların ayrı bir kanunla anılması gerektiği içindir. Hâlihazırda bilişim suçları ile ilgili 26.09.2004 tarihinde kabul edilen 5237 Sayılı Türk Ceza Kanunu 12.10.2004 tarih 25611 sayılı Resmi Gazete ile yürürlüğe giren bir düzenleme yapılmıştır. Kanunun 10.bölümü "Bilişim Alanında Suçlar" başlığını taşımaktadır (Eralp, 2011). Bu kanun Bilişim ile her hangi bir tanımlama yapmamakta sadece suç tanımlamalarını ve cezaları belirlemektedir. Oysaki bilgisayar programı, yazılım, internet, bilişim ve bilişimle ilgili tüm hakların, tanımlamaların, bunlarla ilgili veya bunlarla işlenen suçların genel kanunlar ve suçlar ile değil başlı başına ayrı bir hukuk siteminde ele alınması gerekmektedir. Almanya örneğinde olduğu gibi kimi ülkeler bu konuyu ayrı bir hukuk çerçevesinde ele almaktadırlar.

Bu tez çalışmasının amacı kaynak kodların hukukî durumunu ve eser sahiplerinin haklarını FSEK üzerinden değerlendirmek, olası kaynak kod intihal durumlarında kanunun eser sahibine verdiği hakları incelemek, kaynak kod intihalini ve klonlanmış kodları tespit eden yazılımların kullanımı hakkında bilgi vermek ve mahkeme heyetine ve görevli bilirkişilere kaynak kod intihali olduğundan şüphelenilen davalarda teknik olarak izlenmesi gereken yol hakkında neler yapılabileceğini açıklamaktır.

Tezin “Genel Kısımlar” bölümünde tez çalışması boyunca irdelenecek olan kaynak kod, bilgisayar programı ve yazılım hakkında teknik bilgiler verilmiştir. Devamında ise bu terimlerin FSEK çerçevesinde hukukî durumları incelenmiş, bilgisayar programı ve kaynak kodun FSEK’e göre eser olma ve korunma durumları üzerinde durulmuştur. Olası kaynak kod intihali durumlarında Kanunun kaynak kodun eser sahiplerine tanıdığı haklar açıklanmıştır. Devamında da intihalin tanımı üzerinde durulmuş, kaynak kod intihali, kaynak kod benzerliği, klon kod, klon kodun avantajları ve dezavantajları, kod klonlama tipleri ve klon kod sınıfları gibi konular literatür taraması yapılarak açıklanmaya çalışılmıştır.

Bilgisayar programı ve yazılım kavramlarını ifade etmek için tezde yerine göre “bilgisayar programı” ve yazılım terimleri kullanılmıştır. Bununla birlikte “yazılım” kavramı “bilgisayar programı” kavramını da içermesinden ötürü kastedilen anlam birbirinin yerine kullanılabilir. “Programcı” ifadesinin de bilgisayar programının eser olarak sahibi veya yerine göre program geliştirici, yazılımcı ve yazılım mühendisi gibi bilgisayar programı geliştiren kişilerin tümünü ifade ettiği kabul edilecektir.

Çalışmanın “Malzeme ve Yöntem” kısmında kaynak kod benzerliği ve klon kod tespitine yönelik geliştirilen yazılımların incelemesi yapılmış bu yazılımların kullanımı hakkında bilgi verilmiştir. Ayrıca mahkemelere intikal etmiş iki kaynak kod intihal davası incelenerek bilirkişilerin kaynak kod intihal tespit yazılımı kullanıp kullanmadıkları araştırılmıştır.

Bulgular kısmında kaynak kod intihal tespit yazılımlarının kullanımından elde edilen sonuçlar karşılaştırılmıştır. İncelenen davalarda benzerlik tespit aracı kullanılmadan klon kodu tespit etmenin zorluğu üzerinde durulmuştur.

Tezin Tartışma ve sonuç kısmında tezin konusu tezde araştırılan literatür kapsamında ve kaynak kod intihal tespit yazılımlarının kullanımından elde edilen bulgular doğrultusunda tartışılmıştır.

2. GENEL KISIMLAR

2.1. YAZILIM, BİLGİSAYAR PROGRAMI VE KAYNAK KOD

2.1.1. Yazılım ve Bilgisayar Programı

Bilgisayarlar kendi başlarına iş yapamayan ve insan tarafından yönlendirilen makinelerdir. Bilgisayarlara iş yaptırmak için onlara bir takım komut dizileri verilir. Bu komut dizisine "bilgisayar programı" denir (Çölkesen, 2002). Bilgisayar programı kendi içinde bütünlük taşıyan ve belirli bir görevi yerine getiren algoritmik bir ifadedir (Erdoğan, 2005). Bilgisayar programı bir problemin çözümüne yardımcı olmak veya belirli bir amacı yerine getirmek için bir programlama diliyle yazılmış olan komutlar bütünü olarak da tanımlanabilir.

Yazılım (software) ise bilgisayar programları, veri ve belgeler kümesi olarak adlandırılabilir (Sarıdoğan, 2008). Yazılım bu anlamda bilgisayar programından daha kapsamlı bir terimdir. Yazılım aynı zamanda bilgisayarın donanımını kontrol eden, işleten ve ona neyi, ne zaman ve nasıl yapması gerektiğini bildiren bilgisayarın donanımından ayrı diğer bileşeni olarak da tanımlanabilir (ESL, 2001). Yazılım kelimesi, bilgisayarın donanımsal olmayan kısmını özellikle bilgisayara istenen görevlerin yaptırılmasını sağlayan programları ifade etmek için kullanılmıştır (Rosen, 2003).

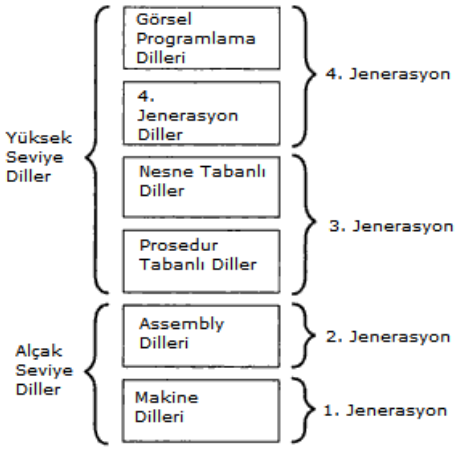
Yazılım üretimi yoğun emek isteyen bir aktivitedir. Kaynak kodun olduğu kısımlar özellikle test, hata ayıklama ve programın işleyişini doğrulama zaman ister (Henderson, 2009:276). Bir bilgisayar programının veya daha geniş kapsamda yazılımın geliştirme aşamasından çalıştırıldığı aşamaya kadar birçok biçimi vardır. Bunlar sırasıyla geliştirme aşamasında kaynak kod (*source code*), derleme veya yorumlama aşamasında amaç kod (nesne kod da denir) (*object code*), çalıştırma aşamasında ise çalıştırılabilir kod (*executable code*) olarak isimlendirilir (Yadav, 2008).

2.1.2. Programlama Dili

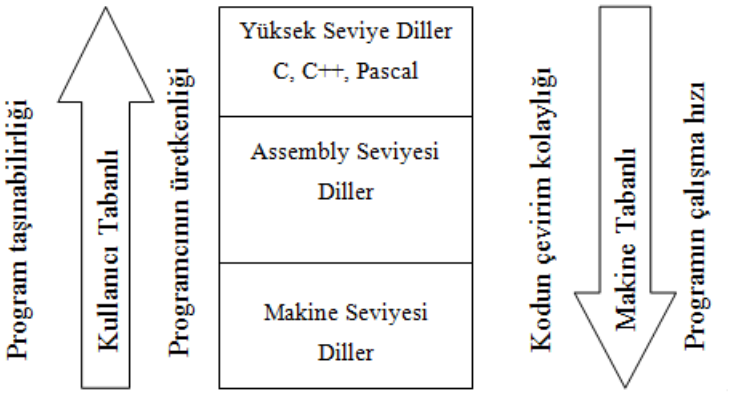
Programlama dilini, bilgisayarın anlayacağı komutları insan diline ve mantığına yakın kelime ve kurallar barındıran yapılarla ifade etmek için tasarlanmış, kendi içinde kurallı, kendine has grameri ve yazım biçimi olan yapay bir dil şeklinde tanımlamak mümkündür (Tucker ve Noonan, 2002; Henderson, 2009).

Programlama dili bilgisayarların gelişimi ve tarihi birlikte değerlendirilerek açıklanabilir. Bilgisayarlar, daha önce bahsedildiği gibi fiziksel parçalardan oluşan donanım ve bunların kullanılmasını sağlayan yazılım olmak üzere iki kısımdan oluşmaktadır. Bilgisayarlar ikili sayı sistemi (*binary system*) denilen sadece 0 ve 1'lerden oluşan komut dizilerini (makine dili) işleyerek çalışan makinelerdir (Blundell, 2007). Örneğin ilk bilgisayarlardan oluşan ENIAC (Electronic Numerical Integrator And Computer), kendisine verilen makine komutlarını işleyerek çalışmaktaydı (Henderson, 2009:388). En düşük programlama dili seviyesi olan ve makine dili olarak da adlandırılan bu komut setleri, insanın anlaması ve işlemesi epey zor olan komut dizilerinden oluşmaktadır. Daha sonra makinenin işleyeceği komut dizileri hatırlanması kolay bir takım kısaltma ve sembollerle (*mnemonic*) ilişkilendirilerek bilgisayarlar bu kelimelerle programlanmaya başlandı. Örneğin toplama (addition) için ADD, karşılaştırma (compare) için CMP gibi komutlar tanımlanarak bu işi yapan makine komutları bu kelimelerle ilişkilendirildi. "Assembly dili" olarak da bilinen bu dil kısa sembol ve kısaltmalardan oluşmaktadır. Programlama dilleri seviye olarak sınıflandırıldığında ara seviyede yer alan bu dil ile programlama yapmak makine diline göre daha kolay olsa da pek kullanışlı ve verimli değildir. Bu zorluğu aşmak için insanın algısına ve konuşma diline daha yakın, kendine has kurallı yapıları olan ve "yüksek seviye diller" olarak da tanımlanan programlama dilleri geliştirilmeye başlanmıştır (Feller ve Fitzgerald, 2002; Henderson, 2009:388).

Şekil 2.1'de programlama dilleri seviyelerine ve içinde yer aldığı jenerasyona göre sınıflandırılması verilmiştir.



Şekil 2.1: Programlama dillerinin seviyelerine göre sınıflandırılması (Bronson ve Rosenthal, 2005)



Şekil 2.2: Programlama dillerinin programcıya sağladığı avantajlar (Venkataram, 2005)'den uyarlanmıştır.

Şekil 2.2'de programlama dillerinin insan algısına göre basitten zora doğru sınıflandırılması gösterilmiştir. Programlama dili seviyeleri makine diline yaklaştıkça programlama zorluğu artar, komutların çalışma süreleri azalır ve program daha hızlı çalışır; insana yaklaştıkça ise programcının üretimi artar ve program geliştirme süreçleri hızlanır (Venkataram, 2005).

2.1.2.1. Makine Dili

Bilgisayarın işlem yapma birimi olan mikroişlemciler (*microprocessor*) sadece 1 ve 0'lardan oluşan komut dizisini okur, anlar ve çalıştırırlar (Bloss, 2003; Blundell, 2007). Makine dili, bilgisayarın mikroişlemcilerinin doğrudan işleyeceği çok kesin bir dizi

talimatlar ve işlemlerdir (Davidson, 2003; Goel, 2010). Makine dili komutları, işlemcinin kendi dili olduğu için her hangi bir çevrim işlemine gerek yoktur ve bu nedenle diğer programlama dillerine göre çok hızlı çalışırlar (Goel, 2010). Makine dilinde tüm veriler ve program referansları bilgisayar hafızasının içindeki asıl adreslere göre belirtilir. Makine dili komutları insan algısına uzak komutlardır bu nedenle sıkıcıdırlar. Bu komutlar neredeyse mikroskobik detaylara sahiptir ve düzeltilmesi de bir o kadar zordur (Davidson, 2003).

Programlama dilleri insan algısına yakınlık durumuna göre sınıflandırıldığında makine dili insana en uzak seviye dil (düşük seviye) olarak bilinir.

2.1.2.2. Assembly dili

İnsana zorluk açısından makine dilinden sonra "Assembly dili" gelir. Assembly dili, üzerinde çalışması ve insanlar tarafından anlaşılması zor olan makine dili kodları yerine insanlar tarafından anlaşılarak programlanması daha kolay olan alfabetik ifadelerin (*mnemonic*) kullanıldığı düşük seviyede programlama için bir ortam oluşturur. Assembly dili, makine dilinin insanın okuyabileceği bir formu olarak düşünülebilir (Aksoy ve DeNardis, 2007). Assembly dili, makine dili ve kolaylık açısından insana daha yakın olan yüksek seviye diller arasında yer alan düşük seviye ara bir dildir (Bronson ve Rosenthal, 2005). Her bir Assembly komutu "Assembler" adı verilen çeviriciler tarafından makine diline çevrilir (Goel, 2010).

Assembly dili donanıma bağımlıdır, yani Assembly komutları arasında her bir işlemci serisi donanımına göre farklılıklar vardır (Noergaard, 2010). Bu nedenle hangi işlemci için kod yazılacaksa o Assembly dilinin çeviricisi Assembler gerekmektedir.

2.1.2.3. Yüksek Seviye Diller

Assembly dilinin programcılar için pek kullanışlı olmaması nedeniyle insanın algılamasına daha yakın, kolay hatırlanabilir ve kullanımı kolay olan yüksek seviye programlama dilleri tercih edilmektedir. Bu nedenle kaynak kodlar genellikle insana yakın yüksek seviye programlama dillerinden biriyle yazılır (Bloss, 2003).

Yüksek seviye dilleri öğrenmek, bu dillerde program geliştirmek ve hata ayıklamak Assembly dili ve makine diline göre çok daha kolaydır. Bu sayede program yazması daha az zaman almakta, programcının üretkenliği artmaktadır (Venkataram, 2005).

2.1.3. Kaynak Kod

Bilgisayar programının programcı tarafından bir programlama diliyle yazılan formuna kaynak kod denir (Yadav, 2008). Kaynak kod (aynı zamanda kaynak, kod ya da kaynak program da denir) aslen bir insan tarafından düz metin şeklinde (okunabilir alfanumerik karakterle) ve bir programlama diliyle yazılmış ve o dile ait komutlar, ifadeler ve bunların komut setlerini barındıran yazılımın bir biçimidir (LINFO, 2006).

Başka bir çalışmada kaynak kod, bir veya daha fazla programlama diliyle bilgisayarda yazılmış ve makine diliyle alakasız olarak bilgisayara verilen kodlar olarak tanımlanmıştır (Davidson, 2003).

Kaynak kodun bir diğer tanımı da, bir bilgisayar programcısının bir metin (text) editörle veya görsel programlama araçları ile oluşturduğu ve bir dosyaya kaydettiği programlama ifadeleridir. Örneğin bir bilgisayar programcısı bir programı C dilinde yazmayı kararlaştırıp bir metin editörü kullanarak birbiri ardına gelen C diline ait ifadelerle yazdıklarını bir isim vererek bir dosyaya kaydeder. İşte bu dosyanın kaynak kod içerdiği söylenir (Kudyba ve Diwan, 2002).

Kaynak koda ilişkin bir başka tarif de şu şekildedir: Bir bilgisayar programının bilgisayarda çalıştırılmaya hazır durumdan önceki (derlenmeden önceki) aşamasına kaynak kod, sonraki aşamasına ise amaç kod da denilmektedir (Kudyba ve Diwan, 2002).

Programlama dilinin tanımından yola çıkarak kaynak kodu, programlama dillerinden biriyle ve istenilen amaca uygun olarak o dilin gramer yapısı ve komutları kullanılarak yazılan komutlar bütünü olarak tanımlamak da mümkündür.

Program üretim sürecinde analiz aşamasından sonra program kaynak kodunun yazıldığı kodlama aşaması gelmektedir. Kaynak kod yazılmış olduğu programlama diline ait

komutları, değişkenleri, sabitleri, değişkenlerin değerlerini, karşılaştırma ve kontrol yapılarını, döngü yapılarını, fonksiyon tanımlamalarını, fonksiyon çağrılarını, matematiksel operatörleri, karşılaştırma ve kontrol operatörlerini, kullanılan dil destekliyorsa sınıf ve miras yapılarını, programda dikkate alınmayan yorum (açıklama) satırlarını ve dile ait diğer yapıları barındırır (Freedman, 2001; Mishne, 2003; Tech Terms Dictionary, 2007; Zeidman, 2008).

Kaynak kod, Windows'ta "*Notepad*", Linux işletim sistemlerinde "*vi*" gibi basit metin editörleriyle yazılabileceği gibi kaynak kodda hata kontrolünü yapan ve kaynak kodun çalıştırılabilir koda dönüşümünü gerçekleştiren derleyici veya yorumlayıcı barındıran Eclipse, Netbeans gibi "bütünleşik geliştirme ortamları" (IDE: Integrated Development Environment) ile de yazılabilir (LINFO, 2006).

Bir programın kaynak kodunun tamamının aynı programlama diliyle yazılması gerekmez (LINFO, 2006). Kaynak kod, farklı programlama dilleriyle yazılmış kısımlardan oluşabilir. Örneğin esasında C diliyle yazılan Linux çekirdeğinin (kernel) içinde assembly kodları da vardır.

Bir program sadece bir kaynak kod dosyasından oluşmayabilir. Küçük programlar bir veya birkaç kaynak kod dosyası içeriyor olabilirler. Öte yandan büyük programlar yüzlerce hatta binlerce büyük kaynak kod dosyasına sahiptirler. Programın birden fazla kaynak dosyaya sahip olması, programın farklı bölümlerini organize etmeye yardımcı olur (Tech Terms Dictionary, 2007). Bir kaynak kod dosyasının içinde birden fazla fonksiyon olabilir. Programda farklı işlev gören tüm kodları bir kaynak kod dosyası içinde birleştirmek, programın, programcı tarafından okunurluğunu azaltır, geliştirilmesini ve hata ayıklamasını zorlaştırır. Programda, programın genelinden bağımsız kendine has özel bir işlevi olan fonksiyonlar, modüller, kod blokları, veritabanı bağlantısını sağlayan kısımlar farklı kaynak kod dosyalarına ayrılarak programın yazılması ve hatalarının düzeltilmesi kolaylaştırılmış olur.

Bir programın yazımında ne kadar çok kaynak kodu olursa olsun, programın çalışır formunda o kadar çok dosya olmayabilir. Bunun sebebi kaynak kod dosyalarının bilgisayarın anlayacağı forma çevrildiğinde bir uygulamanın veya bir programın içinde

birleştirilmesidir (LINFO, 2006; Tech Terms Dictionary, 2007). Bu sayede programın çalıştırılabilir formunun bilgisayarda kapladığı alan kaynak kod dosyalarının bilgisayarda kapladığı alana göre azalır. Örneğin Linux çekirdeğinin 2.6.39 versiyonunun kaynak kodu yaklaşık 14.6 milyon satır içermekte ve 473 MB yer kaplamaktadır (Leemhuis, 2011). Buna karşın derleme sonrası oluşan Linux çekirdeğinin ve başlık (header) dosyasının ikili formunun kapladığı alan kaynak kod dosyalarının büyüklüğüne göre çok daha azdır.

Kaynak kod, derlendikten sonra amaç kod oluşur ve programın çalışması için gerekli kütüphane dosyaları amaç koda adreslenir (Bronson ve Rosenthal, 2005). Bu işleme linkleme (linking) denmektedir. Bu aşamadan sonra çalıştırılabilir kod yani program üretilmiş olur. Kaynak kodun amaç kod haline gelmesi tek yönlü bir süreçtir (St.Amant, 2007; Veselosky, 2011). Kaynak kod ile her zaman amaç kod elde edilebilir fakat amaç koddan kaynak kodun kendisine ulaşmak teorik olarak mümkün değildir. Çeşitli tersine mühendislik (*reverse engineering*) yöntemleri ve araçları ile amaç koddan kaynak koda çevrim yapılsa da elde edilen kodu yorumlamak anlamlı değildir ve programın çalışma mantığını bu koda bakarak anlamak çok zordur. Bu nedenle programcılar, bir programı geliştirmek, değiştirmek veya hatalarını gidermek için o programın kaynak koduna ihtiyaç duyarlar (Kudyba ve Diwan, 2002). Bu sayede programın bir üst versiyonunu çıkarmak mümkün olabilmektedir.

Kaynak kod yazılımlarla birlikte genelde verilmez. Yazılım, genellikle programın çalıştırılmaya hazır derlenmiş formunda satılır veya yayımlanır (Kudyba ve Diwan, 2002; LINFO, 2006). Bunun nedeni kaynak kod veriliğinde programcıların bu kaynak kodu geliştirip ücretsiz dağıtması kaygısıdır. Böyle bir durum şüphesiz ki, yazılım şirketlerinin iş yapamamasına sebep olacaktır. Kaynak kod yerine amaç kodun verilmesi, sıradan kullanıcıların programı kurmasını kolaylaştırmakta ve onları programı derleme zahmetinden kurtarmaktadır (LINFO, 2006).

Öte yandan kaynak kodun verilmemesi, yazılımda tekel firmaların oluşmasına sebep olmakta veya programın çalıştırıldığı bilgisayarda kişisel verilerin bir başka ortama iletilip iletilmediği endişesini beraberinde getirmektedir. Bu gibi sebeplerden ötürü

dünyada Açık Kaynak Kod (Open Source Code) ve Özgür Yazılım (Free Software) kullanımı giderek yaygınlaşmaktadır.

Kaynak kod, program üretmeyi sağladığı gibi başka kullanım amaçları da mevcuttur. Kaynak kodun tercih edilme sebeplerinden biri de gelişmiş kullanıcıların veya sistem yöneticilerinin programı kaynak koddan derleyip kurmak istemeleridir (LINFO, 2006). Derleme sırasında sistemin özelliklerine ve platformun tipine göre çeşitli opsiyonlar bulunmaktadır ve böylece derleme sonrası üretilen program sisteme daha uygun ve daha optimize çalışacaktır.

Kaynak kod aynı zamanda programcılar için iyi bir öğrenme materyalidir. Kaynak kod dosyalarındaki algoritmalar, metodolojiler ve diğer programlama teknikleri programcıların yeni teknikler öğrenmesini sağlar (Spinellis, 2003; LINFO, 2006; Wilson, 2011).

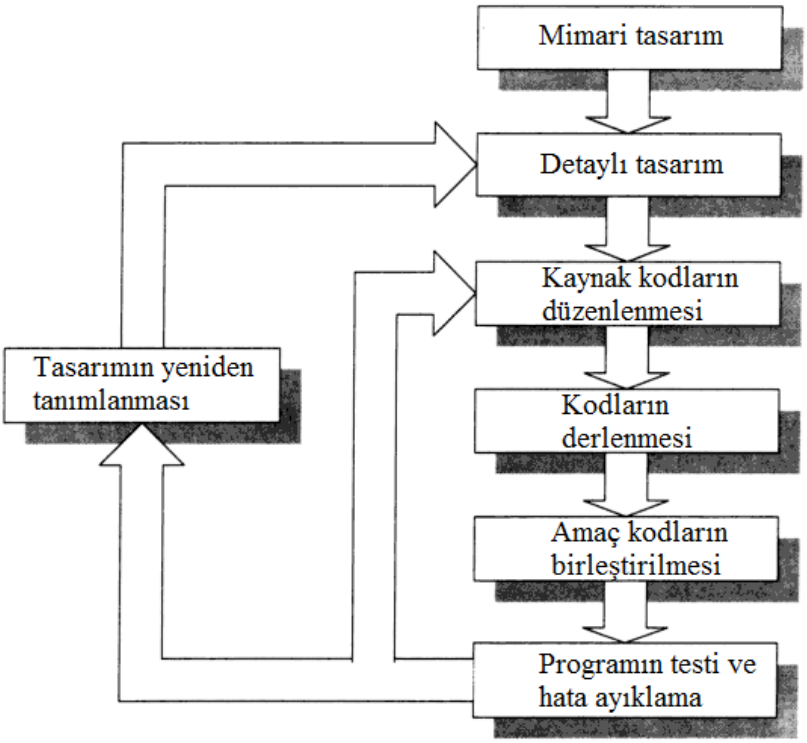
Yazılımın kaynak koduna sahip olmanın bir diğer avantajı kaynak kodu başka platformlara uyarlayarak ve başka programlama dillerine çevirerek, yazılımın o platformlarda ve dillerde de kullanımını sağlamaktır (Wilson, 2011). Linux ve Unix işletim sistemlerindeki kütüphane dosyaları (library) buna örnek olarak verilebilir. Örneğin MySql veritabanı üzerinde işlem yapmak için yazılmış olan açık kaynak libmysql kütüphanesi farklı platform ve farklı işlemciye sahip sistemlerde de kullanılabilmektedir.

Kaynak kodu edinmenin diğer avantajları ise yazılımların tekrar kullanılabilirliğini (*software reusability*) sağlaması ve yazılımın kullanımına ilişkin dokümantasyon hazırlanmasında yararlı olmasıdır (Goldman ve Gabriel, 2005; Fearnley, 2010). Kaynak kodun tekrar kullanılması ile yazılımlarda aynı işi yapacak olan kütüphane ve sınıflar tekrar yazılmayarak yazılımı kodlama zamanı kısalır.

Programın kaynak kodunda bir değişiklik yapıldığında, değişikliklerin programda etkili olması için derlenen programlama dillerinde programın tekrar derlenmesi, yorumlanan programlama dillerinde ise kaynak kodun tekrar yürütülmesi gerekmektedir (Tech Terms Dictionary, 2007).

2.1.4. Kaynak Kodun Bilgisayar Programına Dönüşmesi

Bir bilgisayar programının üretilmesi için "yazılım yaşam döngüsü" de denilen belirli aşamaların yerine getirilmesi gerekmektedir (Bronson ve Rosenthal, 2005). Önce çözülmesi istenen problemin veya yapılması istenen işin yapısal analizi yapılır. Ardından ayrıntılı analizi ve tasarımı gerçekleştirilir. Bu tasarım ve analizlere göre bilgisayar programının hangi dilde yazılması isteniyorsa o dilde kodlama yapılır. Kodlama aşamasının ardından program derlenir ve çalıştırılmaya hazır hale gelen program test aşamasına alınır. Bu aşamadan sonra yazılımın bakım aşaması başlar. Bu aşamada ortaya çıkan hatalar, düzeltilmesi, eklenmesi gereken kodlar Şekil 2.3'teki döngüde olduğu gibi yazılım yaşam döngüsü adındaki yapı içinde gerçekleşir.



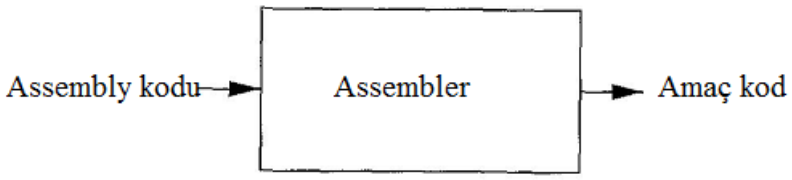
Şekil 2.3: Bilgisayar programının oluşum evreleri (Bronson ve Rosenthal, 2005)

Kaynak kod bilgisayar tarafından doğrudan çalıştırılamaz (Bloss, 2003). Kaynak kod yazıldıktan sonra bir isim verilerek kaydedilir ve kaynak kodun bilgisayar tarafından çalıştırılması için makine diline çevrilmesi gerekmektedir (Bloss, 2003; Davidson, 2003; Yadav, 2008; Freedman, 2001). Bu işi yapan programlar derleyici (*compiler*), assembler ve yorumlayıcı (*interpreter*) olarak üçe ayrılır (Freedman, 2001).

Bilgisayar programlarının bazısı çalışmak için kaynak koda ihtiyaç duymazken kimi yazılımlar kaynak kod olmadan çalışmazlar. Bu ayrım programın yazıldığı dilin özelliğine göre değişir. Yani kaynak kodun bilgisayar programı olarak çalıştırılması, kaynak kodun yazıldığı programlama dilinin derlenebilir (compiled), yorumlanabilir (interpreted) veya her ikisini de barındıran hibrit yani hem yorumlanabilir hem derlenebilir bir yapıda olmasıyla ilgilidir. Bunlara ilaveten programcılar için pek sık tercih edilmese de Assembly diliyle yazılan komutların amaç koda dönüşümü de gerekmektedir.

2.1.4.1. Assembling

Assembly kodlarının Assembler adı verilen araçlarla makinenin anlayacağı dile çevrilmesi gerekir ve bu işlem *assembling* olarak bilinir (Arora ve Bansal, 2005; Henderson, 2009). Çevrim işleminin sonucunda makine dilinde çalıştırılmaya hazır amaç kod üretilir (Şekil 2.4).



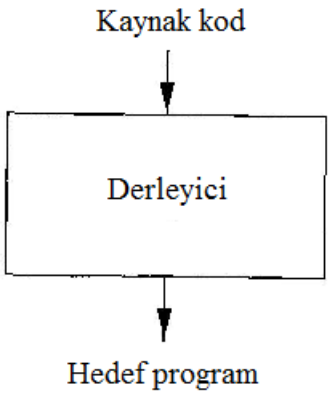
Şekil 2.4: Assembler çalışma ilkesi (Godse ve Godse, 2008)'den uyarlanmıştır.

Popüler Assembler çeviricilerine örnek olarak Microsoft Assembler Program (MASM) ve Borland Turbo Assembler Program (TASM) verilebilir (Arora ve Bansal, 2005).

2.1.4.2. Derleme

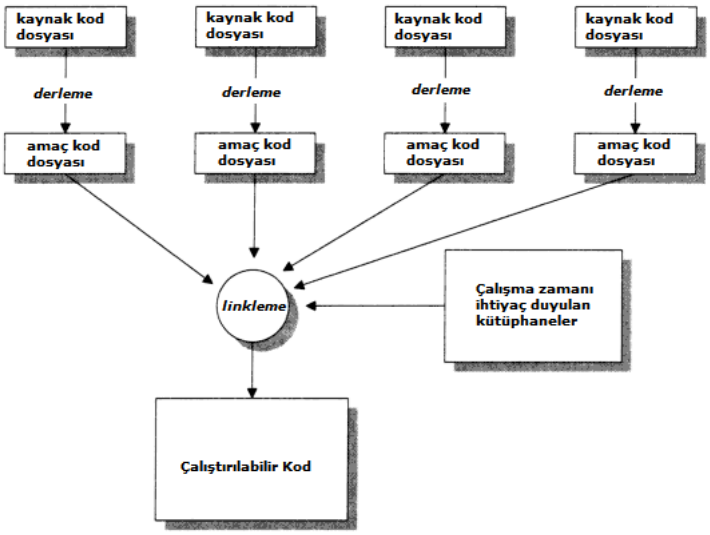
Derleme, derleyici (compiler) adı verilen programların giriş olarak bir programlama dilinde yazılmış bir kaynak kodu alıp, çıkış olarak da eşdeğer bir hedef veya başka bir dilde amaç kod üretmesi işidir (Darnell ve Margolis, 1996; Freedman, 2001). Derleyiciler programın bir formunu başka bir forma çeviren çeviricilerdir (Aho ve diğ., 2006). Derleyiciye verilen kaynak kodun programlama dili genellikle yüksek seviye bir dildir ve hedef programın dili de genellikle makine dilidir. Bununla birlikte kaynak kodu makine diline değil de başka bir programlama diline çeviren preprocessor adında çeviriciler de bulunmaktadır. Örneğin C++ diliyle yazılmış kaynak kodları C diline

çeviren derleyiciler bulunmaktadır (Freedman, 2001). Şekil 2.5'te derleyicinin çalışma prensibi görülmektedir.



Şekil 2.5: Derleyicinin çalışma prensibi (Aho ve diğ., 2006)

Kaynak kod yazıldıktan sonra derleyici her bir kaynak kodu önce amaç koda çevirir. Her bir kaynak kod dosyası için bir amaç kod oluşturulur. Bu aşamada ortada henüz çalıştırılabilir bir program yoktur. Daha sonra linker adı verilen programlar, amaç kodları birleştirir. Linker, programın çalışması için amaç kodların ihtiyaç duyduğu kütüphaneleri ve programda yer alan diğer amaç kod dosyalarını adresler. Bu adresleme işleminin ardından ortaya çalıştırılabilir kod yani bilgisayar programı çıkmış olur (Darnell ve Margolis, 1996). Kaynak kodun program haline gelmesi sırasında geçirdiği bu evreler Şekil 2.6'da gösterilmiştir.



Şekil 2.6: Kaynak kodun derlenmesi ve çalıştırılabilir program haline gelme aşamaları (Darnell ve Margolis, 1996)'dan uyarlanmıştır.

Görüldüğü gibi amaç kod ile çalıştırılabilir kod aslında teknik olarak birbirinden farklıdır. Fakat literatürde genellikle çalıştırılabilir kod yerine amaç kod terimi kullanılmaktadır. Çalışmanın geri kalanında kavram kargaşası yaratmamak adına literatüre uyularak çalıştırılabilir bilgisayar programını ifade etmek için amaç kod terimi kullanılacaktır.

Derleme işlemi bittikten sonra program kullanıcı tarafından çalıştırıldığı taktirde yürütülür ve kullanıcının programa girdiği veriler/girdiler ile sonuç alınır. Şekil 2.7'de derlenmiş programa veri girişi ve programın sonuç üretmesi gösterilmiştir.

Şekil 2.7: Derlenen programın çalıştırılması (Aho ve diğ., 2006)

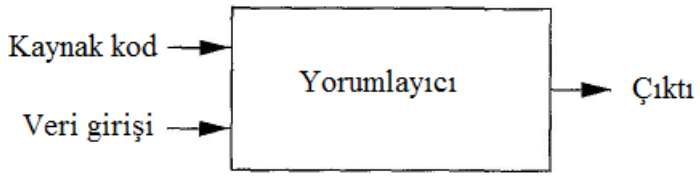
Derlenen programlama dillerinde kaynak kod derlendikten sonra çalıştırılabilir bilgisayar programı üretildiği için kaynak koda artık ihtiyaç kalmaz (Kudyba ve Diwan, 2002). Derlenen programlama dillerine örnek olarak C, C++ ve Fortran verilebilir.

2.1.4.3. Yorumlama

Kaynak koda ihtiyaç duyan yazılımlar yorumlanabilir programlama dilleriyle yazılan programlardır. Bu yazılımlar için çalıştırma değil daha çok bilgisayar tarafından "yürütülme" ifadesi kullanılmaktadır. Bilgisayarda yürütülmesi için kaynak kod gerektiren bu programlara betik (*script*) de denmektedir ve yorumlanabilir dillere "Betik diller" de denir (Tech Terms Dictionary, 2007).

Yorumlanan programlama dilleri ile yazılan programların çalışması için yorumlayıcının kaynak kodu her seferinde yorumlaması gerekmektedir (Aho ve diğ., 2006). Yorumlanan kaynak kod ile hedef program değil çıktı üretilir.

Yorumlanan dillerde, programın yürütülmesi için o dilin yorumlayıcısının bilgisayarda kurulu olması gerekmektedir. Kaynak kod yorumlayıcı tarafından satır satır yorumlanır. Bu nedenle yorumlanan diller derlenen dillere göre daha yavaş çalışmaktadır (Aho ve diğ., 2006). Şekil 2.8'de bir yorumlayıcının çalışma prensibi gösterilmiştir.



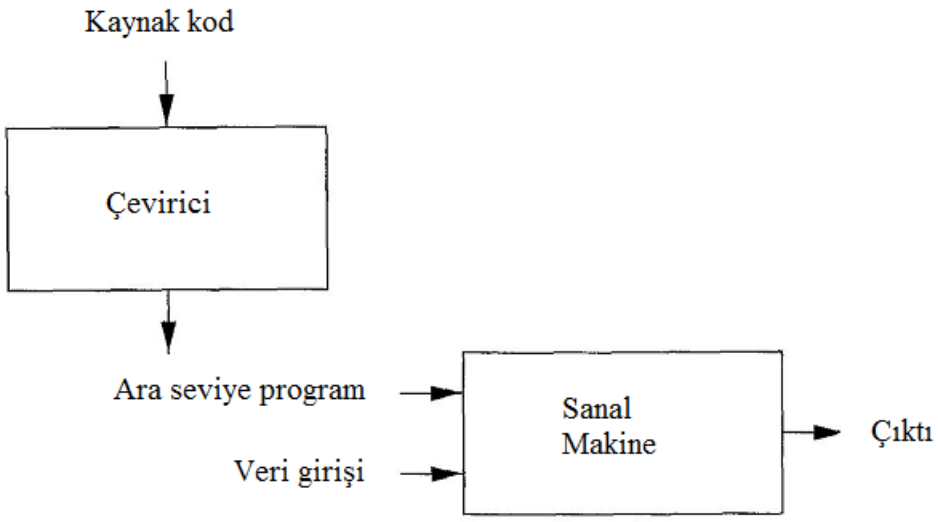
Şekil 2.8: Yorumlayıcının çalışma prensibi (Aho ve diğ., 2006)

Günümüzde yorumlanan dillere örnek olarak betik diller Javascript, PHP, ASP ve Perl, Python ve Ruby verilebilir (Lecky-Thompson, 2005; Galbraith, 2009).

2.1.4.4. Hibrit Sistem

Hibrit sistem yapı olarak hem derleyici hem de yorumlayıcı barındırır. Kaynak kod önce ara bir dile veya programa çevrilir, ardından dilin yorumlayıcısı tarafından yorumlanır (Bloss, 2003). Hibrit sistemle yazılan yazılımlar taşınabilirlik sağlarlar (Lee, 2008). Bu

sayede yazılım ilgili yorumlayıcının olduğu her platformda çalıştırılabilir. Böylece kaynak kodu her platform için tekrar yazmaya veya revize etmeye gerek kalmaz. Şekil 2.9'da hibrit sistemin çalışma esası gösterilmektedir.



Şekil 2.9: Hibrit sistemde kaynak kodun program haline gelme aşamaları (Aho ve diğ., 2006)

Hibrit dillere örnek olarak Java, Python, Perl, .NET Framework dilleri C# ve Visual Basic.NET verilebilir (Parr, 2009; Raymond, 2010).

2.2. FİKRÎ MÜLKİYET AÇISINDAN BİLGİSAYAR PROGRAMI VE KAYNAK KOD

Mülkiyet kelimesi sözlükte "sahiplik, kendisinin olan bir şeyi yasa çerçevesinde istediği gibi kullanabilme hakkı" anlamlarına gelmektedir (TDK, 2011). Hukuk dilinde ise mülkiyet kelimesi bir mala veya eşyaya ait kullanma yetkisini ifade eder (EĞİTEK, 2011). Bu hak ile mülkiyet sahibi, maliki olduğu bir mal üzerinde kanunların çizdiği sınırlar içinde kalmak koşuluyla sürekli ve inhisarî (tekelci) bir tasarruf ve egemenlik hakkına sahiptir (EĞİTEK, 2011). Mülkiyet hakkının konusu mal ve eşyalardır. Yani gözle görülür elle tutulur somut nesneler mülkiyet hukuku kapsamındadır. İnsan düşüncesi ve zihninin mamulü olan ürünler ise bu hukukun koruma kapsamında değildir. Oysa gayri maddî (maddeden oluşmayan) ve soyut düzeyde kalan fikirler ve düşünceler mülkiyet hukukuyla korunamaz (Tekinalp, 2005).

İnsan, doğası gereği maddî yada gayri maddî sahibi olduğu her şeyin korunmasını ister. Nitekim 1948 tarihli İnsan Hakları Evrensel Beyannamesi m. 27/II'de "*herkes, vücuda getirdiği her türlü bilim, edebiyat ve sanat eserlerinden doğan manevi ve parayla ölçülebilir menfaatlerinin korunmasını isteme hakkına sahiptir*" denilerek, fikrî ürün üzerindeki manevî ve malî hakların kişinin en temel insan hakları arasında yer aldığı vurgulanmıştır (Birleşmiş Milletler, 2011).

Türkiye'nin de taraf olduğu dünyada fikrî mülkiyet haklarının korunmasını amaçlayan bir kuruluş olan Dünya Fikrî Mülkiyet Teşkilatı'nın (WIPO: World Intellectual Property Organisation) Cenevre'deki binasının kitabesinde Latince olarak yer alan ifadenin Türkçe karşılığı "*İnsanın yaratıcı gücü, bütün buluş ve sanat ürünlerinin kaynağıdır. Bu eserler, insan hayatını yaşamaya değer kılan bir güvencedir. Buluşlar ve sanat eserlerinin korunmasını sağlamak, dikkatle izlenmesi gereken bir devlet görevidir."* şeklindedir ve görüldüğü gibi fikir ve düşünce ürünü olan eserlerin korunmasının gerekli olduğu ve bu korumanın devletler tarafından sağlanması gerektiği ifade edilmektedir (Beşiroğlu, 2004).

Adalet Bakanlığı'nın yayımlamış olduğu "Fikrî Ve Sınaî Hak Korsanlığı Ve Taklitçiliğinde Devletin Vergi Kaybı Hakkında Genelge"de "*Fikrî ve sınaî haklar, insanın fikrî çalışmalarıyla ortaya koyduğu ürünler üzerindeki çeşitli hak ve menfaatlerin korunması suretiyle fikrî emek ve çalışmaların teşvik edilmesi felsefesine dayanmakta olup, bu teşvik sayesinde kültür, sanat ve teknoloji alanında gelişme sağlanacağı gibi, üretimin ve piyasadaki rekabetin arttırılmasında, kültürel yaşamın geliştirilmesinde ve teknolojinin yaygınlaştırılmasında da sözü edilen hakların önemli bir rol oynayacağı şüphesizdir.*
Fikrî ürün üzerindeki hakların korunması temeline dayanan fikrî ve sınaî haklar hukuku ise, fikrî ve sınaî hak sahibinin belirlenmesini, yetkilerini, bu hakların korunmasını ve bunların çeşitli hukukî işlemlere konu olmasını düzenlemektedir." denilerek hem fikrî hakkın önemine değinmiş hem de fikrî mülkiyet hukukunun amacını belirtmiştir (Adalet Bakanlığı, 2005).

İngilizcede "*Intellectual Property*", Almancada "*Geistiges Eigentum*", Fransızcada "*Propriété Intellectuelle*" olarak geçen fikrî mülkiyet kavramı, insan zekâsı ve

düşüncesi ile yaratılmış olan her nevi eserin hukuken korunmasını amaçlamaktadır (Tekinalp, 2005; Karahan ve diğ., 2009). Literatürde fikrî mülkiyet terimi yerine fikrî ve sınaî mülkiyet bazen de fikrî, sınaî ve ticarî mülkiyet terimleri kullanılmaktadır (Tekinalp, 2005). Bu kavramların hepsinde fikrî emek söz konusu olmakla birlikte tezin geri kalanında *fikrî mülkiyet* terimi kullanılacaktır.

Dünya Ticaret Örgütü (DTÖ)'nün Ticaretle Bağlantılı Fikrî Mülkiyet Hakları Anlaşması'na (TRIPS: Trade-Related Aspects Of Intellectual Property Rights) göre fikrî mülkiyet hakları "*insanın zekâsı, entellektüel birikimi ve zihinsel yaratıcılığı ile ortaya çıkardıkları ürünler nedeniyle sahip oldukları haklardır*" (WTO, 2011b). Bu haklar, fikrî eseri meydana getiren kişilere, bu eser üzerinde belirli bir süre için maddî ve manevî açıdan ayrıcalıklı bir hak kazandırmaktadır.

Fikrî mülkiyet, fikir unsurunun ön planda olmasına ya da fikrî ürünün sanayiye uygulanabilirliği açısından fikrî mülkiyet (telif hakları) ve sınaî mülkiyet olmak üzere iki alt gruba ayrılmaktadır (Karahan ve diğ., 2009; WTO, 2011b). Buna göre, fikrî mülkiyet terimi fikir ve sanat eserlerini (bilgisayar programları ve yazılımları da içeren ilim ve edebiyat eserleri, musikî eserleri, güzel sanat eserleri ve sinema eserleri) kapsamakta; teknolojik buluşlar, mal ve hizmetlerin ticarî markaları, endüstriyel tasarımlar, patentler ve faydalı modeller, coğrafî ad ve işaretler ise sınaî mülkiyet kapsamında sayılmaktadır (Tekinalp, 2005; Karahan ve diğ., 2009).

Fikrî mülkiyet, WIPO'nun 14 Haziran 1967'de Stockholm'de imzalanan kuruluş sözleşmesi m. 2/VIII. bendinde aşağıdaki haklarla ilişkili olarak tanımlamıştır (WIPO, 2011):

- Edebi, sanatsal ve bilimsel çalışmalar;
- İcracı sanatçıların eserleri, fonogram ve radyo yayınları;
- İnsan emeğinin tüm alanlarındaki buluşları;
- Bilimsel keşifler;
- Endüstriyel tasarımlar;
- Ticarî markalar, hizmet markaları, ticarî unvan ve isimler;
- Haksız rekabete karşı koruma ve

- Endüstriyel, bilimsel, edebi ya da sanatsal alanlarda fikrî mülkiyet faaliyetlerinden kaynaklanan diğer tüm haklar.

Görüleceği üzere WIPO, fikrî mülkiyet haklarını tanımlarken fikrî mülkiyet-sınaî mülkiyet ayrımına gitmemiş, insan zihni ile üretilmiş tüm eserleri fikrî mülkiyet kapsamına almıştır.

Fikrî mülkiyet kapsamında yer alan bilgisayar programları sadece fikir ürünü oldukları için korunmaya değer oldukları gibi bunun yanında endüstriye, istihdama, devletlere ve şirketlere bakan yönleriyle de korunmalıdır.

2.2.1. Bilgisayar Programının Korunmasının Zorunluluğu

Bilgisayar teknolojisinin gelişmesiyle, bilgisayarın günlük yaşamdaki kullanım alanları ve kullanım sıklığı artmıştır. Bilgisayar sektöründeki rekabetin de etkisiyle bilgisayar fiyatları hayli ucuzlamış, bilgisayarlar artık lüks bir cihaz olmaktan çıkmıştır. Eskiden bilgisayarların pahalı bir teknoloji ürünü olması nedeniyle bilgisayarlar muhasebe uygulamaları, bilimsel hesaplamalar, akademik ve askeri araştırmalar gibi belli başlı alanlarda kullanılıyor iken artık günlük yaşamın her alanında kullanılır hale gelmiştir. Bunda bilgisayar fiyatlarının ucuzlamasının yanı sıra artık hemen hemen her alanda insanların kullanımına sunulan bilgisayar programları ve yazılımların artması da etkili olmuştur. Bilgisayarın değerli ve kullanılır bir cihaz olmasını sağlayan donanım olarak tanımlanan bilgisayarın tüm fiziksel bileşenleri değil; donanıma neyi, nasıl, ne zaman ve ne sıklıkla yapması gerektiğini "söyleyen" yazılımlardır.

Teknolojinin hızlı gelişimi bilgisayarların kullanımını artırdığı gibi, bilgisayar gibi "akıllı" diye nitelendirilebilecek cihazların çeşitliliğini de artırmıştır. Artık hemen hemen tüm sektörlerde hesaplama yeteneğine sahip ve programlanabilir bilgisayar türevi cihazlar kullanılır olmuştur. Örneğin iletişim sektöründe neredeyse tüm cep telefonları ve el bilgisayarları, ses ve görüntü oynatıcı cihazlar, kimlik algılama cihazları hatta evlerde kullanılan beyaz eşyalar dahi içinde programlanabilir yazılımlar içermektedir. Kısaca, içinde "mikroişlemci" barındıran tüm entegre devreler bir yazılımla veya bilgisayar programları ile işlerlik kazanmaktadır. Böylece yazılım,

sadece bilgisayar üzerinde çalışmaya mahkûm bir kavram olmaktan çıkmış, hemen hemen her alanda kullanılan elektronik cihazlar için üretilir olmuştur.

Yazılımların kullanım alanının yaygınlaşması dünyada ve Türkiye'de yazılım sektörünün büyümesini sağlamıştır. 2010 yılında Türkiye'de yazılım sektörü bir önceki yıla göre % 4.1 büyümüş ve ticaret hacmi 753 milyon dolara ulaşmıştır (BT Dünyası, 2011). Dünyada ise 2011 yılında BT (Information Technology) pazarının büyüklüğü Gartner Teknoloji araştırma şirketinin tahminlerine göre 3.6 trilyon doları bulacak ve bunun yaklaşık 270 milyar dolarlık kısmını yazılım harcamaları oluşturacaktır (Tablo 2.1). Aynı tahminlere göre yazılım harcamaları 2015 yılında 350 milyar doları bulacaktır (Gartner, 2011). International Data Corporation (IDC) 2011 araştırma tahminleri ise BT pazarının büyüklüğünün 1.6 trilyon dolar olacağını ve bunun 310 milyar dolarlık kısmının yazılım harcamaları olacağını öngörmüştür (CBR Staff Writer, 2010).

Tablo 2.1: Dünya BT harcamaları tahmini (Milyar ABD $) (Gartner, 2011)

Harcama Türü	**2008 Harcama**	**2009 Harcama**	**2010 Harcama**	**2011 Harcama**	**2012 Harcama**	**2013 Harcama**	**2014 Harcama**	**2015 Harcama**
Bilgisayar Donanımı	381.6	334.6	375.1	418.9	454.8	496.6	540.5	603.5
Yazılım	**231.2**	**225.5**	**244.4**	**267.6**	**288.1**	**308.4**	**329.4**	**350.9**
BT Servisleri	810.9	769.2	793	845.7	882.5	919.9	960.8	1,004.0
Telekom Ekipmanları	398.8	374.2	412.5	446.1	480.2	514.2	543.4	571.7
Telekom Servisleri	1,578.4	1,533.3	1,602.3	1,693.7	1,748.6	1,803.7	1,860.5	1,913.9
Tüm BT	**3,401.0**	**3,236.7**	**3,427.3**	**3,672.0**	**3,854.2**	**4,042.8**	**4,234.6**	**4,444.0**

Dünya çapında yazılım ve yazılımla alakalı sektörlerde yazılım geliştirici, programcı, yazılım mühendisi, analist, veritabanı programcısı vb. pozisyonlarda 1983 yılında 3 milyon 250 bin, 90'lı yıllarda 7.5 milyonun üzerinde kişi çalışmaktaydı (Jones, 1984).

2009 yılında bu sayının dünya çapında 10 ile 15 milyon kişi arasında olduğu tahmin edilmektedir (Stack Overflow, 2009).

Bilgisayar teknolojisinin gelişmesi ve internetin yaygınlaşması bilgisayar programlarının kopyalama, çoğaltma, depolama ve dağıtma imkânlarını da artırmıştır. Sıradan sayılamayacak oranda düşünce, fikir ve emek ile oluşan ve katma değeri yüksek ürünler olan bilgisayar programlarının çok kolay bir şekilde, sahibinden izin alınmaksızın çoğaltılması ve kullanımı yazılım firmalarında çok büyük ekonomik kayıplara sebep olmaktadır. Business Software Alliance (BSA) raporuna göre 2010 yılında dünya çapında korsan yazılım kullanımından kaynaklı 59 milyar dolar zarar olmuştur. Aynı raporda Türkiye'de korsan yazılım kullanımından dolayı 519 milyon dolarlık bir zararın söz konusu olduğu bildirilmektedir (BSA, 2011).

Bilgisayar programlarının ve yazılımların yasadışı yolla kullanımı yazılım sektöründe faaliyet gösteren firmaları çok büyük bir emek, zaman ve maddî kaynak kaybına uğrattığı gibi devletleri de vergi kaybına uğratmaktadır. Ayrıca korsan yazılım kullanımının yüksek olduğu ülkelere firmalar yazılım yatırımı yapmak istememektedir (Taşçı ve diğ., 2007).

Bu gibi nedenlerden dolayı bilgisayar programı ve yazılımların bir hukukî korumaya muhtaç oldukları açıktır (Eroğlu, 2000). Bu yüzden ülkeler kendi hukuklarında bilgisayar programlarını koruma altına alma yollarına gitmişlerdir. Ayrıca uluslararası anlaşmalarla da bilgisayar programları anlaşmaya taraf ülkeler nezdinde koruma altına alınmaya çalışılmıştır.

Bilgisayar programlarının korunma biçimi Almanya gibi Avrupa kıtası ülkelerinde yazılarak hazırlandıkları (yazılardan oluştuğu) gerçeği temel alınarak "edebi eser" statüsünde ve fikrî hukuk kanunları çerçevesinde, kimi ülkelerde patent hukuku çerçevesinde koruma altına alınmıştır (Eroğlu, 2000).

Türkiye'de bilgisayar programları 5846 sayılı FSEK ile korunma altına alınmıştır. "*Bilgisayar programı*" kelimesinin FSEK'e girmesi ve bilgisayar programlarının birer *eser* olduklarının kabulü ülkemizde 07.06.1995 tarihli 4110 sayılı kanun değişikliği ile

gerçekleşmiştir (Tekinalp, 2005; Aksu, 2006; Dalyan, 2008). Kanunda yazılım kelimesi geçmese de bilgisayar programı ile ilgili hükümlerin yazılımı da içerdiği düşünülmelidir. Türk hukuk sisteminde bilgisayar yazılımları FSEK'e ilaveten Rekabetin Korunması Hakkında Kanun, Türk Ceza Kanunu, Türk Ticaret Kanunu, Borçlar Kanunu ve Medeni Kanun'un ilgili maddelerince de korunmaktadır (Cimilli ve diğ., 2004).

Ülkemizde bilgisayar programları bir buluş olmadığı gerekçesiyle Patent hukuku kanunlarıyla korunmaz (Tekinalp, 2005). 551 sayılı Patent Haklarının Korunması Hakkında Kanun Hükmünde Kararname (KHK) m. 6/3 bilgisayar programlarını patent verilemeyecek konular arasında saymaktadır. İlgili KHK "münhasıran koruma talep edilmesi halinde" bilgisayar programlarına patent verilemeyeceğini söyler.

"Madde 6 - Aşağıda sayılanlar buluş niteliğinde olmadıkları için bu Kanun Hükmünde Kararname kapsamı dışında kalır:

...

c - Edebiyat ve sanat eserleri, bilim eserleri, estetik niteliği olan yaratmalar, bilgisayar yazılımları;

...

Bu maddenin birinci fıkrasında sayılanlar için münhasıran koruma talep edilmesi halinde patent verilmez."

Bu hüküm bilgisayar programlarına başlı başına patent verilemeyeceğini, ancak teknik bir sorunun çözümünde kullanılan bir elektronik aygıtta yer alıyorlarsa patent verilebileceğini ifade eder.

ABD, 1980 yılında Bilgisayar Yazılım Hakları Yasası (Computer Software Copyright Act 17 USC, 101-117), İngiltere de 1985 yılında Bilgisayar Yazılım Yasası (Copyright "Computer Software" Amendment Act) ve Tasarımlar ve Patentler yasasının (Copyright, Design and Patents Act) 3 (1-b) maddesi ile bilgisayar programları üzerindeki hakları koruma altına almıştır (Beşiroğlu, 2004). Aynı şekilde 09.06.1993 tarihinde Alman Fikir Hakları Yasası'na "Bilgisayar Programları için Özel hükümler" başlığı ile bir bölüm eklenmiş, Fransa ise 10.05.1994 tarihinde Fikir Hakları Yasasına

"Bilgisayar Programlarının Hukukî Korunması" ile bir bölüm eklemiştir (Beşiroğlu, 2004).

Gelişmiş ülkeler bilgisayar programlarının kendine has yapısı nedeniyle yasalarında bilgisayar programları için ayrı bir düzenleme yapma yoluna giderek bilgisayar programları üzerindeki hakları ayrıca tarif, tasnif ve tanzim etmişlerdir. Ülkemizde ise bilgisayar programları diğer fikir ürünleri ile aynı kanunda ve aynı haklarla birlikte ele alınmakta, kimi maddelerde bilgisayar programları için ek hükümler bulunmaktadır. Kanaatimizce bilgisayar programları ile ilgili hakların diğer eser türlerine verilen genel haklar çerçevesinde değerlendirilmesi yerine diğer ülkeler örneğinde olduğu gibi ayrı bir kanunla koruma altına alınması daha doğru bir uygulama olacaktır.

2.2.2. Fikir ve Sanat Eserleri Kanununda Eser Kavramı

Eser kelimesinin sözlük anlamı "emek sonucu ortaya konan ürün, yapıt", "iz, işaret, im" olarak tarif edilmektedir (TDK, 2011). Yani eser, gözle görülebilir, elle tutulabilir somut nesneleri isimlendirir. Ateş (2003) hukukta *eser* ve "*fikrî ürün*" kavramlarının birbirinin yerine kullanılan fakat farklı kavramlar olduğunu belirtmektedir. Yazara göre eser, bireyin fikrî mesaisinin ürünüdür. Bu anlamda fikrî hukuk açısından eseri "*kişinin sadece fikrî faaliyet sonucu ortaya koyduğu ve hukukî bakımdan değer ifade eden neticelerdir*" şeklinde tanımlamıştır. Teknik anlamda ise eserin, fikrî üründen farklı olduğunu belirtmiştir. Yazara göre her fikrî ürün eser değildir fakat her fikir ve sanat eseri birer fikrî üründür.

FSEK'de bir haktan söz edilebilmesi için korunacak şeyin bir *eser* olması gerekmektedir. FSEK, eser statüsüne sahip olan fikir ve sanat ürünlerini korumaktadır (Aksu, 2006). Bu nedenle eser kavramını iyi analiz etmek gerekmektedir. FSEK'in m. 1-B/ a bendinde eserin tanımı şöyledir:

"*Sahibinin hususiyetini taşıyan ve ilim ve edebiyat, musikî, güzel sanatlar veya sinema eserleri olarak sayılan her nevi fikir ve sanat mahsullerini ifade eder*"

Bu kategorilere ek olarak FSEK m. 8'de

"*Bir işlenmenin ve derlemenin sahibi, asıl eser sahibinin hakları mahfuz kalmak şartıyla onu işleyendir.*"

ve m. 6'da

"*Diğer bir eserden istifade suretiyle vücuda getirilip de bu eserlere nispetle müstakil olmayan*"

ifadeleri ile işlenme ve derlemeleri de eser olarak kabul etmektedir.

2.2.3. Eser Sayılmanın Şartları

Kanunda nelerin eser olabileceği ve fikir ve sanat ürünlerinin eser sayılabilmesi için aranan şartların ne olduğu belirtilmiştir. Literatürde bu konu, eserde sahibinin hususiyetini taşıması şartı (*Subjektif Unsur-Subjektif Şart-Esasa İlişkin Şart*) ve eserin kanunda belirtilen eser türlerinden birine dâhil olması (*Objektif Unsur-Objektif Şart-Şeklî Şart*) olarak iki kısımda incelenmektedir (Ateş, 2003; Kulaklı, 2008).

Fikrî ve sanat ürünlerin eser olarak korunması için, soyut bir kavram olan fikrin somut bir halde bulunması ve algılanabilir durumda olması gerekir. Tekinalp (2005), eserin bir fikrî çaba mahsulü olması gerektiğini ve eserin sahibinin hususiyetini yansıtacak düzeyde şekillenmiş olmasını yani fikrin bir eşya, bir cisim üzerinde şekil bulmuş olması gerektiğini ifade eder. Öte yandan Karahan ve diğ. (2009) eserin üçüncü kişilerce algılanabilir nitelikte olmasını yeterli görmüştür. Bunu örneklerken irticalen (sözlü) yapılmış bir söylev, spontane bir müzik veya bir tiyatro gösterisinin de telif hukuku ile korunabileceğini, eserin üçüncü kişilerce algılanabilir seviyede olmasının yeterli olduğunu ifade eder.

FSEK, eser tanımı veya eserde bulunması gereken özellikler içinde sanatsal ya da estetik güzellik bulunmasını şart koşmamıştır. Şüphesiz ki, kimi eserlerde sanatsal ya da estetik kaygı ön plandadır. Bununla birlikte kimi eserler, sanatsal ya da estetik kaygı aranmaksızın ticarî amaç için üretilir. Bilgisayar programlarında da sanatsal yaratım değil ticarî kaygılar ön plandadır (Cimilli ve diğ., 2004). Bu nedenle bilgisayar programlarının estetik açıdan göze hoş gelmesi, eser olması için aranan şartlardan biri değildir. Buna paralel olarak Avrupa Konseyi Direktifinin giriş kısmında bilgisayar programlarının eser olarak tespiti için estetik açıdan değerlendirilmemesi gerektiği yazılıdır:

"*Bir bilgisayar programının özgün bir eser olup olmadığının kararlaştırılması için uygulanacak kriterler ile ilgili olarak, programın nitelik ve estetik değerinin tespit edilmesi için her hangi bir test uygulanmamasını…*"

Kanaatimizce de kişiye göre değişen (subjektif) bir unsur olan güzellik kavramını eser olmanın şartları arasında saymak şüphesiz ki akla ve mantığa ters bir durumdur. Bir fikrî veya sanatsal çalışmanın sanatsal ve estetik niteliğini kanunlar değil o sanatın ve sahanın uzmanları takdir eder.

Bir fikir veya sanat ürünün ekonomik açıdan değerli olması, onun eser sayılmasını gerektirmemektedir. Sahibinin hususiyetini taşıyan ve kanunda yer alan eser türlerinden birine dâhil olan her fikir veya sanat ürünü ekonomik değerine veya iktisadi elverişliliğine bakılmaksızın eser olarak korunur (Dalyan, 2008).

Bunlara ek olarak fikir ve sanat eserleri hukuku açısından eserlerin işlevselliğinin de bir önemi yoktur (Aksu, 2006). FSEK eserin bir işlev yerine getirdiğine bakmaksızın eseri koruma altına alır. Bu açıdan bilgisayar programları da işlevi bulunup bulunmadığına veya işlevini yerine getirdiğine bakılmaksızın korunurlar.

2.2.3.1. Objektif Şart

Kanun ilim ve edebiyat, musikî, güzel sanatlar ve sinema eserleri olmak üzere eser kategorilerini *numerus clausus (sınırlı sayıda olma, sayılabilme)* prensibiyle sınırlayarak dört ana başlıkta toplamıştır. Buradan hareketle bir fikir ve sanat ürününün eser olarak korunabilmesi için bu kategorilerden birine dâhil olması gerekmektedir (Aksu, 2006). Bu kategorilerin dışındaki bir ürün FSEK çerçevesinde eser olarak korunamaz.

2.2.3.2. Subjektif Şart

FSEK, eser tanımlamasında "*sahibinin hususiyetini taşıyan*" ibaresi ile bir fikir ve sanat ürününün eser olarak korunabilmesi için, eserin yaratıcısının hususiyetini taşıması gerektiğini belirtmiştir. Sahibinin hususiyetini taşıması şartından kastın ne olduğu ile ilgili literatürde farklı yorumlar yapılmıştır.

Hirsch (1943), "*herkes tarafından vücuda getirilemeyen, yani bir hususiyeti haiz bulunan mahsuller, himayeye layıktır ve ancak bunlara eser vasfı izafe edilebilir. Eğer bir mahsul herkes tarafından vücuda getirilebilecek mahiyette ise, hususiyet de (özellik) mevcut olmayacağından, bu kabil mahsulleri himaye etmekte cemiyetin hiçbir menfaati yoktur."* diyerek eserin tek veya var olandan başka bir biçimde ifade edilmesi gerektiğini söyler.

Diğer bir görüş ise eserde özgünlük kavramı ile kastedilenin o eserin kendi alanında tek ve benzersiz olması demek değildir. Ateş (2003) eserde özgünlüğü, sahibine özgü anlatım, yöntem ve biçimindeki özgünlüğü ifade etmesi olarak yorumlamıştır.

Aynı bağlamda Tekinalp (2005) hususiyetin eserin anlatımında ve üslubunda aranması gerektiğini; her eser türüne göre farklı bir üslup olacağını belirtmekle birlikte eserde orijinallik aramanın FSEK kapsamında korunacak eser kapsamını çok daraltacağını, *sahibinin hususiyetini taşıma* şartını "sıradan olmama" veya "fikrî ürünün belirli bir düzeye ulaşmış olması" şeklinde yorumlamanın kanunun mantığına daha uygun olacağını söylemektedir.

Erel (1994), bir eserin sahibinin özelliğini taşıyor sayılması için bağımsız bir fikrî çalışma ürünü olması ve böylece sahibinin yaratıcı gücünün özelliğini yansıtabilmesi gerektiğini ancak fikrî çalışmanın bağımsızlığının ve yaratıcılığının mutlak bir anlamda anlaşılamayacağını, eserin bağımsız bir fikrî çalışma ürünü olmasının kendisinden önce yaratılan diğer eserlerden istifade edilemeyeceği anlamına gelmediğini ifade eder. Eserde sahibine atfedilebilecek az çok bağımsız bir fikrî emeğin bulunmasının bu açıdan yeteceğini, eserde daha önce duyulmamış veya görülmemiş mutlak bir orijinalite aranmaması gerektiğini ve yaratıcılıktan kastın var olandan başkasını meydana getirmek şeklinde anlaşılması gerektiğini söyler. Bu konuyu bilgisayar programları çerçevesinde değerlendirirken de "*bir bilgisayar programının eser sayılıp korunabilmesi için programcının kendisinden öncekilere nazaran daha fazla bir yaratıcılık veya hüner göstermesine yahut programın basit veya karmaşık oluşuna göre yapılacak bir kalite değerlendirmesine ihtiyaç olmadığını, eserin, sahibinin kendi fikri faaliyetinin ürünü olmasının ve başka bir eserden kopyalanmamış olmasının bilgisayar programlarına fikri hukukun sağlayacağı koruma için yeterli olacağını"* söyleyerek bilgisayar

programının kopyalanmamış olmaması ve sahibi tarafından meydana getirilmiş bir fikrî ürün olmasını bilgisayar programının eser olarak korunması için yeterli görmüştür.

Aksu (2006) sahibinin hususiyetini *"eserde yaratıcısının öznel yapısının, fikrinin, bilgisinin, katkısının olması ve başkası tarafından yaratılmış bir eseri üstlenilmemesi"* olarak ele alıp, eserin mutlak anlamda yeni olması gerekmediğini ama kendini daha önceki biçim hazinesinden yeterli derecede ayırması gerektiğini söyler. Yazar ayrıca "sahibinin hususiyeti" ile "eserin belirli bir yaratım seviyesine sahip olmasının" farklı kavramlar olduğunu, belirli bir yaratım seviyesine sahip olmayan olağan nitelikteki eserlerin fikir hukuku bakımından korunamayacağını belirtir.

Bayamlıoğlu (2007) orijinalite unsurunun, eser korumasının herhangi bir sanatsal veya bilimsel yeterlilik gereğinden bağımsız oluşuyla beraber değerlendirilmesi gerektiğini belirterek, orijinaliteden kastın, eserde belirli bir kalitatif nitelik bulunması anlamına gelmediğini ifade eder. Yazar devamla belirli ölçüde yaratıcısının fikrî ürünü olarak kabul edilen her çalışmanın estetik veya bilimsel değerine bakılmaksızın korumadan faydalanacağını söyler.

Uslu (2003), yaratıcılık kavramını bilgisayar programları açısından değerlendirirken "*Bir bilgisayar programı, eser sahibinin kendi fikri yaratımı olması bakımından orijinalse korunur. Koruma kapsamına alınacak eserin tesbitinde başka hiçbir kriter uygulanmaz.*" şeklindeki Avrupa Konseyi Direktifi m. 3'e atıfla bilgisayar programlarının herhangi bir kalitatif veya estetik değerlendirmeye tabi tutulmadan, eser sahibinin kişisel yaratımı mevcutsa ve bu intihale varmayacak derecede orijinalse fikrî korumadan yararlanabileceğini ve bundan başka bir kriter aranmaması gerektiğini ifade eder.

Eserde hususiyet kavramı ile yakın fakat farklı bir konu da eserin özgülüğüdür. Özgün kelimesinin, yalnız kendine özgü bir nitelik taşıyan, orijinal, ilk meydana getirilen, kopya olmayan, (çevirme esere karşın olarak) asıl metin, başkalarını örnek tutmayıp kendisi örneklik eser veren (yazar) ve bu yolda meydana getirilen (eser) gibi lügat anlamları vardır (TDK, 2011). Bu kavram fikir hukukunda, lügat anlamında olduğu şekilde eserin kendi alanında tek ve benzersiz olması demek değildir. Böyle bir

durumda FSEK yalnızca şaheser, başyapıt gibi eserleri koruma altına alacak; bunun dışındaki diğer fikir ve sanat üretimleri koruma kapsamı dışında kalacaktır. Öte yandan şaheser nitelikte her fikrî veya sanatsal ürünün eser sayılamayacağı da ortadadır. Tekinalp (2005) özgünlüğün, eserin üslubunda ve anlatımında kendini gösterdiğini söyler. Yine aynı paralelde Yılmaz (2007), fikir hukukunun, fikirlerin bizatihi kendisinin özgün olup olmadığıyla değil, fikirlerin anlatımıyla yani dışa yansıyan biçimiyle ilgilendiğini belirtir. Yazar devamla, eserin özgünlüğü ifadesinden anlatılmak istenenin eserin başka bir eserden kopya olup olmadığı ve eser sahibinin esere kendisine has özellikleri verip vermediği olduğunu belirtir.

Bu anlatılanlar çerçevesinde kanaatimizce de bir eserin başka bir eserden birebir kopya edilmesi, çoğaltılması veya intihal edilmesi gibi bir durum söz konusu değilse; başka bir eserden istifade edilmek suretiyle meydana getirilen tüm eserlerde, sahibinin bir fikrî emeği mevcutsa eserde sahibinin hususiyetini bulundurduğu kabul edilmelidir. Eserin yeni olması veya daha önce bir benzerinin üretilmemiş olması bir şart olarak ele alınamaz. Doktrinde yer alan sahibinin hususiyetini taşıması şartından "eserin orijinal olması gerektiği" anlamı çıkarılması fikir ve sanat eserlerinde üretimi kısıtlar ve çoğu fikir ve sanat ürünü kanun korumasının dışında kalır. Nitekim FSEK m. 17/III'de "*Eserin tek ve özgün olması durumunda eser sahibi, kendisine ait tüm dönemleri kapsayan çalışma ve sergilerde kullanmak amacıyla, koruma şartlarını yerine getirerek iade edilmek üzere eseri isteyebilir."* denilmekte, kanun metninde geçen "*eserin tek ve özgün olması durumunda*" ifadeleri ile FSEK kapsamında tek ve özgün olmayan *eserlerin* de olabileceği dolaylı da olsa ifade edilmektedir. Böylelikle eserde hususiyetin anlamının, eserin orijinal olması gerektiği tezi geçersiz kalmaktadır.

Hususiyet ile ilgili bir diğer tartışma konusu ise bilgisayar programlarında kaynak kodun, amaç koda çevrildiği son aşamada insan müdahalesinin bulunmaması nedeniyle bilgisayar programlarının eser niteliğini kazanıp kazanamayacağıdır. Bilgisayar programlarının sahibi tarafından meydana getirilmediği, bir makine aracılığıyla oluşturulduğu yönünde düşünenlere göre bilgisayar programları eser olarak kabul edilmemelidir. Buna gerekçeleri de kaynak kodun bilgisayar programına dönüşme sürecinde son aşama olan derleme aşamasında insan müdahalesi bulunmaması, bu işin derleyici denilen programlar tarafından yapılmasıdır. Oysa bir eser yaratımında

kullanılan teknik araçlar, bireysel yaratımı ortadan kaldırmaz (Aksu, 2006). Bilgisayar programcıları, derleyiciyi program oluşturmada teknik bir araç olarak kullanmaktadırlar. Bu nedenle, bilgisayar programlarının eser olarak korunamayacağı yönündeki görüş geçersizdir.

2.2.4. Bilgisayar Programlarının Eser Niteliği

Bilgisayar programları, bir dizi programlama faaliyeti sonrasında oluşan son üründür. Bir bilgisayar programı bağımsız olarak geliştirilebileceği gibi, başka bir bilgisayar programından istifade edilmek suretiyle de geliştirilmiş olabilir. Bu iki durumun eser statüsü, FSEK çerçevesinde ayrı ayrı incelenecektir. Bununla birlikte bilgisayar programlarının aşamalı geliştirme yapısı göz önüne alındığında, bilgisayar programının unsurlarından hangilerinin eser niteliğinde olduğu hem yazılım endüstrisi hem de yazılımın eser sahibi olan geliştiriciler açısından önemlidir. Bunlara ilaveten tamamlanmamış bilgisayar programının FSEK kapsamında korunup korunmayacağı, yani eser olarak niteliği ayrıca incelenecektir.

Bilgisayar programlarının eser korumasından yararlanabilmesi için bir cisim üzerinde kaydedilmiş olması gerekir. Fikir aşamasında veya biçim kazanmamış hiçbir program eser olarak korunmaz. Bilgisayar programlarının bir eşya üzerinde tecessüm etmesi ise ancak bir veri depolama ünitesi üzerinde bulunmasıyla mümkün olmaktadır. Bilgisayar programları yapıları gereği cisimlendikleri somut eşyadan bağımsız olduklarından üzerine kaydedildikleri veri taşıyıcısının çeşidinin veya türünün bir önemi yoktur. Tamamlanmış halde olan; CD-HDD-manyetik teyp gibi optik-elektronik-manyetik gibi herhangi bir kayıt ortamına aktarılmış veya elektronik cihazlara, donanıma veya belleklere gömülü (entegre) haldeki programlar korumadan yararlanırlar (Karahan ve diğ., 2009).

Bilgisayar programlarının eser olarak kabulü için FSEK'te ilim ve edebiyat eserleri kapsamında değerlendirildiğinden eser kategorilerinden birine girme şartı zaten sağlanmaktadır. Bu nedenle bilgisayar programlarının eser olarak korunması için gereken tek şart sahibinin hususiyetini taşıması şartıdır (Dalyan, 2008). Sahibinin hususiyetini taşıyan işlenme veya bağımsız bilgisayar programları eser olarak korunur.

2.2.5. Bağımsız Bilgisayar Programlarının Eser Niteliği

Bilgisayar programları, FSEK m. 2/I'de "*Herhangi bir şekilde dil ve yazı ile ifade olunan eserler ve her biçim altında ifade edilen bilgisayar programları ve bir sonraki aşamada program sonucu doğurması koşuluyla bunların hazırlık tasarımları*" ifadeleriyle *ilim ve edebiyat eserleri* kategorisinde eser kapsamına dâhil edilmiştir. Yine kanunda bilgisayar programının tanımı da yapılmıştır. FSEK, m. 1/B g bendinde bilgisayar programını; "*Bilgisayar programı: Bir bilgisayar sisteminin özel bir işlem veya görev yapmasını sağlayacak bir şekilde düzene konulmuş bilgisayar emir dizgesini ve bu emir dizgesinin oluşum ve gelişimini sağlayacak hazırlık çalışmalarını… ifade eder.*" şeklinde tanımlamıştır.

ABD Telif Hukuku ise (US Copyright Act 1980) bilgisayar programlarını "Bir bilgisayar programı, bilgisayarda belirli bir sonuç elde etmek üzere doğrudan veya dolaylı olarak kullanılacak bir dizi veri veya talimatlardır." şeklinde tanımlamıştır (Erel, 1994).

Avrupa Konseyi Direktifi bilgisayar programlarının fikir ve sanat eserleri hukuku kapsamında, Bern Sözleşmesinin kastettiği anlamda edebi eser olarak korunmasını esas almıştır (Aksu, 2006; EU, 2011). Bu direktif paralelinde Kıta Avrupası ülkeleri bilgisayar programlarının fikrî hukuk çerçevesinde korunmasına ilişkin kanunlarında düzenlemeler yapmışlardır. Erel (1994), direktifte "eser" olarak bilgisayar programının bir tanımı verilmediğini ancak talimata eklenen memoranduma göre bu terimin *"bilgisayarın belirli bir fonksiyon veya görevi ifa etmesi amacıyla verilen bir dizi talimatın herhangi bir şekil, lisan, notasyon veya kodla ifade edilmiş halini belirttiğini, aynı zamanda bu terimin, akış şeması (flow chart) ve donanım içinde yerleştirilmiş program gibi asıl programın hazırlık ve tasarım malzemesini de kapsayacak şekilde geniş yorumlanması gerektiğini"* ifade eder.

TRIPS m. 10/I fıkrasında da "*Kaynak veya nesne kodundaki bilgisayar programları Bern Sözleşmesi (1971) kapsamında edebi eserler olarak korunacaktır.*" denilerek bilgisayar programlarını edebi eser kategorisinde ele almıştır (Beşiroğlu, 2004; Aksu, 2006; WTO, 2011a).

WIPO tarafından yayımlanan ve 5647 sayılı kanunla 2.5.2007 tarihinde Türkiye'nin de taraf olduğu "WIPO Telif Hakları Andlaşması" m. 4'te "*Bilgisayar programları, Bern Sözleşmesi'nin 2 nci maddesi anlamında edebiyat eseri olarak korunur. Öngörülen koruma, hangi koşul ya da biçimle ifade edilirse edilsin bilgisayar programlarına uygulanacaktır.*" denilerek bilgisayar programlarının, kaynak kodu yada nesne kodu formunda olduğuna bakılmaksızın, edebi eser olarak korunduğu belirtilmiştir (WIPO, 1996; Aksu, 2006; TBMM, 2007).

FSEK, bilgisayar programlarını, "*Herhangi bir şekilde dil ve yazı ile ifade olunan eserler*" ifadeleriyle edebî eser kategorisinde tanımlamıştır. Gerçekte de bilgisayar programları konuşma dillerine benzemeyen, kendi içinde belli bir yapısı ve kuralları olan ve kendine has belli bir yazım tarzı olan programlama dilleri ile yazılmaktadır. FSEK, dil ve yazı ile ifade olunan eserler içerisinde özel bir dil, notasyon, işaret, sinyal, harf gibi özel bir tarife ya da ayrıma gitmemiştir. Bu nedenle bilgisayar programlama dillerinin ne olması gerektiğinin veya hangi şartları taşıması gerektiğinin bir önemi yoktur. Herhangi bir programlama dili ile yazılmış bir bilgisayar programı eser olarak korunur. Avrupa Konseyi Direktifinden önce de FSEK, kanunda *bilgisayar programı* ifadesi olmamasına rağmen bilgisayar programlarını ilim ve edebiyat eseri kapsamında "*her tür dil ile ifade edilen eserler*" altında değerlendirerek bilimsel eserler olarak koruma sağlamaktaydı (Aksu, 2006). Bu konuda yazarın eserinin dipnotunda belirttiği Danıştay 10. Dairesinin 27.4.1994 tarihli kararı (E. 1992/4550 esas numarası ve 1994/1856 karar numarası) örnek olarak verilebilir. Kararda dil ile ifade olunan eserler geniş bir fikir alanını kapsadığı ve korumanın tek gereğinin fikrî içeriği herhangi bir şekildeki dil ile ifade edilen eserin, sahibinin hususiyetini taşıması gerektiği belirtilmiştir. Kararda devamla, "*bilgisayar programları; program sonucu doğuran hazırlık tasarım çalışmalarıyla birlikte ve her biçim altındaki ifadesiyle; eğer sahibinin kendi fikri yaratımı anlamında özgün ise, Fikir ve Sanat Eserleri Kanunu'nun 2/1.maddesi kapsamında ilim ve edebiyat eseri olarak korunacaktır."* denilmektedir (HukukTurk, 2004).

Hem uluslararası antlaşmalar hem de FSEK ham halde işlenmemiş fikri korumaz. Fikir hukukunda esas olan fikir değil, fikrin dışa vurumunu, fikrin ifade ediliş biçimini korumaktır (Karahan ve diğ., 2009). TRIPS Antlaşmasının m. 9'un 2. paragrafında;

"*Telif hakkının korunması fikirleri, usulleri, işletme yöntemlerini veya buna benzer matematiksel kavramları değil, ifadeleri kapsayacaktır.*" denilmektedir (Beşiroğlu, 2004; WTO, 2011a).

Aynı şekilde Avrupa Konseyi Direktifi m. 1/II'de *"... bir bilgisayar programının herhangi bir elemanına esas olan fikir ve prensipler, fikri hukuk tarafından bu talimata göre korunmaz."* ifadeleri ile fikrin korunmadığını belirtmiştir.

Bu esas, bilgisayar programları için de geçerlidir. Herhangi bir konu veya sektöre yönelik bilgisayar programı geliştirmiş olan biri, fikrin kendisine ait olduğunu iddia ederek o alanda bilgisayar programı geliştirmek isteyenleri engelleyemez. FSEK ile sağlanan eser koruması, düşüncenin ifade ediliş şeklini kapsar. Bu nedenle bilgisayar programındaki ilke veya düşünce korunmaz. Nitekim FSEK m. 2'nin sonlarında *"Ara yüzüne temel oluşturan düşünce ve ilkeleri de içine almak üzere, bir bilgisayar programının herhangi bir öğesine temel oluşturan düşünce ve ilkeler eser sayılmazlar."* denilmektedir. Bu konuda bir istisna olarak patentli bir algoritma üzerine program geliştirmek isteyenler, patent sahibinden izin almak durumundadır. Örneğin tescilli/patentli (*Proprietary Format*) sıkıştırma formatı olan .rar formatını açan, dosyaları sıkıştıran bir program yazımı için .rar patentinin sahibinden izin alınmalıdır.

2.2.6. İşlenme Bilgisayar Programlarının Eser Niteliği

Bir eserden istifade edilmek suretiyle meydana getirilen fikir ve sanat eserleri işlenmedir (Karahan ve diğ., 2009). FSEK m. 1/B bendinde işlenme eseri şöyle tanımlamıştır:

c) İşlenme eser: Diğer bir eserden istifade suretiyle vücuda getirilip de bu esere nispetle müstakil olmayan ve işleyenin hususiyetini taşıyan fikir ve sanat mahsullerini;
... ifade eder.

Yine FSEK m. 6/10. bendinde işlenme eserler arasında *"Bir bilgisayar programının uyarlanması, düzenlenmesi veya her hangi bir değişim yapılması. ... İstifade edilen eserin sahibinin haklarına zarar getirmemek şartıyla oluşturulan ve işleyenin hususiyetini taşıyan işlenmeler, bu Kanuna göre eser sayılır."* ifadeleri ile bilgisayar programlarını zikrederek işlenme bilgisayar programlarını da eser olarak kabul etmiştir.

Aynı madde işlenme eserlerin korunması için iki şart koşmuştur:

"İstifade edilen eserin sahibinin haklarına zarar getirmemek şartıyla oluşturulan ve işleyenin hususiyetini taşıyan işlenmeler, bu Kanuna göre eser sayılır."

Buna göre asıl eser sahibinin eseri üzerindeki herhangi bir hakkına zarar getirmeyen ve sahibinin hususiyetini taşıyan bilgisayar programları da bir işlenme eserdir ve kanun çerçevesinde korunurlar.

İşlenme eserlerin sahibi FSEK m. 8'de asıl eser sahibi değil, eseri işleyen olarak tanımlamıştır:

"Bir eserin sahibi onu meydana getirendir. Bir işlenmenin ve derlemenin sahibi, asıl eser sahibinin hakları mahfuz kalmak şartıyla onu işleyendir."

Bu hükme dayanarak işlenme bilgisayar programlarının sahibinin, işleyen olduğu açıktır.

İşlenme eser sahibi, eseri üzerindeki malî hakları FSEK m. 20'ye göre asıl eser sahibinin müsaade ettiği nispette kullanabilir:

"Bir işlenmenin sahibi, kendisine bu sıfatla tanınan malî hakları, işleme hususunun serbest olduğu haller dışında, asıl eser sahibinin müsaade ettiği nispette kullanabilir."

İşlenme bilgisayar programının sahibi, asıl bilgisayar sahibinin izni olmadan programı şahsî kullanımın ötesinde ekonomik menfaat elde etmek için kullanamaz (Karahan ve diğ., 2009).

Kısaltmalar, eklemeler ve teknik değişiklikler gibi teferruata ait yardımlar FSEK m. 10/2 de *"Bir eserin vücuda getirilmesinde yapılan teknik hizmetler veya teferruata ait yardımlar, iştirake esas teşkil etmez."* ifade edildiği gibi eser üzerinde esaslı olmayan değişiklikler işlenme sayılmaz (Karahan ve diğ., 2009). Bu bakımdan bir bilgisayar programına hususiyet katmayacak kadar az olan değişiklikler işlenme olarak değerlendirilmemelidir.

Kanun işlenme eser ile asıl eser arasındaki ilişkiyi m. 15/II ile belirlemiştir:

"Bir güzel sanat eserinden çoğaltma ile elde edilen kopyalarla bir işlenmenin aslı veya çoğaltılmış nüshaları üzerinde asıl eser sahibinin ad veya alametinin, kararlaştırılan veya adet olan şekilde belirtilmesi ve vücuda getirilen eserin bir kopya veya işlenme olduğunun açıkça gösterilmesi şarttır."

Buna göre işlenme eserde asıl eser sahibinin adı veya alameti yer almalıdır. Aksi takdirde işleyen başkasına ait bir eseri kendisi meydana getirmiş, intihal yapmış gibi bir izlenim verecektir (Ateş, 2003).

Bilgisayar programının başka bir dile uyarlanması işlenme eser olarak değerlendirilmelidir (Aksu, 2006). FSEK m. 6/10'da işlenmeleri sayarken kanun metninde geçen *"Bir bilgisayar programının uyarlanması, düzenlenmesi veya her hangi bir değişim yapılması."* ifadeleri kanaatimizce kaynak kodun başka bir dile uyarlanmasını kastetmektedir.

FSEK m. 38/III bir bilgisayar programını yasal yollarla elde eden kişinin program üzerinde değişiklik yapma hakkı olduğunu beyan eder:

"Bilgisayar programını yasal yollardan edinen kişinin programı yüklemesi, çalıştırması ve hataları düzeltmesi sözleşme ile önlenemez."

Burada *hataları düzeltmesi* ifadesinin üzerinde durmak gerekmektedir. Programın asıl amacına uygun düzeltme ve değişiklikler işlenme olarak ele alınamaz (Aksu, 2006).

Bilgisayar programları kaynak kodu ile birlikte alınmışsa ve programın lisansında veya satış sözleşmesinde aksini öngören bir madde yoksa amacı doğrultusunda iyileştirme yapmak üzere işlenebilir. Nitekim aynı maddenin II. bendi, *"Sözleşmede belirleyici hükümlerinin yokluğu durumunda, hata düzeltme de dahil, bilgisayar programının düşünüldüğü amaca uygun kullanımı için gerekli olduğu durumda, bilgisayar programının onu hukuki yollardan edinen kişi tarafından çoğaltılması ve işlenmesi serbesttir."* demektedir. Kanun burada programın işlenmesini serbest bırakmış fakat yapılan değişiklikler sonrası bilgisayar programına işlenme eser statüsü vermemiştir. Kanaatimizce de yasal yollarla elde edilmiş bilgisayar programının üzerinde yapılan her değişiklik, o bilgisayar programının işlenme eser sayılmasını sağlamaz. Kanun

metninde geçen "*işlenmesi"* ifadesinden dolayı, değiştirilmiş bilgisayar programını işlenme bilgisayar programı olarak ele almamak gerekmektedir (Aksu, 2006).

Bir eserin işlenmesinden amaç, bağımsız bir eser yaratmak değil, mevcut bir eseri başka bir şekle dönüştürerek ifade etmektir. Bu nedenle, değişiklik veya uygulama suretiyle geliştirilen bilgisayar programının, işleme eser olarak kabulü için asıl bilgisayar programlarının özelliklerini yansıtması gereklidir. FSEK m. 6/III hükmüne göre, program geliştiricisinin yaptığı işlenme faaliyetlerinde de kendisine ait bir özellik bulunması şarttır. Bu bakımdan, herkesin bir bilgisayar programında rutin olarak yapabileceği değişiklikler, işlenme eser sayılmaz.

2.2.7. Tamamlanmamış Bilgisayar Programlarının Eser Niteliği

Tekinalp (2005), tamamlanmamış da olsa "*belirli bir düzeye gelmiş"* çalışmaların eser olarak korunabileceğini belirtir. FSEK henüz tamamlanmamış, bir sonraki aşamada bilgisayar programı hüviyeti kazanacak, kanunun eserlerde öngördüğü şartları yerine getiren bilgisayar programlarına eser olarak koruma sağlamaktadır (Karahan ve diğ., 2009). Fakat sadece tasarım halinde kalmış bir bilgisayar programı, eser olarak korunamaz (Tekinalp, 2005).

2.2.8. Bilgisayar Programı Unsurlarının Eser Niteliği

Bilgisayar programları kendine has (*sui generis*) yapısı nedeniyle geliştirme sürecinde çeşitli aşamalardan geçerler. Bilgisayar programları, birbirinden ayrılması ve gözlemlenmesi mümkün olan ve biri diğerinin üzerine inşa edilen veya diğerinin yardımıyla ortaya konan bir geliştirme süreci içerisinde meydana gelen program akışı, algoritma, kaynak ve amaç kodu ve kullanıcı arayüzü unsurlarından oluşmaktadır (Aksu, 2006).

Eroğlu (2000), sürekli bir gelişme sürecinde ortaya çıkan eserlerde, eser kavramının sadece tamamlanmış olan eserleri değil, önceki geliştirme aşamalarında da ortaya çıkacağını belirtir. Öte yandan FSEK m. 13/II "*Eser sahibine tanınan hak ve salahiyetler eserin bütününe ve parçalarına şamildir.*" demektedir. Fakat kanun metninde geçen "*eserin parçaları*" ifadesinden neyin kastedildiği açık değildir. Bu

nedenle bilgisayar programını oluşturan unsurların eser niteliğini FSEK kapsamında tek tek ele almak gerekmektedir.

Bitirilmemiş bir bilgisayar programının unsurlarının hukukî durumu, diğer eser türlerindeki durum gibidir. Henüz bitirilmemiş bir eserin bölümleri kendi başına eser olarak nitelenebilecek konumda iseler, kanunun koyduğu şartları yerine getirmek kaydıyla eser olarak korunurlar (Aksu, 2006). Bu durum bitirilmemiş bilgisayar programının kanunun bilgisayar programında eser sayıp koruma kapsamına aldığı bölümleri için de geçerlidir.

2.2.8.1. Hazırlık Tasarımları

FSEK m. 1/B'de tanımlar yapılırken,

"*Bilgisayar programı: Bir bilgisayar sisteminin özel bir işlem veya görev yapmasını sağlayacak bir şekilde düzene konulmuş bilgisayar emir dizgesini ve* ***bu emir dizgesinin oluşum ve gelişimini sağlayacak hazırlık çalışmalarını****; ifade eder.*" şeklinde bilgisayar programını, hazırlık çalışmalarını da içine alacak şekilde tanımlamıştır. FSEK m. 2/I. bendinde "*...her biçim altında ifade edilen bilgisayar programları ve* ***bir sonraki aşamada program sonucu doğurması koşuluyla*** *bunların* ***hazırlık tasarımları***" ifadeleri ile hazırlık tasarımları koruma altına alınmıştır. Hazırlık tasarımlarının neyi ifade ettiği veya neleri kapsadığı tam olarak belirtilmemiş sadece bir sonraki aşamada bilgisayar programı olabilecek çalışmalar olduğu kanun koyucu tarafından vazedilmiştir.

Avrupa Konseyi Direktifinin giriş kısmında ve birinci maddesinde "*preparatory work", "preparatory design material", "preparatory design work"* ifadeleri ile bilgisayar programı kavramının daha sonraki bir aşamada program olarak sonuçlanabilecek bilgisayar programı geliştirmeye öncülük eden ve program geliştirmenin doğası gereği yapılan hazırlık tasarımlarını, ön çalışma-taslak mahiyetteki tasarım materyallerini de içerdiği belirtilmiştir (Aksu, 2006; EU, 2011). Aksu (2006) Direktifte "*sonraki bir aşamada program olarak sonuçlanabilecek taslak ve hazırlık tasarımları*" ifadelerini bir şart değil, açıklayıcı mahiyetteki ifadeler olduğunu söylemiştir. Fakat FSEK'te hazırlık tasarımlarının eser olarak korunmasında "*bir sonraki aşamada"* ifadesi ile program sonucunun aranması bir şart olarak belirlenmiştir. Buradan henüz program haline gelmemiş bilgisayar programı çalışmalarının hazırlık tasarımları koruma kapsamında

olmadığı sonucuna ulaşılabilir. Fakat hazırlık tasarımlarından kastın ne olduğu, neleri kapsadığı ve nelerin bir sonraki aşamada bilgisayar programı olacağı konusu hukukumuzda tartışmalıdır.

Suluk ve Orhan (2005), daha ilk safhalardaki hazırlık tasarımlarının korumadan faydalanamayacağı görüşündedir.

Ateş (2003), "hazırlık tasarımı" deyiminin, henüz bilgisayar programı sayılacak düzeyde olmamakla birlikte, bir sonraki safhada bilgisayar programı olma niteliğini kazanacak durumda bulunan bir program yapım tasarımı şeklinde anlaşılması gerektiğini söyler.

Aksu (2006)'ya göre kanun metninin lâfzî olarak değerlendirilmesi halinde "bir sonraki aşamada program sonucunu doğurması koşulu" bilgisayar ortamında işlerlik kazanmasından bir önceki aşama şeklinde anlaşılacağından dar bir içeriğe sahip olacak ve bu durum FSEK'in eser aşamalarının kendi başlarına eser niteliğine sahip olması kaydıyla korunmalarına imkân veren genel yaklaşımına aykırı olacaktır. Yazara göre, bilgisayar programının hazırlık tasarımının eser olarak korunabilmesi için, bu hazırlık tasarımının nitelik yönüyle nihaî olarak amaçlanan bilgisayar programının temel yapısını içermesi ve bu safhadan hareketle, nihaî bilgisayar programının diğer safhalarının oluşturulabilmesi yeterli görülmelidir.

Tekinalp (2005) tasarım halinde kalan bilgisayar programlarının eser olarak korunamayacağını, hazırlık tasarımlarının eser olarak korunabilmeleri için tasarımın bir sonraki aşamasının bilgisayar programı haline gelmesi gerektiği görüşündedir. Yazar devamla, kanundaki "*bir sonraki aşama*" ifadesinin somut olayda belirlenmesi gerektiğini, tasarımın eser olarak korunup, ilk aşamalardaki tasarımın korunmayacağını söyler.

Memiş (2009), Avrupa Konseyi Direktifinin üye ülkelerde aynen kabul edilmediğini, Türk hukukunda direktife uygun yorumlanması için bir şart olmadığını ifade ederek FSEK'teki "*bir sonraki aşamada program sonucu doğurması koşuluyla bunların hazırlık tasarımları*" ifadesinin korumayı daraltıcı mahiyette bulunmadığını; basit,

sıradan veya her tür hazırlık tasarımını bilgisayar programı olarak sayabilecek anlayıştan uzak tutmayı amaçladığını söyler. Yazar FSEK m. 2/I. bendine atıfla hazırlık tasarımlarının belirli bir bütünü ve sistematiği oluşturması halinde bilgisayar programı olarak değil ilmi ve edebi eserler olarak korunabileceğini ifade eder.

Dalyan (2008), FSEK m. 2/I. bentteki ifadenin Avrupa Konseyi Direktifinden yanlış tercüme yapılarak kanunlaştığına, FSEK'te *bir sonraki aşamada* program sonucu olması koşulunun Avrupa Konseyi Direktifindeki gibi geniş bir şekilde *daha sonraki bir aşamada* program sonucu olması şeklinde yorumlanması gerektiğini ifade eder. Yazarın gerekçesi ise, bu durumda bir sonraki aşamada sadece kaynak kodların program oluşturabileceği, bundan önceki taslak materyallerin ve çalışmaların hazırlık tasarımı olarak algılanmaması gerektiğidir. Yazar bu şekildeki yorumların kanunun amaçlamak istediği şeyin dışında olduğunu ifade eder.

Kanaatimizce de FSEK'teki ifadeleri lügat manasıyla değil de, Avrupa Konseyi Direktifinde olduğu gibi yorumlamak gerekmektedir. Zira hazırlık tasarımları ile bir sonraki aşamada kaynak kod olmaksızın program üretilemez. Programın son hali ile hazırlık tasarımı arasında kaynak kod yazımı yer aldığından dolayı kanunun birebir metnine bağlı kalınması halinde, kaynak kodlar "hazırlık tasarımı" olarak ele alınmalıdır ve program yapımına yönelik önceki çalışmaların hiçbirinin hazırlık tasarımı olarak ele alınmaması gerekmektedir. Öte yandan kaynak kod zaten başlı başına koruma altındadır. Bu durumda kanun koyucu hazırlık tasarımı ifadesi yerine doğrudan kaynak kod diyebilirdi. Ayrıca program yazımına yönelik önceki çalışmaların hazırlık tasarımı olarak ele alınmaması yazılım sektörü ve yazılım geliştiricilerinin menfaatine bir durum değildir. Dalyan (2008) böyle bir durumda, hazırlık tasarımlarının yetkisiz bir kimse tarafından alınarak bunların dayanağını oluşturduğu bilgisayar programına benzer, onun kopyası sayılabilecek bir program yapılması halinde ortadaki mağduriyetin çözülemeyeceğini belirtmiştir. Çünkü hazırlık tasarımları koruma kapsamında değildiler. İşte böyle bir mağduriyetin önüne geçebilmek için hazırlık tasarımları bilgisayar programları olarak korunmalıdır (Dalyan, 2008). Özetle hazırlık tasarımları somut halde iseler ve kanun koyucunun eserde aradığı diğer şartları yerine getiriyorlarsa koruma kapsamında yer almalıdırlar. Öte yandan Memiş (2009)'un de belirttiği gibi sıradan hazırlık çalışmalarına koruma sağlamak da gereksizdir.

Hazırlık tasarımı kavramına programın kullanım sırasında faydalanılacak olan kullanma kılavuzu, geliştirmeye ilişkin tutanaklar, kullanıcı el kitapları ve diğer dokümanlar girmemektedir ve dolayısıyla bunlara bir koruma sağlanamaz (Aksu, 2006; Dalyan, 2008).

2.2.8.2. Program Akışı

Akış diyagramları, bir problemin çözümüne yönelik geliştirilen algoritmanın şekil ve sembollerle gösterimidir. Programcı, akış diyagramına bakarak nerede neyi yapacağını kestirir. Program akışı, programın mantıksal olarak izleyeceği yolu da gösterebilir. FSEK'in fikir ve ifade ikileminde ifade tarafında yer alan program akışı, programın hazırlık tasarımına dahil sayılır ve koruma kapsamındadır (Dalyan, 2008).

2.2.8.3. Algoritmalar

Algoritma, bilgisayar programının temel düşünce ve yapısı ile özünü oluşturan programın işleyişine ilişkin teknik, metotsal ifadelerdir (Aksu, 2006). Algoritmalar bilgisayar programına temel oluşturan düşünce ve ilkeleri ihtiva eder. Fikir aşamasında kaldıkları için FSEK çerçevesinde korunmazlar.

Nitekim FSEK m. 2/III'de fikirlerin korunmadığı ifade edilmektedir.

"*Ara yüzüne temel oluşturan düşünce ve ilkeleri de içine almak üzere, bir bilgisayar programının herhangi bir öğesine temel oluşturan düşünce ve ilkeler eser sayılmazlar.*"

Aynı paralelde daha önce değinilen Avrupa Konseyi Direktifi m. 1/II ve WIPO Telif Hakları Andlaşmasının m. 2 "*Telif hakkının korunması, düşünceleri, yöntemleri, uygulama esaslarını ya da matematiksel kavramları değil, ifadeleri kapsar.*" ifadeleri ile fikri korumamaktadır.

Burada algoritmaların niteliği üzerinde de durmak gerekmektedir. Bazı algoritmalar aynen kaynak koda aktarılmayı icap edecek derecede teknik niteliğe haiz olabilirler. Aksu (2006) kaynak koda koruma sağlayan kanunun bu tarz algoritmalara da koruma sağlaması gerektiğini belirtmiştir.

2.2.8.4. Kaynak Kodu

Kaynak kodlar FSEK m. 2/I'de "*her biçim altında ifade edilen bilgisayar programları*" ifadeleri ile koruma altındadır. Bilgisayar programının hazırlık aşamasından son haline gelinceye kadar kaynak kodu, nesne kodu ve makine kodu gibi biçimleri bulunur. Kanun metnindeki *her biçim altında ifade edilen* ibaresinde açık olarak kaynak kod belirtilmemiş, kaynak kod dâhil olmak üzere bilgisayar programının tüm biçimleri koruma kapsamına alınmıştır.

TRIPS anlaşması da m. 10/I. bendinde "*Kaynak veya nesne kodundaki bilgisayar programları Bern Sözleşmesi (1971) kapsamında edebi eserler olarak korunacaktır.*" ifadeleri ile kaynak kodlarına *edebi eserler* statüsünde koruma sağlamaktadır.

WIPO Telif Hakları Andlaşması'nın m. 4 de *"Bilgisayar programları, Bern Sözleşmesi'nin 2 nci maddesi anlamında edebiyat eseri olarak korunur. Öngörülen koruma, hangi koşul ya da biçimle ifade edilirse edilsin bilgisayar programlarına uygulanacaktır."* denilerek *"hangi koşul ya da biçimle ifade edilirse edilsin"* ifadeleri ile kaynak koda koruma sağlamaktadır (WIPO, 1996; TBMM, 2007).

Kaynak kod, fikir bazında kalan algoritmanın ifadeye bürünmüş şeklidir. FSEK'te kaynak kod teriminin yer almaması sorun değildir. Çünkü kanun metnindeki "*her biçim altında ifade edilen*" ifadelerinin sadece nesne kodunu kastetmediği ve kaynak kodu da içerdiği açıktır.

Memiş (2009), kaynak kodların, programa dönüştürülüp dönüştürülmediğine bakılmaksızın sahibinin hususiyetini taşıdıkları ve edebi bir ifadeye dönüştürüldükleri gerekçesiyle eser olarak korunduğunu söyler. Yazar devamla kaynak kodlara sağlanan korumanın bilgisayar programı olarak nitelenen bir koruma olmadığını, kaynak kodların sahibinin hususiyetini taşımaları halinde tamamen bir ilmi eser olarak korunabileceğini söyler.

Kaynak kodların oluşturulması, programlama sürecinin belki de en kritik aşamasıdır. Tezin ilk kısmında da değinildiği gibi kimi programlama dilleri ile oluşturulan kaynak kod derlenen bir yapıdadır ve derlemeden sonra koda bir gereksinim kalmaz. Kaynak

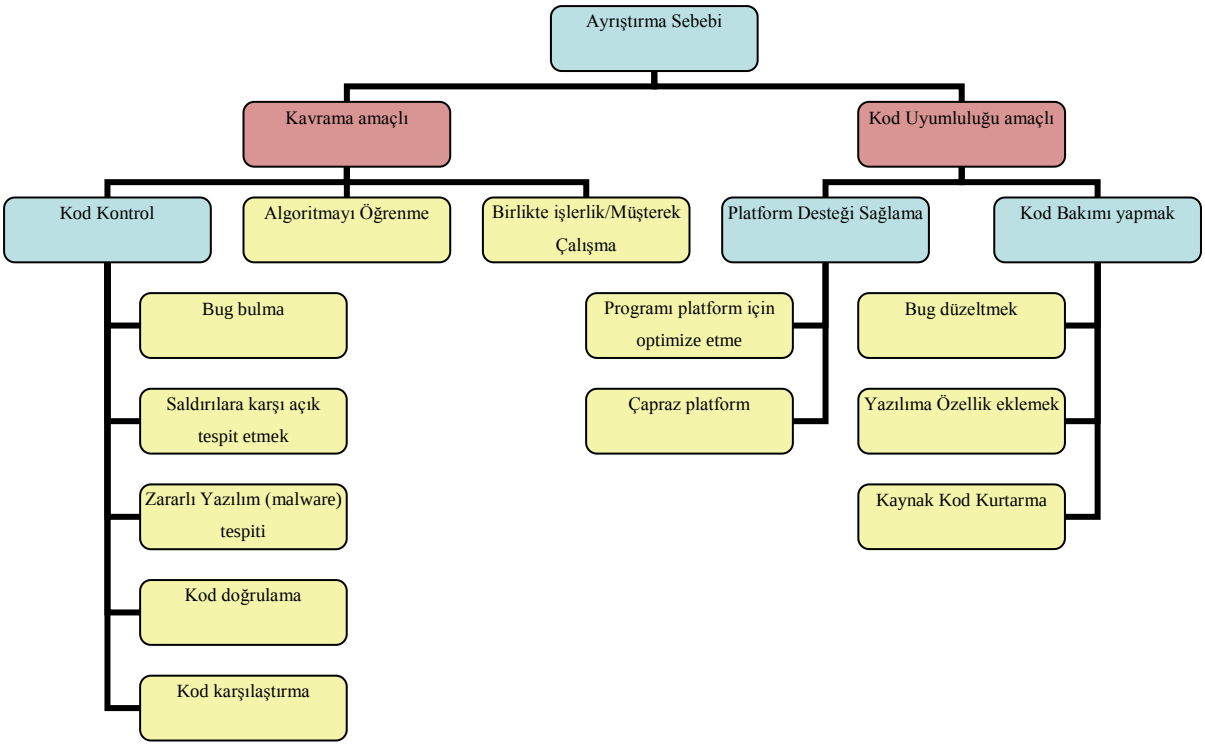
koda sadece programdaki hataları giderme, programa ilave özellikler ekleme, sürüm (versiyon) yükseltme, istenilmeyen kısımları çıkarma gibi değişiklik gerektiren durumlarda başvurulacaktır. Kaynak kodlar bu nedenle son kullanıcıya verilmez ve program geliştiriciler kodu kendilerinde muhafaza ederler. Öte yandan kimi programlama dilleri ile oluşturulan kodlar yorumlanan bir yapıya sahiptir ve o dille yazılan tüm programların çalışması için kaynak koda her seferinde ihtiyaç vardır. Kaynak kod, bu yapıya sahip programlama dillerinde özellikle önem taşımaktadır.

Kaynak kodlarda değişken, modül ve fonksiyon isimlendirmeleri, veri tipi seçimi, fonksiyon ve temel programlama yapılarının kullanımı, kod kısımlarının bölümlenmesi ve sıralanışı, algoritmanın koda uyarlanışı, kodun içinde ilgili satırların ne yaptığını açıklamak için yazılan yorumlar vb. gibi alanlar, özetle programcının süreç esnasındaki kişisel tercihleri programcının programa hususiyetini yansıtabileceği alanlar olarak sıralanabilir. Programın yapısı programcıya hususiyetini yansıtacağı bir biçimlendirme imkânı sağlıyorsa, programcı bu alanlar üzerinde kişisel yaratımını ortaya koyarak programını yazar. Örneğin programın kaynak kodları içerisindeki altprogramların sıralaması keyfi olarak değiştirilebilir. Bu durum programın davranışında bir değişikliğe neden olmaz. Böyle bir durumda programda kaynak kodlar ile programın davranışı birbirine gevşekçe bağlıdır (Davis ve diğ., 1996). Fakat kaynak kodlarda birbirleriyle ilintili komut setleri ve kod satırlarının yer değişiminde programın mantıksal yapısı değişebileceğinden kodun tekrar derlenmesi gerekebilir. En basit durum olarak bu kod bloklarının sıralanması örneği programcının programa hususiyetini katabileceği örnekler arasında sayılabilir.

Tezin esasını oluşturması bakımından kaynak kodların ne kadar özgün olabileceği ve sahibinin hususiyetini ne kadar yansıtabileceği tartışmaya açık bir konudur. Özellikle son yıllarda yazılım sektöründe yazılımların tekrar kullanılabilirliği artmaktadır. Bu aynı zamanda kaynak kodların benzerlik durumlarının ve oranlarının artması anlamına gelmektedir. Bu nedenle bu şekilde tekrar kullanılan kaynak kodlar, sahibinin hususiyetini yansıtmayabilir. Bir başka kaynak kodla benzerlik gösteren kaynak kodların ne kadar özgün olabileceği, teknik bir problemin çözümüne yönelik veya bir formül hesabına dayanan programların sahibinin hususiyetini ne kadar yansıtabileceği tartışmalıdır. Bununla birlikte, benzerlik olmasa da kaynak kodların özgün

olamayabileceği ihtimali de göz ardı edilmemelidir. Şüphesiz bir problemin birden fazla çözümü olabileceği gibi, birden fazla kişi aynı çözüm yolunu izlemiş de olabilir. Fikir ve sanat eserleri hukuku, eserlerde özgün olma gibi bir şart aramamaktadır. Öte yandan her programcının aynı ya da benzer şekilde yapabileceği basit programlar koruma altına alınmazlar (Aksu, 2006). Birden fazla kişinin aynı amaca yönelik olarak yazdıkları kaynak kodlarda benzerlik olması olasıdır. Bu benzerliğin intihale vardığı durumların tespiti ve değerlendirilmesi tezin ilerleyen kısımlarında incelenecektir.

Kaynak kodların kullanımı ile bir başka önemli konu da "ayrıştırma" (*Decompilation – Tersine Mühendislik*) denilen, makine kodunun kaynak koda çevrim işlemidir. Sadece bilgisayarın anlayacağı nesne koduna bakılarak bilgisayar programının arkasındaki fikir ve ilkelere ulaşmak mümkün değildir. Bu nedenle makine kodunun insanın anlayabileceği forma yani kaynak koduna çevrilmesi gerekir. Ayrıştırma, nesne kodu halindeki bilgisayar programı üzerinde tersine mühendislik yöntemlerini ve araçlarını kullanarak programın kaynak kodunu elde etmeye çalışmaktır. Ayrıştırma işlemi, programın korunmayan kısımlarından olan fikre ve ilkelere ulaşmak için yapılabileceği gibi programı üretenler dışında programı kullanma yetkisi olanların o programla uyumlu veya onu tamamlayıcı bir program geliştirmek istemesi gibi meşru sebeplerle de yapılabilir. Ayrıştırma işlemi bilgisayar güvenliği, diğer program veya modüllerle müşterek çalışabilirlik/iş birliktelik (*interoperability*) sağlamak amacıyla, hata düzeltme ve kodu kurtarma için de yapılabilir (Singh, 2009). Bunlara ek olarak diğer ayrıştırma sebepleri Şekil 2.10'da gösterilmiştir.



Şekil 2.10: Ayrıştırma işleminin yapılma sebepleri (Transform Group, 2011)'den uyarlanmıştır.

Ayrıştırma işlemi çok zahmetlidir ve uzun süreç ister. Ayrıca ayrıştırma işlemi ile elde edilen kod, kaynak kodun birebir aynısı olmayıp, onun benzeri ve bir temsilidir (Wang, 2010).

FSEK'te ayrıştırma m. 38'de bilgisayar programı formunu tersine çevirmek şeklinde ele alınmıştır. Kanunun ilk fıkrasında programı analiz etmek ve incelemek amacıyla program üzerinde tersine mühendislik yapmaya izin verilmiştir:

"Bilgisayar programının kullanım hakkına sahip kişinin yapmaya hak kazandığı bilgisayar programının yüklenmesi, görüntülenmesi çalıştırılması, iletilmesi veya depolanması fiillerini ifa ettiği sırada, bilgisayar programının herhangi bir öğesi altında yatan düşünce ve ilkeleri belirlemek amacı ile, programın işleyişini gözlemlemesi, tetkik etmesi ve sınaması serbesttir."

Kanun aynı maddede ayrıştırma işlemine bağımsız yaratılmış program ile diğer programlar arasında araişlerliği sağlamak amacıyla izin vermektedir:

"Bağımsız yaratılmış bir bilgisayar programı ile diğer programların ara işlerliğini gerçekleştirmek üzere gerekli bilgileri elde etmek için, bilgisayar

programının çoğaltılması ve işlenmesi anlamında kod'un çoğaltılmasının ve kod formunun çevirisinin de zorunlu olduğu durumlarda, bu fiillerin ifası aşağıdaki şartların karşılanması halinde serbesttir:"

Bununla birlikte maddenin devamında ayrıştırma işlemin yapılmasına kısıtlamalar getirilmiş, işlemin belirli şartlar altında yapılmasına izin verilmiştir:

"1. Bu fiillerin, ruhsat sahibi veya bir bilgisayar programının kopyasını kullanma hakkı sahibi diğer bir bilgisayar sahibi diğer bir kişi tarafından veya onların adına bunu yapmaya yetkili kişi tarafından ifa edilmesi,

2. Ara işlerliği gerçekleştirmek için gerekli bilginin, (1) numaralı bentte belirtilen kişilerin kullanımlarına sunulmaması,

3. Bu fiillerin, ara işlerliği gerçekleştirmek için gereken program parçaları ile sınırlı olması."

Kanun yapılmasını şarta bağladığı ve yapımını belirli alanlar ile kısıtladığı ayrıştırma işlemi ile elde edilen bilgilerin kullanımına da kısıtlamalar getirmiş ve bunları üç maddede saymıştır:

"*Yukarıdaki fıkra hükümleri, onun uygulanması ile elde edilen bilgilerin;*

1. Bağımsız yaratılmış bilgisayar programının ara işlerliğini gerçekleştirmenin dışında diğer amaçlar için kullanılmasına,

2. Bağımsız yaratılmış bilgisayar programının ara işlerliği için gerekli olduğu durumlar dışında başkalarına verilmesine,

3. İfade ediliş bakımından esastan benzer bir bilgisayar programının geliştirilmesi, üretilmesi veya pazarlanması veya fikri hakları ihlal eden herhangi diğer bir fiil için kullanılmasına, izin vermez."

Ayrıştırma işlemi ile elde edilen kod, her ne kadar programın orijinal kaynak koduna birebir benzemese de programın işleyişi, ilkeleri ve programın algoritması ile ilgili bilgi vermektedir. Bu kodu başka bir programın kaynak koduna olduğu gibi ekleme imkânı yoktur. Yani ayrıştırma ile elde edilen kod, bizatihi kaynak kod olmadığı için başka bir programa eklenemez dolayısıyla doğrudan bir intihal durumu söz konusu olamaz.

Ayrıştırma işlemi ile elde edilen kod, ayrıştıranlar tarafından FSEK veya diğer bağlayıcı anlaşmaların aksine aşağıda sayılan amaçlar için de kullanılmaktadır:

- Ayrıştırma ile elde edilen koda tamamen benzeyen aynı dilde başka bir program geliştirilmektedir.
- Ayrıştırılma ile elde edilen fikir ve algoritmadan istifade edilerek başka bir program geliştirilmektedir. Programın fonksiyonlarının işleyiş mantığı, fikri veya programın algoritması baz alınarak, program başka bir programlama diline veya başka bir platforma çevrilmektedir. Burada kodun tamamen veya kısmen kullanılması söz konusu değildir.

Ayrıştırma ile elde edilen bilgilerden yasamızda sadece araişlerliği sağlamak amacıyla istifade edilebileceği belirtilmişti. Bunun dışındaki kullanımlar bilgisayar programının eser sahibinin haklarına tecavüze girmektedir.

2.2.8.5. Nesne Kodu

Kaynak kodu yazılmış olan bilgisayar programının, makinenin anlayacağı forma çevrilmesinin ardından oluşan nesne kodu hem FSEK çerçevesinde ve hem de TRIPS ve WIPO anlaşmalarına göre eser olarak korunur. Nesne kodunun 1 ve 0'lardan müteşekkil sayılar dizisi olması onun dil eseri olmadığı anlamına gelmez. Nesne kodu, kaynak kodun makinenin anlayacağı dile çevrilmiş halidir. Yani kendisinden bir önceki aşamanın çevrilmiş (derlenmiş/yorumlanmış) halidir. Daha önce değinildiği gibi bu çevrim olayının meydana gelmesi sebebi ile nesne kodunu işlenme eser olarak ele almak doğru değildir. Bu durumda "*işlenme eserin sahibi eseri işleyendir"* ilkesi gereği derleyicilerin yani üzerinde çalıştıkları bilgisayarların eser sahibi olmaları gerekirdi. Oysaki eserin sahibi programı yazandır ve aradaki derleme işlemi işlenme olarak ele alınmamalıdır. Bilgisayar programlarının kendine has yapısı nedeniyle bu çevrim işlemi zorunludur. Burada hususiyeti nesne kodunda değil ona kaynaklık eden kaynak kodda aramak gerekmektedir (Aksu, 2006).

Makine diliyle yazılmış kodlar hiçbir ara işleme (derleme/yorumlama) gerek kalmadan çalıştırılır. Bu nedenle makine kodu, yüksek seviye dillere göre çok daha hızlıdır. Günümüzde bazı işletim sistemlerinin çekirdeğinde donanımla iletişim sağlayan kodların az da olsa bir kısmı, işletim sisteminin daha hızlı olması için nesne kodu ile yazılmaktadır. Ayrıca bilgisayar programlama dillerinin henüz fazla olmadığı

dönemlerde doğrudan makine kodlarıyla program/program kısımları yazılmaktaydı. Fikrî emek sonucu meydana gelen makine kodları da eserin bir görünümü olarak korunurlar. Arada bir çevrim işleminin olmaması nedeniyle bu kodların eser olmayacağı gibi bir durum söz konusu olamaz.

Nesne kodların hangi ortamlarda korunabileceği konusu önemsizdir. Nesne kodlar, bilgisayar üzerinde herhangi bir manyetik, optik cihaz (HDD, DVD), sabit/taşınabilir bellek (ROM, Flash Disk), bir entegre (bütünleşik) devreye gömülü veya kâğıt ortamda kaydedilmiş/taşınabilir olabilir. Kanun bu açıdan bir sınırlama getirmemiştir.

2.2.8.6. Arayüz

Arayüz, FSEK 1/B (h) bendinde şu şekilde tanımlanmıştır.

"Arayüz: Bilgisayarın donanım ve yazılım unsurları arasında karşılıklı etkilenme ve bağlantıyı oluşturan program bölümlerini ifade eder"

FSEK m. 2 son bendi arayüzü koruma sağlanmayan unsurlar arasında saymıştır:

"Ara yüzüne temel oluşturan düşünce ve ilkeleri de içine almak üzere, bir bilgisayar programının herhangi bir öğesine temel oluşturan düşünce ve ilkeler eser sayılmazlar."

2.2.9. Bilgisayar Programlarının Eser Sahipliği

FSEK m. 8'de eser sahibini "*Bir eserin sahibi onu meydana getirendir.*" şeklinde açık bir şekilde tanımlamıştır. Aynı şekilde bilgisayar programlarının sahibi de programı yazanlardır ve eser sahibi olmanın kendilerine sağladığı haklarda münhasıran söz sahibidir. Bu hakların nasıl ve ne şekilde kullanılacağı bilgisayar programının tek kişi veya birden fazla kişinin iştirakiyle üretilmiş olmasına göre değişebilir.

FSEK çerçevesinde bir eser yaratıldığı andan itibaren koruma kapsamına girer. Bu durum aynı zamanda eser sahibine, eserini meydana getirdiği andan itibaren kanunun kendisine verdiği eseri üzerindeki hakları elde etmesi için yeterlidir. Bunun dışında eser sahibinin herhangi bir irade açıklamasına, herhangi bir hukukî işleme veya herhangi bir tescil işlemi yapmasına gerek yoktur (Ateş, 2003). Bilgisayar programı sahiplerinin de eserini herhangi bir yere bildirmesine gerek yoktur. Bilgisayar programı yazıldığı andan itibaren koruma altına alınır.

Ateş (2003) eserin sahibinin küçük veya gayri mümeyyiz (iyiyi kötüden ayıramayan) şahıs dahi olsa eser sahibi olarak kabul edileceğini söyler. Bu prensip bilgisayar programları için de geçerlidir.

Bilgisayar programı tek kişi tarafından yazılmış veya birden fazla kişinin iştiraki ile meydana getirilmiş olabilir. Günümüzde ticarî amaçla yazılan bilgisayar programları, orta ve büyük ölçekteki yazılımlar tek kişiden ziyade birden fazla kişinin katkısı ile meydana getirilmektedir. Bu nedenle bilgisayar programlarında eser sahipliği açısından birden fazla kişinin eser sahibi olduğu durumlar daha fazla önem taşımaktadır.

Kanun eser sahipliğini bilgisayar programları için ayrıca ele almamış, genel olarak tüm eserler için eser sahipliğini açıklamıştır. Bu nedenle konunun daha anlaşılır olması açısından eser sahipliği durumu önce tüm eserler için incelenecek ardından bilgisayar programı bu çerçevede ele alınarak değerlendirilecektir.

2.2.9.1. Tek Kişinin Eser Sahipliği

Tek kişinin eser sahipliğinde eseri bir kişinin meydana getirdiği anlaşılmalıdır ve eserin sahipliği tek bir kişiye aittir (Ateş, 2003). Eseri bir kişi meydana getirmiş olmakla birlikte, eserin meydana getirilmesinde eser sahibi başkalarından yardım almış olabilir. Ateş (2003) yardımcı kişilerin eser sahibi sayılıp sayılmayacağı hususunda yardım eden kişilerin eser üzerinde hususiyet kazandıracak mahiyette bir yardımlarının olup olmadığına bakılması gerektiğini söyler. Yazar Yargıtay'ın eser yapımındaki yardımlarda nicelikten ziyade niteliğe baktığını ifade etmiştir. Özetle, eserin vücut bulmasında eser sahibi esere kendi hususiyetinin dışında hususiyet katacak bir yardım almışsa, eser sahipleri arasına yardımcı kişiler de girecektir. Esere hususiyet katmanın dışındaki basit yardımlar yardım edene eser sahibi olma vasfını kazandırmaz.

Bilgisayar programlarında tek kişinin eser sahibi olması durumu, diğer eser türlerinden farksızdır. Programlama faaliyetleri sonrası programı tek başına yazan kişi, bilgisayar programının eser olarak da sahibidir. Yine diğer eser türlerinde olduğu gibi programı herhangi bir kuruma tescil ettirmesine veya program yazdığını herhangi bir yere bildirmesine gerek yoktur. Programı yazmasıyla birlikte program üzerindeki tüm maddî ve manevî hakların sahibi olur.

2.2.9.2. Birden Fazla Kişinin Eser Sahipliği

Bir eser, yapımında en az iki kişinin çalışmasıyla meydana getirilmişse eser sahipliğinin çokluğundan söz edilir (Ateş, 2003). FSEK birden fazla kişinin eser sahibi olmasına bir kısıtlama veya engel getirmemektedir. Literatürde ve kanunda bu konu m. 9'da ve m. 10/I'de eserin kısımlarına ayrılamaması ve ayrılabilmesine göre *İştirak Halinde Eser Sahipliği* ve *Müşterek Eser Sahipliği* olarak ele alınmaktadır (Ateş, 2003; Karahan ve diğ., 2009).

FSEK m. 9 birden fazla kişinin eser sahibi olmasıyla ilgili durumu açıklamaktadır:

"*Birden fazla kimselerin birlikte vücuda getirdikleri eserin kısımlara ayrılması mümkünse, bunlardan her biri vücuda getirdiği kısmın sahibi sayılır. Aksi kararlaştırılmış olmadıkça, eseri birlikte vücuda getirenlerden her biri bütün eserin değiştirilmesi veya yayımlanması için diğerlerinin iştirakini isteyebilir. Diğer taraf muhik bir sebep olmaksızın iştirak etmezse, mahkemece müsaade verilebilir. Aynı hüküm mali hakların kullanılmasında da uygulanır.*"

FSEK'te bilgisayar programlarının birden çok kişi tarafından meydana getirildiği durumlara ilişkin ayrıca bir ifade yer almamakta, birden çok kişinin meydana getirdiği eser sahipliği durumları bilgisayar programlarını da kapsamaktadır. FSEK bu durumu m. 18/II'de tarif etmiştir:

"Aralarındaki özel sözleşmeden veya işin mahiyetinden aksi anlaşılmadıkça; memur, hizmetli ve işçilerin işlerini görürken meydana getirdikleri eserler üzerindeki haklar bunları çalıştıran veya tayin edenlerce kullanılır. Tüzel kişilerin uzuvları hakkında da bu kural uygulanır.

Bir eserin yapımcısı veya yayımcısı, ancak eserin sahibi ile yapacağı sözleşmeye göre mali hakları kullanabilir."

Böylece birden fazla kişinin meydana getirdiği bilgisayar programlarının eser olarak malî haklarının kullanım yetkisi tüzel kişiye veya işverene verilmiştir (Eroğlu, 2000). Manevî hakları ise sadece eseri geliştiren gerçek kişiler kullanabilirler. Bu hakkın devri ise söz konusu değildir.

Cinoğlu (2010), Fransız Fikrî Hukuk Yasası'nda birden fazla kişinin iştirakiyle meydana getirilmiş bilgisayar programlarının ülkemizdeki durumun aksine m. 113/9'da ayrıca düzenlendiğini ifade etmiştir:

"*Bilgisayar yazılım programlarına ilişkin düzenlemede; görevleri dâhilinde veya işverenlerinin talimatları doğrultusunda bir veya daha fazla çalışan tarafından meydana getirilmiş bilgisayar yazılım programı ve onun dokümanları üzerindeki malî haklara ilişkin mülkiyet hakkı, münhasıran işverene aittir. Bu hüküm aynı zamanda devlet memurları, yerel makamlar ve kamu kuruluşları bakımından da geçerlidir.*"

Metinde de açıkça görüldüğü gibi bir veya daha fazla çalışan tarafından meydana getirilmiş *"bilgisayar yazılım programı ve dokümanları"* ifadeleriyle bilgisayar programlarından ayrıca bahsedilmiş ve bilgisayar programına ilişkin malî hakların kullanım yetkisinin işverene ait olduğunu belirtmiştir.

a. İştirak Halinde Eser Sahipliği

Literatürde, eserin yapımında sarf edilen katkıların birbirinden ayrılmasının mümkün olmadığı duruma *iştirak halinde* eser sahipliği denilmektedir (Ateş, 2003). Yani eser sahiplerinin emekleri sonucu eser, bir bütün olarak meydana gelmektedir. Bu nedenle parçalı olmayan, bir bütün halinde ortaya konan bu emeklerin daha sonra ayrıştırılması söz konusu değildir. Böyle durumlarda eserin sahipleri, eseri meydana getiren birden fazla kimsedir. Her biri aynı ve eşit haklara sahiptir. FSEK böyle durumları "*Eser Sahipleri Arasındaki Birlik*" başlığı altında m. 10'da açıklamıştır:

"*Birden fazla kimsenin iştirakiyle vücuda getirilen eser ayrılmaz bir bütün teşkil ediyorsa, eserin sahibi onu vücuda getirenlerin birliğidir. Birliğe adi şirket hakkındaki hükümler uygulanır. Eser sahiplerinden biri, birlikte yapılacak bir muameleye muhik bir sebep olmaksızın müsaade etmezse, bu müsaade mahkemece verilebilir. Eser sahiplerinden her biri, birlik menfaatlerine tecavüz edildiği takdirde tek başına hareket edebilir.*

Bir eserin vücuda getirilmesinde yapılan teknik hizmetler veya teferruata ait yardımlar, iştirake esas teşkil etmez.

(Ek: 21.2.2001-4630/6) Birden fazla kimsenin iştiraki ile vücuda getirilen eser, ayrılmaz bir bütün teşkil ediyorsa bir sözleşmede veya hizmet şartlarında veya eser meydana getirildiğinde yürürlükte olan herhangi bir yasada aksi öngörülmediği

takdirde birlikte eser üzerindeki haklar eser sahiplerini bir araya getiren gerçek veya tüzel kişi tarafından kullanılır. Sinema eseri ile ilgili haklar saklıdır."

Ateş (2003), iştirak halinde meydana getirilen eserlerde, eser için harcanan emeğin birbirinden ayrılamaz durumda olduğunu, birbirinden ayrılması durumunda eserin niteliğinin bozulacağını ifade ederek her bir eser sahibinin emeğinin eserin bütününden ayrı olarak tek başına telif konusu olamayacağını ve ekonomik olarak değerlendirilemeyeceğini, iştirak halindeki eserlerde tek bir hakkın söz konusu olduğunu ve eseri meydana getirenlerin bu hakka birlikte sahip olduklarını söylemektedir.

Bilgisayar programlarının iştirak halinde yapılmış olması durumu bu çerçevede incelenecek olursa, programın bir bütün olarak ele alınıp, programcıların programın her aşamasında görev aldığı veya kodu kısım kısım, modül modül olarak değil de birlikte yazdığı durumlarda programın eser olarak sahibi programın yapımında emeği geçenlerdir. Böyle durumlarda kanuna göre hiçbir programcı, programın belli bir modülünün veya belli bir kısmının kendisine ait olduğunu iddia edemez, maddî fayda sağlamaya kalkışamaz. Aynı şekilde hiçbir programcı, eser haline gelmiş program üzerindeki hakları tek başına kullanamaz. Bu hakkı diğer programcılarla birlikte kullanır. Yine kanunun söylediği şekilde, bu hakkın kullanımına programcılardan biri haklı bir sebep olmaksızın izin vermezse mahkeme yoluyla bu izin diğer eser sahibi olan diğer programcılara verilebilir.

Kanunun temas ettiği başka bir nokta eser haline gelmiş program üzerinde her emeği olanın eser sahipleri arasında yer alamayacağıdır. Örneğin program bir şirket ortamında üretilmişse ve sözgelimi şirketin yazılımla ilgili olmayan temizlik veya sekreterlik işleriyle uğraşan personeli şirkete olan çalışmasından ötürü şirkette üretilen program üzerinde hak talep edemez, eser sahibi statüsü alamaz. Bununla birlikte eser haline gelmiş bilgisayar programına, program bilgisi olarak yardımcı olan herkes de eser sahibi sayılamaz. Kanunun m. 10/II'de ve Ateş (2003)'ün de daha önce belirttiği gibi, esere hususiyet kazandırmayacak kadar az veya eserin bütününe değil de teferruata ilişkin yardımda bulunanlar eser sahibi olarak anılamazlar.

Bilgisayar programı yapımında sadece programcı veya yazılım geliştiriciler değil, analist, programın testini yapan test ekibi üyeleri, gerektiğinde programa görsel veya işitsel öğe yapan personel vs. de istihdam edilmektedir. Burada ilgilenilmesi gereken nokta program bitiminde programın üretiminde görev alan herkesin eser sahibi olup olamayacağı konusudur. Daha önce de değinildiği gibi programın teferruatına ilişkin yardımlar ve teknik hizmetler esere konu olamazlar.

Kanaatimizce programın test ekibinde yer alanlar eser sahibi sayılmamalıdırlar. Test ekibinin işi, program piyasaya sürülmeden önce ve sonra programdaki hataları, programı geliştiren ekibe bildirmektir. Carnegie Mellon Üniversitesi CyLab Sustainable Computing Consortium'a dayanarak ticarî programlarda her 1000 satır kodda ortalama 20-30 hata olabileceğini raporlamıştır (Coverity.com, 2004). McConnell (2004) de tecrübelere göre endüstri ortalamasının her 1000 satırda 1-25 arasında hata olabileceğini, Microsoft uygulama geliştirme bölümünde her 1000 satırda 10-20 arasında hata oluştuğunu, geliştirmesi tamamlanıp yayımlanan (*release*) programlarda ise her 1000 satırda 0.5 hata ortalamasının olduğunu ifade etmiştir (Schnieder, 2011). Bu veriler ışığında test ekibi üyelerinin emeklerinin teferruata ilişkin olduğu rahatlıkla söylenebilir. Ayrıca test ekibinin raporladığı hataları yine program geliştirme ekibi düzeltmektedir. Bu sebeple test ekibinin eser sahipleri arasında sayılmaması kanaatimizce yerinde olur.

Nitekim Dalyan (2008) eserinin dipnot bölümünde, bu konuyla ilgili olarak İngiltere'de telekomünikasyon araçları ile bunlara ilişkin bilgisayar yazılımları üreten ve pazarlayan davacı Fylde Microsystems Ltd. ile taşınabilir radyolar üreten ve ithal eden davalı Key Radio Systems arasındaki ihlâl davasını örnek olarak verir. Söz konusu davada iki şirket, yeni üretilecek radyolarda kullanılmak üzere bir yazılım tasarımı üzerinde anlaşmaya varıyorlar. Davacı taraf (Fylde Microsystems Ltd), KEYPORT adını verdikleri yazılımda kodu kendilerinin yazdığını beyan ederek, davalının bu kodu lisanssız kopyaladıklarını öne sürmüştür. Davalı taraf, söz konusu yazılımda kod geliştirme işinde görev almadıklarını kabul ederek; hata raporlama, yazılımın monte edileceği donanımla ilgili teknik bilgi verme, hata düzeltme, önerilerde bulunma ve programın test işlerini yaptıklarını beyan ederek iştirak halinde eser sahibi (*joint authorship*) olduğunu ileri sürmüştür. Mahkeme, davalının yaptığı katkıların çok önemli

ve teknik olarak ileri düzeyde olduğunu belirtmekle beraber, bu katkıların, kodların yazılmasına yönelik olmadığı için davalının bu programın iştirak halinde eser sahibi olmadığına hükmederek, yazılımın tek sahibinin davalı Fylde Microsystems Ltd. olduğunu ve davacının kodu lisanssız şekilde kopyaladığına hükmetmiştir (British and Irish Legal Information Institute, 1998).

Kanaatimizce yine aynı şekilde programın analiz ekibinde görev alan üyeler de test ekibi üyeleri gibi eser sahibi sayılmamalıdırlar. FSEK, yukarıda değinildiği gibi bilgisayar programlarının kaynak kod ve amaç kod, araişlerliği sağlayan arayüz gibi parçalarını eser olarak korumaktadır. Bu nedenle analistlerin emeği bu sayılanlar içinde yer almadığından, eser sahibi olarak anılmamalıdırlar.

Kanunun bu maddede temas ettiği bir diğer nokta, iştirak halindeki eserlerin hak sahibinin, tüzel kişiler de olabileceği konusudur. Yine FSEK m. 18/II "*Aralarındaki özel sözleşmeden veya işin mahiyetinden aksi anlaşılmadıkça; memur, hizmetli ve işçilerin işlerini görürken meydana getirdikleri eserler üzerindeki haklar bunları çalıştıran veya tayin edenlerce kullanılır. Tüzel kişilerin uzuvları hakkında da bu kural uygulanır.*" diyerek çalışanların meydana getirdiği eserler üzerindeki hakları, çalışan ve çalıştıran arasındaki sözleşmede aksi bulunmadığı müddetçe çalıştıranlar kullanacağını beyan etmiştir.

FSEK m. 1/B-b'deki tanımlarda eser sahibini "*Eser sahibi Eseri meydana getiren kişiyi; ... ifade eder.*" ile tanımlamıştır. Burada kişiden kasıt gerçek kişidir. Bir eseri gerçek bir kişi veya kişiler meydana getirir. Ve eser sahibi yalnızca gerçek bir kişi veya kişiler olabilir. Tüzel kişilerin bir eser meydana getirmesi söz konusu olamaz. Fakat FSEK m. 10/II ve m. 18/II'de, eser üzerindeki hakların tüzel kişilerce kullanılabileceği ifade edilmiştir. Burada anlaşılması gereken nokta, Ateş (2003)'ün de dediği gibi tüzel kişilerin "eser sahibi" olarak değil de "hak sahibi" olarak anılması gerektiğidir. Yani eser sahiplerinin birlikteliğinde oluşturulan tüzel kişiler, eser sahipleri adına bu hakları kullanırlar.

Ateş (2003), kanunun tüzel kişiyi işveren ve tayin eden, organları oluşturan gerçek kişileri de tüzel kişinin yani işverenin işçisi, memuru veya müstahdemi olarak

değerlendirdiğini ifade etmiştir. Yazar devamla, tüzel kişilerin eser üzerindeki haklara sahip olabilmesi için işgörenin ürettiği eserin işin görülmesi sırasında ve eserin tüzel kişinin faaliyet alanı ile ilgili olması gerekmektedir. Yani işgörenin ürettiği eserler iş sırasında üretilmemişse veya iş ile ilgili değilse tüzel kişi yani işveren bu eserlerin üzerinde hak iddia edemez.

Bu konu bilgisayar programları çerçevesinde değerlendirecek olursa, programcıların iş saatleri dışında kendi bireysel veya grup olarak ürettikleri programlar üzerinde veya çalışma alanı dışında ürettikleri eser niteliğine ulaşmış programlar üzerinde işveren hak iddia edemez.

b. Müşterek Eser Sahipliği

Eser sahiplerinin bireysel katkıları ve yaratım faaliyetleri sonucu birbirinden ayrı ve bağımsız olarak meydana getirdikleri eserlerin daha sonra tek bir bütün halinde eser sahiplerinin iradeleri ile birleştirilmesi ile oluşan eserin sahipliği konusu literatürde *müşterek* eser sahipliği olarak geçmektedir (Ateş, 2003; Dalyan, 2008). Bu tür eserlerde eseri oluşturan her bir parçayı veya her bir eser sahibinin bireysel katkısını hususiyetlerine zarar vermeden birbirinden ayırmak mümkün olabilmelidir (Tekinalp, 2005; Dalyan, 2008). Ayrıca sinema eserinde senaryo ve film müziği gibi eseri oluşturan her bir parça farklı hukukî niteliğe de sahip olabilir (Ateş, 2003).

Kanun koyucu, FSEK m. 9'da bu durumu "*Eser Sahiplerinin Birden Fazla Oluşu*" başlığı altında hükme bağlamıştır:

"*Birden fazla kimselerin birlikte vücuda getirdikleri eserin kısımlara ayrılması mümkünse, bunlardan her biri vücuda getirdiği kısmın sahibi sayılır. Aksi kararlaştırılmış olmadıkça, eseri birlikte vücuda getirenlerden her biri bütün eserin değiştirilmesi veya yayımlanması için diğerlerinin iştirakini isteyebilir. Diğer taraf muhik bir sebep olmaksızın iştirak etmezse, mahkemece müsaade verilebilir. Aynı hüküm mali hakların kullanılmasında da uygulanır.*"

Kanunda da belirtildiği gibi müşterek eserlerin üzerindeki haklar, eser sahiplerinin tümüne aittir. Bu hakkı tüm eser sahipleri birlikte kullanırlar. Bu hakkın kullanılmasına ilişkin kararlar oybirliği ile alınabilir. Dalyan (2008), bu haklardan kastın sadece malî

haklar olmayıp manevî hakları da eser sahiplerden birinin diğer eser sahiplerinin rızasını alarak kullanabileceğini ifade eder.

İştirak halindeki eser sahipliğinde olduğu gibi, müşterek eser sahipliğinde de eser sahiplerinin katkısının yoğunluğu, niteliği veya ekonomik katkısı önemli değildir (Dalyan, 2008).

Müşterek eser sahipliğinde, birlik üyelerinin maddî menfaati ile ilgili olarak, eser sahipleri arasında yapılan sözleşme geçerli olur. Eser sahiplerinden birisi razı gelmezse programını yeni oluşan programdan çekebilir, anlaşmaya bağlı olarak kullanılmasını men edebilir, yayımlanmasını durdurabilir. Eserin tek taraflı feshedilmesi veya sözleşmeden dönülmesi gibi durumlarda Tekinalp (2005) ve Karahan ve diğ. (2009) Borçlar Kanununun genel hükümleri ile adi ortaklık (adi şirket) hükümlerinin uygulanacağını söyler.

Müşterek eser sahipliğinin bilgisayar programlarına bakan yönü, iştirak halindeki eser sahipliği gibidir. Birbirinden ayrı ve eser niteliğine haiz bilgisayar programları eser sahiplerinin rızasıyla yeni bir bilgisayar programı oluşturmak için kullanılabilir. Bu tarz kullanım daha çok açık kaynak kod projelerde görülür. Daha önce bir alanla ilgili yazılmış bir açık kaynak kod program, betik veya kütüphane lisansı buna izin veriyor ise başka bir programda aynı işi görmek için kullanılabilir. Ticarî yazılımlarda da, bağımsız programlar kullanılarak eser sahiplerinin rızası ile yeni bir program oluşturulabilir.

2.3. KAYNAK KOD İNTİHALİNDE ESER SAHİPLERİNİN HUKUKSAL HAKLARI

Kanun her türlü intihal suçuna karşı eser sahibinin haklarını koruma altına almıştır. İntihal mağduru eser sahipleri, haklarını aramak için ceza ve hukuk davası açabilirler (Telif Hakları ve Sinema Genel Müdürlüğü, 2007; Ceritoğlu, 2008). Hak sahiplerinin birlikte yada münferit olarak intihale yönelik açabilecekleri davalar, hukuk davaları, tespit davaları (eser sahibinin tespiti davası, tecavüzün tespiti davası), eda davaları (tecavüzün ref'i davası, tecavüzün men'i davası, maddî ve manevî hakların ihlali veya

tecavüzü halinde şahsî, maddî ve/veya manevî tazminat davaları) ve ceza davalarıdır (Gündem, 2006; Telif Hakları ve Sinema Genel Müdürlüğü, 2007; Ceritoğlu, 2008). Bunun yanında eser sahibi, tecavüz fiilinde kullanılan malzemelere, cihazlara, makinelere el konulmasını, mahkeme masraflarının karşılanmasını, kararın ilgililere tebliğ edilmesini ve kamuya duyurulmasını talep edebilir (Coşgun, 2006).

Bu davalara bakmakla görevli mahkemenin Fikrî ve Sınaî Haklar Hukuk Mahkemeleri ve Fikrî ve Sınaî Haklar Ceza Mahkemeleri olan İhtisas Mahkemeleri olduğu Kanun m. 76'da tarif edilmiştir. Kanun, İhtisas mahkemelerinin yokluğu halinde Adalet Bakanlığı'nın teklifi üzerine Hakimler ve Savcılar Yüksek Kurulu'nun belirleyeceği Asliye Hukuk ve Asliye Ceza Mahkemelerinden birinin görevlendirileceğini de ayrıca açığa kavuşturmuştur.

2.3.1. Hukuk Davaları

2.3.1.1. Tespit Davaları

Hak sahipleri, intihal yapıldığını kanıtlamak amacıyla *Tecavüzün Tespiti Talebi*nde bulunarak dava açabilir. Ayrıca eserin kime ait olduğunun bilinmediği veya esere birden fazla hak talep eden bulunması durumunda eserin sahibinin belirlenmesi amacıyla *Eser Sahibinin Tespiti* davası da açılabilir.

a. Eser Sahibinin Tespiti Talebi

Kanunun eser sahibine tanıdığı manevî haklardan biri "*Adın belirtilmesi salahiyeti*"dir. Eserin sahibi eserinde adını belirtmekten men edilemez. Eser sahibi, isterse eserde adını belirtmek istemeyebilir veya adını tam olarak değil rumuz olarak da belirtebilir. Bu hak eser sahibinin tekelindedir, istediği gibi kullanabilir. Kanun m. 15 bu durumu

"Eseri, sahibinin adı veya müstear adı ile yahut adsız olarak, umuma arzetme veya yayımlama hususunda karar vermek salahiyeti münhasıran eser sahibine aittir."

ile ortaya koymuştur.

İntihal olayında eserin sahibi zaten bellidir. İntihal, zaten var olan bir eserden sahibinin adı belirtilmeden yararlanma yani eserin veya eserin bir kısmının sahiplenilmesi fiilidir (Uygur, 2004). Bu nedenle intihalde "eser sahibinin eserde adının belirtilmesi salahiyetine" tecavüz söz konusudur. Yani eser bir kişiye ait olduğu halde aslında öyle

olmasa da aynı eserin birden fazla sahibi görünmekte veya bir başkası eserin sahipliğini iddia etmektedir. Bu nedenle eserin asıl sahibi, eserinin kendisine ait olduğu ve başkası tarafından intihal edilerek sahiplenildiği gerekçesiyle bu davayı açabilir. Kanun eser sahipliğinin tespitinin mahkemeden istenebileceğini m. 15/III'te belirtmiştir:

"Bir eserin kimin tarafından vücuda getirildiği ihtilaflı ise, yahut herhangi bir kimse eserin sahibi olduğunu iddia etmekte ise, hakiki sahibi, hakkının tespitini mahkemeden isteyebilir."

Kanun metninden de anlaşılacağı gibi sahibi ihtilaflı veya eser üzerinde birden fazla sahiplik iddia edilen durumlarda eser sahipliğinin tespiti mahkemeden istenebilir.

Bununla birlikte eserde sahibinin adı hiç belirtilmemişse eser sahibi Kanun m. 67/II'ye göre bu durumun düzeltilmesini isteyebilir:

"Eser üzerinde sahibinin adı hiç konulmamış veya yanlış konulmuş yahut konulan ad iltibasa meydan verecek mahiyette olup da eser sahibi 15 inci maddede zikredilen tespit davasından başka tecavüzün ref'ini talep etmişse, tecavüz eden gerek aslına, gerek tedavülde bulunan çoğaltılmış nüshalar üzerine eser sahibinin adını derç etmeye mecburdur. Masrafı tecavüz edene ait olmak üzere, hükmün en fazla 3 gazetede ilan talep edilebilir."

Bu davanın sonucu, intihali kapsamamaktadır (Gündem, 2006). Davanın sonucu intihalin yapılıp yapılmadığıyla ilgili değil, intihale söz konusu eserin sahibinin tespit edilmesiyle ilgilidir. Bu dava sonucunda eserin sahipliğinin kimde olduğu belirlenecektir (Gündem, 2006).

b. Tecavüzün Tespiti Talebi

Eser sahibi, eserinin intihal edilmesi halinde mevcut eylemin bir eser sahibi olarak haklarına bir saldırı olup olmadığına yönelik Türk Ticaret Kanunu m. 58/a ve Türk Medeni Kanunu m. 25'e dayanarak mahkemeden tespit isteyebilir (Ceritoğlu, 2008; Tuncer, 2010). Bu dava tek başına açılmamakta, tazminat veya eski hale getirme gibi bir eylemin yapılmasını gerektiren eda davasının açılabileceği hallerde mahkemeden tecavüzün tespiti istenebilmektedir (Gündem, 2006).

Bu dava için zamanaşımı söz konusu değildir (Gündem, 2006). Eserde sahibinin adının belirtilmesi hakkı manevî haklardan olduğu için zamanaşımına uğraması söz konusu değildir. Bu nedenle tecavüzün tespitinin istenmesi talebinin, intihalden sonra belli bir müddet içinde yapılması şart değildir (Gündem, 2006).

Tecavüzü tespit davası, tecavüzün sona erdiği fakat tecavüzün etkilerinin devam ettiği durumda da açılır (UPB, 2011).

2.3.1.2. Tecavüzün Ref'i (Kaldırılması) Davası

Kanun, eser sahibinin manevî ve malî haklarına yönelik tecavüzlerde eser sahibinin açabileceği hukuk davalarını sayarken m. 66, 67, 68 "Tecavüzün Ref'i Davası"na yer vermiştir. Söz konusu dava mevcut ve devam etmekte olan tecavüzün ve tecavüz ile ortaya çıkan sonuçların ortadan kaldırılması amaçlamaktadır (Suluk, 2004; Gündem, 2006).

Kanun m. 66'da eser sahiplerinin intihali yapanlara karşı uğramış oldukları maddî veya manevî mağduriyeti gidermek adına dava açabileceklerini beyan etmiştir:

"Manevi ve mali hakları tecavüze uğrayan kimse tecavüz edene karşı tecavüzün ref'ini dava edebilir."

Kanun devamla tecavüz halinde bu davanın açılması için kusur aranmayacağını belirtmiştir. Yani intihal vakalarında intihali yapan kimselerin bir kusuru yada kastı olmasa da intihal mağduru eser sahibi tecavüzün kaldırılması için dava açabilir:

"Tecavüz edenin veya ikinci fıkrada yazılı kimselerin kusuru şart değildir."

Ancak tecavüzden doğan mağduriyetle ilgili tazminat talep edilecekse kusurun varlığı gereklidir (Gündem, 2006).

Devam eden fıkrada da mahkemenin intihalin kaldırılması için her türlü tedbiri alabileceği açıklanmıştır:

"Mahkeme, eser sahibinin manevi ve mali haklarını, tecavüzün şümulünü, kusurun olup olmadığını, varsa ağırlığını ve tecavüzün ref'i halinde tecavüz edenin düçar olması muhtemel zararları takdir ederek halin icabına göre tecavüzün ref'i için lüzumlu göreceği tedbirlerin tatbikatına karar verir."

Kanun, uğranılan tecavüz manevî haklara yönelik ise, eser sahiplerinin buna karşı neler yapabileceklerini de m. 67'de sıralamıştır:

"Henüz alenileşmemiş bir eser sahibinin rızası olmaksızın veya arzusuna aykırı olarak umuma arz edildiği takdirde tecavüzün ref'i davası, ancak umuma arz keyfiyetinin çoğaltılmış nüshaların yayımlanması suretiyle vaki olması halinde açılabilir. Aynı hüküm, eser sahibinin arzusuna aykırı olarak adının konulduğu hallerde de caridir."

Kanun, henüz alenileşmemiş bir eserin sahibinin rızası olmadan veya rızasına aykırı olarak kullanılması durumunda, eser sahibinin dava açabilmesi için çalıntı eserin çoğaltılmış nüshalarının kamuya sunumunun gerçekleşmesi şartını koymuştur. Tekinalp (2005) eserin çoğaltılma ve yayımlanma dışında sergilenme, seslendirme, temsil, radyo-televizyon ile yayınlanma vb. yollarla kamuya sunulması hallerinde de, tecavüzün ref'i davasının açılabileceğini kabul etmiştir (Uygur, 2004).

Uygur (2004) henüz kamuya sunulmamış bir eserden ancak sahibinin açık izni ile iktibas yapılabileceğini, izin alınmaksızın alenileşmemiş bir eserin belli bölüm ya da parçalarının başka bir eserde kullanılarak yayımlanması halinde yapılan iktibasın hukuka aykırı bir durum oluşturacağını ve de eser sahibinin umuma arz yetkisinin ihlâl edilmiş olacağını söyler.

Bilgisayar programlarının kaynak kodları da aynı şekilde sahibinin rızası dışında bir edebi eserde yayımlanırsa, eser sahibi yayının durdurulması için tecavüzün ref'i davasını açabilir. Buna Microsoft Corp. ile Shuuwa System Trading K.K. arasındaki dava örnek olarak verilebilir (SOFTIC, 2006; Wang, 2010). Davalı Shuuwa System Trading K.K. (Shuuwa) davacı Microsoft Corp. (Microsoft) firmasına ait BASIC yorumlayıcısının kaynak kodlarını terine mühendislik yöntemleri ile ayrıştırmış ve bu kodları, yayımcılığını yine kendilerinin yaptığı "Pc-8001 Basic Source Program Listings The Whole Analysis Of Ver. 1.0 And 1.1" isimli kitapta yayımlamıştır. Tokyo Bölge Mahkemesi, davanın sonucunda davalı Shuuwa'nın telif hakkı ihlali yaptığına hükmederek davalıdan söz konusu kitabın yayımının ve dağıtımının durdurmasını ve mahkeme masraflarının karşılanmasını istemiştir.

Kanun m. 67/III. fıkrada "*yanlış veya kifayetsiz kaynak tasrih edilmiş veyahut hiç kaynak gösterilmemişse"* diyerek intihalden bahsetmiş ve intihal durumunda uygulanacak hükmü aynı maddenin ikinci fıkrası olarak tespit etmiştir.

"32, 33, 34, 35, 36, 39 ve 40 ıncı maddelerde sayılan hallerde ***yanlış veya kifayetsiz kaynak tasrih edilmiş veyahut hiç kaynak gösterilmemişse*** *ikinci fıkra hükmü uygulanır."*

Söz konusu ikinci fıkra, kaynak kod intihali durumunda eser sahibinin neler talep edebileceğini saymaktadır. Bunlar Kanun'da tespit davası açmak, yayımlanan bilgisayar programında asıl eser sahibinin adının yer alması ve mahkeme sonucunun en fazla 3 gazetede yayımlanması şeklinde belirlenmiştir.

"... eser sahibi 15 inci maddede zikredilen tespit davasından başka tecavüzün ref'ini talep etmişse, tecavüz eden gerek aslına, gerek tedavülde bulunan çoğaltılmış nüshalar üzerine eser sahibinin adını derç etmeye mecburdur. Masrafı tecavüz edene ait olmak üzere, hükmün en fazla 3 gazetede ilan talep edilebilir."

Kanun, intihal mağduru hak sahibinin malî haklarına yönelik tecavüzde neler talep edebileceğini de m. 68'de "Mali Haklara Tecavüz Halinde" başlığı altında tespit etmiştir. Anılan maddenin ilk fıkrası intihal edilen kaynak kodların başka bir programlama diline çevrilmesi durumunu da içermektedir. Kanun metninde bilgisayar programı ve kaynak kod terimleri geçmese de eser kelimesi bunları da içine almaktadır. Söz konusu fıkra:

*"**Eser, eser sahibinin izni olmadan çevrilmiş**, sözleşme dışı veya sözleşmede belirtilen sayıdan fazla basılmış, diğer biçimde işlenmiş veya radyo-televizyon gibi araçlarla yayınlanmış veya temsil edilmiş ise; izni alınmamış eser sahibi, sözleşme yapılmış olması halinde isteyebileceği bedelin veya emsal veya rayiç bedel itibarıyla uğradığı zararın en çok üç kat fazlasını isteyebilir. Bu bedelin tespitinde öncelikle ilgili meslek birliklerinin görüşü esas alınır."*

Burada intihal edilen kaynak kodun sahibine, karşı tarafla arasında sözleşme varmış gibi davranarak sözleşmedeki telif ücretinin üç kat fazlasını talep etme haklarını tanımıştır. Burada faraza bir sözleşmeden bahsedilmektedir. Taraflar arasında sözleşme olmasa bile asıl eser sahibinin eserini her hangi birine sözleşme ile kullanım izni vermesi

durumunda ne istiyorsa o bedelin üç mislini kaynak kodlarını intihal eden taraftan isteyebileceği Kanun tarafından kararlaştırılmıştır. Kanun bu bedelin ne kadar olacağını ilgili meslek kuruluşlarının görüşü ile belirleneceğini de hükme bağlamıştır.

Kanun intihal edilen kaynak kodun program olarak çoğaltılması durumunda izinsiz çoğaltılan bilgisayar programı kopyalarının ve bunların üretiminde kullanılan ekipmanın imhasını veya asıl eser sahibine bedeli karşılığında verilebileceğini veya eser sahibinin taraflar arasında bir sözleşme varmışçasına bu sözleşme bedelinin üç kat fazlasını talep edebileceğini m. 68/II ve m. 68/III'de belirtmiştir.

"Bir eserden izinsiz çoğaltma yolu ile yarar sağlanıyorsa ve çoğaltılan kopyaları satışa çıkarılmamışsa, eser sahibi; çoğaltılmış kopyaların, çoğaltmaya yarayan film, kalıp ve benzeri araçların imhasını veya maliyet fiyatını aşmamak üzere çoğaltılmış kopyaların ve çoğaltmaya yarayan film, kalıp ve benzeri gereçlerin uygun bir bedel karşılığında kendisine verilmesini ya da sözleşme olması durumunda isteyebileceği miktarın üç kat fazlasını talep edebilir. Bu husus, izinsiz çoğaltma yoluna giden kişinin yasal sorumluluğunu ortadan kaldırmaz.

Bir eserin izinsiz çoğaltılan kopyaları satışa çıkarılmışsa veya satış haksız bir tecavüz oluşturuyorsa, eser sahibi tecavüz edenin elinde bulunan nüshalar hakkında ikinci fıkrada yazılı şıklardan birini seçebilir.

Bedel talebinde bulunan kişi, tecavüz edene karşı onunla bir sözleşme yapmış olması halinde haiz olabileceği bütün hak ve yetkileri ileri sürebilir."

Kanun bu son fıkra ile asıl eser sahibinin taraflar arasında sanki bir sözleşme varmışçasına hareket edebileceğini ve buna dayanarak hak talep edebileceğini "*bir sözleşme yapmış olması halinde haiz olabileceği"* ifadeleri ile beyan etmiştir. Örneğin sözleşmede karşı tarafa cezaî yaptırımlar öngören maddelerin varlığı da bu fıkra gereğince kabul edilecek ve asıl eser sahibi sadece sözleşme bedelinin üç mislini değil, cezaî yaptırım bedelini ve karşı tarafı bağlayıcı diğer maddelerin tazminini de istenebilecektir.

2.3.1.3. Tecavüzün Men'i (Önleme) Davası

Tecavüzün önlenmesi davası, eser sahibinin haklarına herhangi bir tecavüz veya tecavüz tehlikesinin bulunduğu hallerde ve aynı zamanda tecavüzün devamı veya tekrarı muhtemel olan hallerde tecavüzü veya tehlikesini ortadan kaldırmak veya önlemek

amacıyla açılabilir (Suluk, 2004; Uygur, 2004). Kanun m. 69'da bunu hükme bağlamıştır:

"Mali veya manevi haklarında tecavüz tehlikesine maruz kalan eser sahibi muhtemel tecavüzün önlenmesini dava edebilir. Vaki olan tecavüzün devam veya tekrarı muhtemel görülen hallerde de aynı hüküm caridir. 66 ncı maddenin ikinci, üçüncü ve dördüncü fıkralarının hükümleri burada da uygulanır."

Bu davanın açılabilmesi için tecavüzün ref'i davasında olduğu gibi kusur ve zamanaşımı aranmaz (Suluk, 2004; Ceritoğlu, 2008).

Tecavüzün ref'i davasında, işlenmiş bulunan tecavüz fiilinin sonuçlarının ortadan kaldırması amaçlanmışken, tecavüzün men'i davasında tecavüz tehlikesinin bulunması halinin tecavüze dönüşmesinin engellenmesi amaçlanmıştır (Suluk, 2004; Ceritoğlu, 2008).

2.3.1.4. Tazminat Davaları

İntihalin tanımı gereği eser sahibinin adı belirtilmediğinden eser sahibinin manevî haklarından biri olan "*Adın belirtilmesi salahiyeti*" çiğnenmiş olur. Kanun m. 15/I. fıkrada eserde sahibinin adının yer alıp almayacağını, adın kısaltma, müstear ad veya başka herhangi bir isimle yayınlanıp yayınlamayacağı hakkını sadece eser sahibine tahsis etmiştir.

"Eseri, sahibinin adı veya müstear adı ile yahut adsız olarak, umuma arzetme veya yayımlama hususunda karar vermek salahiyeti münhasıran eser sahibine aittir."

Bununla birlikte eser sahibinin manevî haklarından bir diğeri olan "*Umuma Arz Salahiyeti*" de çiğnenmiş olmaktadır. Kanun m. 14/I. fıkrada eser sahibinin, eserini yayınlayıp yayınlamamakta veya istediği zaman ve istediği şekilde yayınlamakta tek söz sahibi kimse olduğunu beyan etmiştir.

"Bir eserin umuma arz edilip edilmemesini yayımlanma zamanını ve tarzını münhasıran eser sahibi tayin eder."

İntihal olayında eser, sahibinden izinsiz başka bir isimle umuma arz edilmekte ve böylece eser sahibine ait olan bu hak intihali yapanlar tarafından tecavüze uğramaktadır.

İntihalde ayrıca eser sahibinin Kanun m. 16'da tarif edilen "*Eserde Değişiklik Yapılmasını Menetme*" hakkı da çiğnenmiş olur.

"Eser sahibinin izni olmadıkça eserde veyahut eser sahibinin adında kısaltmalar, ekleme ve başka değiştirmeler yapılamaz."

İntihalde tezin önceki bölümlerinde değinildiği üzere, eserin olduğu gibi üzerinde hiçbir değişiklik yapılmadan sadece eser sahibinin adını değiştirerek yayımlanması söz konusu olduğu gibi eserin çok büyük bir kısmının kopyalanması ve üzerinde ufak değişiklikler ve eklemeler yapılarak aynı veya başka bir isimle yayımlanması da söz konusu olmaktadır. Bu durum Kanun m. 16'da eser sahiplerine tanınan eserde değişiklik yapılmasını men etme hakkının gaspı demektir.

Kanun m.70'de eser sahibinin manevî haklarına tecavüz halinde manevî tazminat davası açma hakkının olduğunu açıklamıştır. Ceritoğlu (2008), FSEK uygulaması açısından manevî tazminat davası açan eser sahibinin hangi manevî haklarının ihlal edildiğini özellikle belirtilmesi gerektiğini ifade etmiştir. Kanun m.70'de mahkemenin para yerine takdir ettiği başka bir cezaya hükmedebileceğini de ifade etmiştir.

"Manevi hakları haleldar edilen kişi, uğradığı manevi zarara karşılık manevi tazminat ödenmesi için dava açabilir. Mahkeme, bu para yerine veya bunlara ek olarak başka bir manevi tazminat şekline de hükmedebilir."

Kanun metninde geçen "*para yerine veya bunlara ek olarak başka başka bir manevi tazminat"* ifadeleri ile eser sahibinin talebi doğrultusunda manevî cezalar da verilebileceği anlaşılmaktadır.

Ceritoğlu (2008) manevî haklara tecavüz halinde açılabilecek tazminat davasının zamanaşımının süresinin Borçlar Hukuku'na göre değerlendirileceğini fakat Ceza kanununun daha uzun bir süre öngörmesi durumunda ona uyulacağını belirtmiştir.

Manevî tazminat istemi ancak eser sahibinin manevî haklarına zarar gelmesi durumunda açılabilir (Ceritoğlu, 2008). Manevî tazminat davası açabilmek için kusur şartı aranmaz. Bu davayı eser sahibi veya Kanun m. 19/I ve m. 19/II'de sayılan kimseler açabilir. Yazar FSEK uygulaması açısından manevî tazminat davası açan eser sahibinin hangi manevî haklarının ihlal edildiğini özellikle belirtmesi gerektiğini ifade etmiştir.

Kanun m. 70/II. fıkrada malî yönden zarar gören intihal mağduru eser sahiplerinin tazminat davası açabileceğini beyan etmiştir.

"Mali hakları haleldar edilen kimse, tecavüz edenin kusuru varsa haksız fiillere müteallik hükümler dairesinde tazminat talep edebilir."

Ceritoğlu (2008) bu tazminat talebi için kusur şartının arandığını ve Borçlar Kanunu'nda yer alan haksız fiile ilişkin hükümlerin kıyasen uygulanacağını belirtmiştir.

Kanun intihal mağduru eser sahibinin maddî tazminat talebi dışında mağduriyet süresince intihali yapan kimselerin elde ettiği kârı da isteyebileceğini m. 70/III'te hükmetmiştir.

"Birinci ve ikinci fıkralardaki hallerde, tecavüze uğrayan kimse tazminattan başka temin edilen karın kendisine verilmesini de isteyebilir. Bu halde 68 inci madde uyarınca talep edilen bedel indirilir."

Burada ayrıca geri istenen kârın m. 68'de ceza olarak istenebilecek olan sözleşme bedelinin en fazla üç katı olan bedelden çıkarılmasını da münasip görmüştür.

2.3.2. Ceza Davaları

Kanun, eser sahiplerinin manevî haklarını ihlal eden kimselere karşı hukuk davalarının yanı sıra ayrıca ceza davaları açabileceğini de belirtmiştir. Kanun m. 71/III'te ve 71/IV'te bu duruma ilişkin hüküm bulunmaktadır.

"3. Başkasının eserini kendi eseri veya kendisinin eserini başkasının eseri olarak gösteren veya 15'nci maddenin ikinci fıkrası hükmüne aykırı hareket eden,

4. 32, 33, 34, 35, 36, 37, 39 ve 40 ıncı maddelerdeki hallerde kaynak göstermeyen veya yanlış yahut kifayetsiz veya aldatıcı kaynak gösteren,

...

Kişiler hakkında, iki yıldan dört yıla kadar hapis veya ellimilyar liradan yüzellimilyar liraya kadar ağır para cezasına veya zararın ağırlığı dikkate alınarak her ikisine birden hükmolunur."

Kanun metninden de açıkça anlaşılacağı gibi "başkasının eserini kendi eseri olarak göstermek" fiili ve "kaynak belirtmeden alıntı yapmak veya yetersiz veya aldatıcı kaynak göstermek" intihal suçunun tanımları arasındadır ve bu suça ilişkin cezanın 2-4

yıl arası hapis, 50-150 milyar lira arası ağır para cezası veya her ikisinin birden olacağı açıkça belirtilmiştir.

FSEK dışında ayrıca 765 sayılı Türk Ceza Kanunu m. 525/a bendinde programların, verilerin veya diğer herhangi bir unsurun hukuka aykırı olarak ele geçirilmesi ile ilgili bir hüküm bulunmaktadır:

"Madde 525/a – Bilgileri otomatik olarak işleme tabi tutmuş bir sistemden, programları, verileri veya diğer herhangi bir unsuru hukuka aykırı olarak ele geçiren kimseye bir yıldan üç yıla kadar hapis ve birmilyon liradan onbeşmilyon liraya kadar ağır para cezası verilir.

Bilgileri otomatik işleme tabi tutmuş bir sistemde yer alan bir programı, verileri veya diğer herhangi bir unsuru başkasına zarar vermek üzere kullanan, nakleden veya çoğaltan kimseye de yukarıdaki fıkrada yazılı ceza verilir."

2.3.3. Kanunda Verilen Diğer Haklar

Kanun m. 77'de intihal mağduru eser sahibi mütecaviz kimselere karşı ihtiyati tedbir kararı aldırabilir, mahkemenin verdiği mahkûmiyet hükmünün ilânını isteyebilir ve yolsuz çoğaltılmış eser nüshalarıyla çoğaltılmaya yarayan kalıp vb. nin zabıt, müsadere ve imhasını isteyebilir. Haksız çoğaltılan eser nüshalarına gümrüklerde el konulmasını isteyebilir.

"Madde 77 - (Değişik: 21.2.2001-4630/31) Esaslı bir zararın veya ani bir tehlikenin veya emrivakilerin önlenmesi için veya diğer herhangi bir sebepten dolayı zaruri ve bu hususta ileri sürülen iddialar kuvvetle muhtemel görülürse mahkeme, bu Kanunla tanınmış olan hakları ihlal veya tehdide maruz kalanların veya şikayete selahiyetli olanların talebi üzerine, davanın açılmasından önce veya sonra diğer tarafa bir işin yapılmasını veya yapılmamasını, işin yapıldığı yerin kapatılmasını veya açılmasını emredebileceği gibi, bir eserin çoğaltılmış nüshalarının veya hasren onu imale yarayan kalıp ve buna benzer sair çoğaltma vasıtalarının ihtiyati tedbir yolu ile geçici olarak zaptına karar verebilir. Kararda emre muhalefetin İcra ve İflas Kanununun 343 üncü maddesindeki cezai neticeleri doğuracağı tasrih edilir.

Haklara tecavüz oluşturulması ihtimali halinde yaptırım gerektiren nüshaların ithalat veya ihracatı sırasında, 4458 sayılı Gümrük Kanununun 57 nci maddesi ile 4926 sayılı Kaçakçılıkla Mücadele Kanununun ilgili hükümleri uygulanır.

Bu nüshalara gümrük idareleri tarafından el konulmasına ilişkin işlemler Gümrük Yönetmeliğinin ilgili hükümlerine göre yürütülür."

Kanun ayrıca intihalde mağdur olan eser sahibinin mahkeme hükmünün gazetelerde ilanını isteyebileceğini m. 78'de belirtmiştir.

"Madde 78 - 67 nci maddenin ikinci fıkrasında yazılı halden maada, haklı olan taraf, muhik bir sebep veya menfaati varsa, masrafı diğer tarafa ait olmak üzere, kesinleşmiş olan kararın gazete veya buna benzer vasıtalarla tamamen veya hulasa olarak ilan edilmesini talep etmek hakkını haizdir.

İlanın şekil ve muhtevası kararda tespit edilir.

İlan hakkı, hükmün kesinleşmesinden itibaren üç ay içinde kullanılmazsa düşer."

Kanun ayrıca intihal sonucu ortaya çıkan ürünlerin basımına ve çoğaltılmasına yarayan her türlü araç, gereç, bilgisayar vb. vasıtalara el konulmasının ve imhasının istenebileceğini m. 79'da hükmetmiştir.

"Madde 79 - Bu Kanun hükümlerine göre imali veya yayımı cezayı mucip olan çoğaltılmış nüshalarla bunları çoğaltmaya yarayan kalıp ve buna benzer vasıtaların zabıt, müsadere ve imhasında Ceza Kanununun 36 ncı maddesi hükümleriyle Ceza Muhakemeleri Usulü Kanununun 392, 393 ve 394 üncü maddeleri uygulanır."

2.4. KAYNAK KOD İNTİHALİ

2.4.1. İntihal

TDK (2011) intihali (*plagiarism*) aşırma olarak tanımlamaktadır. Türk Hukuk Lügatinde ise intihal terimi "*başkasına ait bir telifi, güzel sanatlardan bir eseri kendisine nispet etmek; bir kitabın ibarelerini, musiki bestenin nağmelerini, takdim ve tehir veyahut aslının baştanbaşa hissolunur derecede ifade tarzını tahrif ile kendi namını vermek."* olarak tanımlanmıştır (Türk Hukuk Kurumu, 1991).

Encyclopædia Britannica Online (2011) intihali, kişinin başka birinin yazdıklarını kendininmiş gibi sunması olarak tanımlayarak, bunun telif haklarını ihlal ettiğini belirtmiştir.

Oxford University Press (2011) ise intihali kişinin bir başkasının işini veya fikrini kendisininmiş gibi geçirmesi olarak tanımlamıştır.

Kanunda intihalin doğrudan verilmiş bir tanımı yoktur. Ancak FSEK m. 34'de *"... hak sahibinin meşru menfaatlerine haklı bir sebep olmadan zarar verir veya eserden normal yararlanma ile çelişir şekilde kullanılamaz."* ifadeleri ile iktibas serbestisinin eser sahibinin haklarına zarar verecek şekilde kullanımının yasak olduğunu belirterek dolaylı da olsa intihale değinmiştir. Kanun maddesinin devamında *"Yayımlanmış musiki, ilim ve edebiyat eserlerinden ve alenileşmiş güzel sanat eserlerinden, iktibaslar yapılmak suretiyle eğitim ve öğretim gayesi dışında seçme ve toplama eserler vücuda getirilmesi ancak eser sahibinin izniyle mümkündür. Bütün bu hallerde eser ve eser sahibinin adı mutat şekilde zikredilmek icap eder."* denilerek iktibasın eğitim ve öğretim gayesi dışında yapılmasının eser sahibinin iznine bağlı olduğu belirtilmiştir. Buna ilaveten yapılan iktibaslarda faydalanılan eserin ve eser sahibinin belirtilmesi kanun tarafından emrolunmuştur. Kanun maddesinde de belirtildiği gibi eser sahibinin rızası dışında ticarî menfaat veya başka gayelerle eserden faydalanmak da yasaklanmıştır. Her ne kadar üretilen eserde faydalanılan eserin ve eser sahibinin adı belirtilmiş olsa da, eser sahibinin izni olmadığından kanun bunu manevî haklara tecavüz olarak değerlendirmiştir. Bu gibi durumlarda eser sahibinin haklarının ve taleplerinin neler olabileceğine tezin önceki bölümünde değinilmiştir.

Kanun ayrıca m. 67'de de intihalden bahsetmektedir:

"... maddelerde sayılan hallerde yanlış veya kifayetsiz kaynak tasrih edilmiş veyahut hiç kaynak gösterilmemişse ikinci fıkra hükmü uygulanır."

Kanun metninde geçen "*yanlış veya kifayetsiz kaynak tasrih edilmiş veyahut hiç kaynak gösterilmemişse*" ifadeleri intihalden bahsetmektedir.

İntihalin genel ve tek bir tanımı olmadığından literatürde çeşitli intihal tanımları yapılmıştır (Tekinalp, 2005).

Messina (1954) intihali, "*Müşterek bir anlayışa göre intihal, başkasına ait olup intihali yapanın kendisininmiş gibi gösterdiği edebiyat veya sanat eserinin temellük edilmesidir*." şeklinde tanımlamıştır.

Tekinalp (2005) intihali, "*sahibinin adına atıfta bulunmayan, eseri kısmen veya tamamen, doğrudan veya dolaylı yansıtan her alıntı*" olarak tanımlamaktadır.

Bayamlıoğlu (2007) "kaynak göstermeksizin bir eserden faydalanmak, alıntı yapmak" şeklinde intihali tanımlamıştır.

Ceritoğlu (2008) intihali "*başkasına ait bir eserin belli bir parçasının bazen de tamamının kaynak göstermeksizin, izin alınmaksızın kendi eserine aktararak esere ve eser sahibinin haklarına yapılan tecavüz*" olarak tarif etmiştir.

Gökyayla (2009) fikir ve sanat eserleri hukukunda en basit anlamda intihalin "*bir eseri ya da eserde yer alan bir ifadenin kişinin kendisine aitmiş gibi göstermesi*" olarak tarif etmiştir.

Anlatılanlar ışığında intihal başkasına ait bir metnin, bir işin tamamının veya bir kısmının olduğu gibi veya üzerinde sahibine ait hususiyet barındırmayacak miktarda değişikliklerle kendisininmiş gibi sunulması anlamına gelmektedir. Hem akademik düzeyde hem de diğer edebiyat ve sanat eserlerinde kasıtlı veya değil her türlü intihal kabul edilemez bir suçtur, hırsızlıkla eşdeğerdir.

İntihal sonucu meydana getirilen ürünlerin hukukî durumu da konumuz açısından önemlidir. Hirsch (1950) "*Bağlılık derecesine göre yeni eser ya bir intihal veya bir işleme yahutta müstakil bir eser olarak vasıflandırılabilir, bir intihal ise her himayeden mahrumdur."* ifadeleri ile intihal ürünü eserlerin himayeden mahrum olacağını belirtmiştir. Aynı paralelde Gökyayla (2009) da intihal sonucu meydana getirilen ürünlerin eser olarak korunmayacağını vurgulamıştır.

Tekinalp (2005) intihalde esas şartın "özgün eserin sahibinin hususiyetinin çalıntı esere aynen geçmesi" olduğunu ifade etmiştir. Yani çalıntı eserde, eser sahibine ait hususiyetin bulunmayışı veya hissedilemez derecede az olması intihalin varlığını doğurur.

İntihalde önemli diğer bir husus da, intihal edilen kısmın büyüklüğüdür. Hangi boyutta, ne kadar olursa olsun başkasına ait bir eserin bir bölümü dahi alıntı yapılmadan veya eser sahibinin adı zikredilmeden başka bir eserde kullanılırsa intihal gerçekleşmiş olur.

Yasaman (2006) da "*uygun görülemeyecek ölçüde*" aktarmanın mevcudiyetinin intihal olacağını söyler. Gökyayla (2009) da aynı şekilde intihalin ölçüsü konusunda kabul görmüş bir kural bulunmadığını, intihalin varlığının kabulü için eserin tamamının kullanılmasının da şart olmadığını söyler. Ölçüsü olmamakla birlikte atıf yapılsa dahi alıntılanan kısmın büyüklüğü iktibas serbestisi sınırını aşarsa intihal gerçekleşebilir (Gökyayla, 2009).

Kanaatimizce de intihal sınırını belirlemek için bir oran belirlemek mantıklı değildir. Böyle bir durum tüm ilim ve sanat eserlerinde iktibas serbestisinin suistimaline yol açabileceği gibi, bu orana dayanarak eserlerinde iktibas yapan ilim adamlarının, sanatçıların ve diğer eser sahiplerinin üreticiliği düşecek, yaratıcılıkları törpülenecektir.

2.4.2. Kaynak Kod İntihali

İntihal sadece akademik çevrede işlenen bir suç değildir. Diğer edebiyat ve güzel sanat dallarında ve sanayinin her alanında intihal vakalarına rastlanmaktadır. Bu intihal türlerinden biri de kaynak kodun intihal edilmesidir.

Kaynak kod intihali, kaynak kod dosyalarının veya kaynak kod kısımlarının asıl eser sahibinin izni olmaksızın kopyalanarak veya değiştirilerek başka bir yazılım üretmek amacıyla kullanılmasıdır. Korsan bilgisayar programı veya korsan kod kullanımından farkı, kodun bir ürün oluşturmak amacıyla kullanılmasıdır. Korsan kaynak kod kullanımında ise kodlar bir program üretme amacı olmaksızın kullanılır. Kaynak kod intihalinde kanaatimizce aşağıda sayılan sebeplerden ötürü atıf yapılmaması şartı aranmamalıdır. Zira atıf iktibas edilen eserlerde söz konusudur. Kaynak kodlar arasında iktibastan değil intihalden söz edilebilir.

Kanun m. 34 ve m. 35'de iktibas edilmesine belli şartlarla izin verilen eserler arasında bilgisayar programlarını ve kaynak kodunu saymamıştır. Bilgisayar programlarının kendine özgü yapısı düşünüldüğünde de zaten böyle olması gerektiği sonucuna

varılabilir. Kaynak kodlar yapısı itibariyle bir ilim ve edebiyat eseri değildir ve bu nedenle kanaatimizce kaynak kodlar arası iktibas yapılması izinli durumlar hariç söz konusu olamaz. Kanunda diğer ilim ve edebiyat eserlerinden iktibas yapılmasına izin verilme amaçları arasında düşünceyi bir adım öteye taşıma, ilgili alana yeni ve farklı ufuklar kazandırma, kültürün ve sanatın ilerlemesini sağlama sayılabilir. Hâlbuki bilgisayar programlarının böyle ulvî bir amacı yoktur.

Kaynak kod intihalinde üzerinde durulması gereken bir diğer önemli nokta kaynak kodun fikrinin, algoritmasının veya programda gerçekleştirdiği işlevin tersine mühendislik yöntemleriyle veya başka herhangi bir şekilde elde edilip kullanımının intihal olup olmadığı konusudur. Kanun her ne kadar fikri değil fikrin ifade ediliş tarzını koruyor olsa da kendine özgü yapısı gereği bilgisayar programlarının tersine mühendislik yöntemleri ile belirlenen amaçlar (örneğin ara işlerliğin sağlanması) dışında fikrinden istifade edilmesini, fikirsel yapısının ortaya çıkarılmasını yasaklamıştır. Bu çerçevede bilgisayar programları üzerinde yapılan tersine mühendislik işlemleri sonrası, kaynak kodun asıl sureti olmasa da temsili konumundaki koddan istifade etmek suretiyle yeni bir bilgisayar programı yazmak, intihal olarak değerlendirilmelidir. Aynı bağlamda Dalyan (2008) bilgisayar programının kaynak kodlarının tersine mühendislik işlemleri ile elde edilerek kanunun izinli saydığı ara işlerliğin sağlanması amacının dışında "*ifade ediliş bakımından esas ölçüde benzer bir başka programın geliştirilmesi, üretilmesi veya pazarlanması ya da program sahibinin haklarını ihlâl eden diğer herhangi bir fiil için kullanılmasının"* FSEK'in m. 38/7 ile açıkça yasaklanmış hukuka aykırı bir fiil olduğunu söylemiştir.

Kaynak kodların intihal edilmesi ve başka bir programda kullanımı asıl eser sahibinin malî ve manevî haklarının ihlâl edilmesine yol açar (Dalyan, 2008). İntihal edilmiş kaynak kodların başka bir programda kullanımı asıl eser sahibinin "eser sahibi olarak anılma hakkının" tecavüzüne de sebebiyet verecektir (Dalyan, 2008). Kaynak kodların aynen veya değiştirilerek kopyalanması ile üretilecek bir bilgisayar programı, asıl eser sahibinin çoğaltma hakkının gaspına yol açacaktır. Aynı şekilde bu çalıntı programın çoğaltılarak yayımlanması da asıl eser sahibinin yayma hakkının ihlaline neden olacaktır. Yayma ve çoğaltma haklarının ihlali ise eser sahibinin eserden kazanmayı

umduğu maddî menfaatlerine zarar vererek malî haklarına tecavüz oluşturacaktır (Dalyan, 2008).

Konuyla alakalı olarak açık kaynak kodlu yazılımların kaynak kodunun kullanımına ayrıca değinmek gerekmektedir. Açık kaynak kod yazılımlarda kodun kullanılmasının serbest olması, o yazılımların istenildiği gibi kullanılacağı anlamına gelmemektedir. Açık kaynak yazılımın lisans şartlarının ihlali intihale veya yazılım telif haklarının çiğnenmesine yol açabilir. Bu durum açık kaynak kodlu yazılımların lisans şartları göz önüne alınarak değerlendirilmelidir. Tablo 2.2'de bazı açık kaynak lisansların yazılımlardan yeni bir yazılım geliştirilmesi durumunda sağladığı olanaklar ve kısıtlamalar gösterilmiştir.

Tablo 2.2: Açık kaynak lisansların sunduğu imkânlar ve kısıtlamalar (Croft, 2003; Todd, 2007; New Media Rights, 2008; Khason, 2009).

	Yazılımlarla lisans yayımlanmak zorundadır	**Ticarî Yazılımlarla birlikte kullanılabilir**	**Değiştirilen kaynak kod yayımlanmak zorundadır**	**Yazılımlar ücretle satılabilir**	**Yazılımların lisans tipi değiştirilebilir**
Apache 2.0	✓	✓	×	✓	✓
GPL v2	✓	×	✓	✓	×
LGPL v2	✓	✓	×	✓	✓
MIT	✓	✓	×	✓	✓
BSD	✓	✓	×	✓	✓
Mozilla Public License (MPL) 1.1	✓	✓	×	✓	✓

* Yayımlanırsa evet

** GPL gibidir. Fakat derlenen kütüphane açık kaynak olmayan yazılımla kullanılabilir

Tablo 2.2'de açık kaynak lisansların sağladığı imkânlara ve kısıtlamalara bakıldığında; lisanslar, kaynak kodun ticarî yazılımlarda kullanılmasını yasaklayabilir, kullanılan kodun yayımlanmasını şart koşabilir, üretilen yazılımın yine açık kaynak bir yazılım olması gerektiğini şart koşabilir, üretilen yazılımın lisansının değiştirilmesine izin

vermeyebilir, üretilen yazılımda kullanılan kaynak kodunun lisansının yayımlanmasını şart koşabilir. Lisans koşullarının ilgili haklarını ihlal etmek açık kaynak kodlu yazılımlarda intihale veya yazılımın telif haklarının çiğnenmesine yol açar.

Kaynak kod intihalinde, intihal edilen kaynak kodun eser niteliğinde olup olmadığı da önemli bir konudur. Bu nedenle intihal olayının söz konusu olması için intihal edilen orijinal ürünün FSEK çerçevesinde eser statüsünde olması gerekmektedir. Eser niteliği taşımayan çalışmalardan yapılan alıntılar intihal sayılmazlar (Gökyayla, 2009). Tezin önceki bölümünde de değinildiği gibi FSEK, kaynak kodu bilgisayar programının korunan kısımları arasında saymıştır. Bu meyanda intihal edilen orijinal kaynak kodların eser niteliği üzerinde durmak gereksizdir. Buradan hareketle eser korumasında olan kaynak kodların (açık kaynak kodların durumuna ayrıca değinilmişti) sahibine atıf yapılmadan, bir kısmının ya da tamamının, olduğu gibi veya değiştirilerek kullanımı intihaldir.

Kaynak kod intihalinin diğer intihal türlerinden farkı; edebiyat, sanat, musiki ve diğer güzel sanat dallarında intihal yapılan eserleri fark etmek, kaynak kod intihali yapılan bilgisayar programını fark etmekten daha kolaydır. Çünkü diğer sanat dallarında üretilmiş eserlerde, o konunun uzmanları veya meraklıları, söz konusu eserde intihal olduğunu herhangi bir teknik incelemeye gerek kalmadan fark edebilirler. Fakat bilgisayar programlarının kaynak kodları görünmediğinden teknik bir inceleme söz konusu olmaksızın intihali tespit etmek zordur.

İntihal sonucu meydana getirilmiş ürünler hırsızlık mamulü ürünlerdir. Kaynak kodların intihali ile üretilen programlar da aynı bağlamda çalıntı programlardır. Parker ve Hamblen (1989) çalıntı programı, başka bir program üzerinde az sayıda rutin dönüşümler yapılarak üretilen program olarak tanımlamıştır. Yazar rutin değişiklikleri programın detaylı olarak anlaşılmasını gerektirmeyen basit metin değişiklikleri olarak tarif etmiştir (Parker ve Hamblen, 1989). Çalıntı programa getirilen bir başka tanım da, orijinal bir programın tam bir kopyası yada kaynak kodda çeşitli metinsel dönüşümler yapılmış bir varyantıdır (Jones, 2001).

Daha önceki araştırmalar tipik yazılım sistemlerinin kaynak kodlarının %7 - %23 oranında klon kod içerdiğini göstermiştir (Roy ve diğ., 2009). Tablo 2.3'te bazı yazılımlardaki klon kod yüzdeleri gösterilmiştir.

Tablo 2.3: Bazı yazılımlarda klon kod bulunma yüzdesi (TAIRAS, 2010).

Program Adı	**Satır Sayısı (bin)**	**Klonlama Oranı (%)**
Linux kernel	4.365	15
Java Development Kit (JDK) 1.4.2	2.418	8
JDK 1.3.0	570	9
Process-Control System	400	12
JHotDraw 7.0.7	71	19
JavaGenes 0.7.68	45	10

Kaynak kodlar arasında intihal olduğunun tespiti için birbirine benzeyen kaynak kodların karşılaştırılması gerekmektedir. Unutulmamalıdır ki birbirine benzeyen her kod intihal edilmiş kod değildir. Cosma (2008) benzerlik ve intihalin birbirine karıştırılmaması gerektiğini söyler. Yazara göre, kaynak kod dosyaları arasında benzerlik olsa bile, intihal ancak benzerliğin dikkatlice incelenmesi sonucunda saptanabilir. Benzerliğinden şüphelenilen kaynak kod dosyaları aynı karakteri paylaşırlar. İntihalden şüphelenilen durumlarda kaynak kod dosyaları arasında belirgin bir şekilde programın mantığı, yaklaşımı ve işlevselliği benzeşir ve bu tür bir benzerlik intihalin olduğuna yönelik güçlü bir kanıt teşkil eder (Cosma, 2008).

Kaynak kod intihalinin gerçekleşmesine yol açan birçok neden olmakla birlikte en bilinen yöntem kod klonlama olarak da adlandırılan kaynak kodun dosyalar arasında kopyalanmasıdır.

2.4.2.1. Kod Klonlama

Başka bir kodun bir kısmının veya tamamının kopyalanması yoluyla meydana gelen koda klon kod (*clone code*) veya klon denir.

Klon kodun tek veya genel bir tanımı olmamakla birlikte, bir kod bölümünün başka bir kod bölümüyle eşdeğer (aynı) veya benzer olması şeklinde tarif edilir (Higo, 2006).

Klon kodlar birbirlerinin kopyası olan iki veya daha fazla kod bölümlerini temsil eder (TAIRAS, 2010).

Yazılım mühendisliğinde kaynak kod benzerliği de klon kod olarak bilinir (Jiang, 2009). İki kaynak kod birbirine benzeşiyorsa bir klon çifti oluştururlar (Koschke ve diğ., 2006).

Bir kod klon çifti (*code clone pair*) birbirine yapısal veya sözdizimsel olarak benzeşen bir çift kaynak kod dosyasını veya kaynak kod bölümünü ifade eder. Klonlama, kaynak kodun bazen olduğu gibi bazen de küçük değişiklikler yapılarak bir programdan diğerine kopyalanması suretiyle gerçekleşir. Böylece birbirinin eşdeğeri kaynak kodlar yaratılmış olur (Kapser ve Godfrey, 2003).

Weissgerber ve Diehl (2006) kaynak kodun benzerlik durumunu üç kısma ayırmıştır. Bunlar her iki kaynak kodun eşit olduğu, kaynak kodların özdeş olduğu ve benzerliğin olmadığı durumlardır.

Bilgisayar biliminin pek çok alanında, akademide ve endüstride ilgili alanların gerektirdiği amaçla yazılım kaynak kodlarını karşılaştırmak gerekmektedir.

Yazılım endüstrisinde bilgisayar programcıları, fikrî mülkiyet avukatları, proje yöneticileri ve açık kaynak taraftarlarının benzer kaynak kodu tanımlamasında ve amaçlarında farklı bakış açıları vardır (Zeidman, 2008).

Yazılım mühendisliğinde kaynak kod karşılaştırma yapılmasının esas amacı kopyalanmış kod kısımlarını belirlemek ve onları tek bir fonksiyon altında birleştirmektir (Jafar, 2007; Lozano ve diğ., 2007). Bununla birlikte programı anlama, kaynak kod kalite analizi, yazılım gelişimi analizi, kod sıkıştırma, virüs algılama ve hata algılama gibi alanlarda da kaynak kod karşılaştırılması yapılmaktadır (Roy ve diğ., 2009).

Akademide ise öğrenci ödevlerinde kopya/intihal tespiti, araştırmacıların makalelerinde intihal tespiti, öğrencilere verilen programlama ödevlerinin kaynak kodlarında benzerlik

araştırması gibi alanlarda benzerlik tespiti yapılmasına ihtiyaç duyulmaktadır (Cosma ve Joy, 2006).

Hukuksal boyutta ise telif hakkı ihlalinin tespiti, ticarî sır hırsızlığının ve patent ihlalinin önlenmesi amacıyla kaynak kodların karşılaştırmasına ihtiyaç duyulmaktadır (Zeidman, 2008).

Açık kaynak kod savunucuları kaynak kodun yazılımla birlikte verilmesini ve yazılım lisansının da buna uygun olması gerektiğini savunurlar. Bir konuda geliştirilmiş olan açık kaynak kod başka bir projede kullanılabilir veya uyarlanabilir. Böylece farklı açık kaynak kod projelerinde kaynak kod bölümlerinin aynı olması veya benzerliği gibi durumlara sık rastlanılır. Fakat bu durum açık kaynak kod lisans şartları yerine getirildiği müddetçe sorun olmaz.

Tüm sanat dallarında, özellikle edebi ve akademik ürünlerde intihal olaylarına rastlanmaktadır. Fakat bilgisayar programının kaynak kodu diğer metin türlerinden farklıdır. Kaynak kod edebi metinlere göre daha kurallı bir yapıya sahiptir ve kaynak kod intihalini tespit etmek daha fazla dikkat gerektirir (Konecki ve diğ., 2009). Sadece sözdizimsel intihalleri tespit etmek nispeten daha kolayken, yapısal değişiklik yapılmış intihallerin tespiti daha zordur. Kod sözdizimsel olarak farklı ifade edilip intihal gizlenmeye çalışılsa da iki programın işlevselliği aynıdır.

2.4.2.2. Kaynak Kod Klonlama Sebepleri

Yazılım geliştiriciler açısından kodu klonlama, kodun tekrar kullanılabilirliğini sağladığı için kimi zaman gereklidir. Bazen de yazılım geliştiriciler zaman baskısı veya işin kolayına kaçmak sebepleri ile kodu klonlarlar.

Kodun klonlanmasına yol açan sebeplerden birkaçına aşağıda değinilmiştir (Kim ve diğ., 2004; Higo, 2006; Jiang, 2006; Cosma ve Joy, 2008) :

- **Kodun kopyalanması**

Kaynak kodun klonlanma sebeplerinin en başında kodun kopyala-yapıştır yapılması gelmektedir. Geliştiriciye zaman kazandırması bunun yapılmasındaki ana sebeplerden

biridir. Aynı işleve sahip bir kodu tekrar yazmaktansa, kodun kopyalanması daha pratik bir yöntem olarak uygulanmaktadır.

- **Yazılımın tekrar kullanımı**

Yazılım geliştirme sürecinde daha önce çözülen bir problemle tekrar karşılaşıldığında, o sorunu baştan tekrar çözmek yerine sorunun çözümünde kullanılan kodu tekrar kullanmak normal bir durumdur. Kodu tekrar kullanmanın çeşitli tipleri olmakla birlikte en bilinen yöntem kodun kopyalanarak yeni yazılıma uygulanmasıdır. Programcılar hız ve kolaylık sağladığı için kullanacakları kod bölümlerini yeni yazılıma kopyalamayı ve gereken değişiklikleri yapmayı tercih etmektedirler. *Kopyala–yapıştır programlama* da denilen bu durum kodun tekrarlanmasına yol açar (Koni-N'sapu, 2001).

- **Programlama dilinin sınırlamaları**

Bazı programlama dilleri soyutlama (abstraction), sınıf (class) ve miras (inheritance) yapısı gibi programlama yapılarını barındırmadığından, bir fonksiyonun birden fazla kaynak kod dosyasına kopyalanması kaçınılmaz olabilir. Soyut veri tiplerinin veya yerel değişkenlerin kullanılamaması durumunda yazılımın farklı bölümlerinde aynı işlevi gören kod kısımları tekrarlanarak yazılmak zorundadır. Aynı işlevi gören veya aynı mantığa yönelik kod ihtiyacı kodun klonlanmasına yol açabilir.

- **Şablon kodların kullanımı**

Genel bir şablonu mevcut olan konularda kodu tekrar yazmaktansa o şablona birkaç fonksiyon ekleyerek ve biraz değişiklik yaparak istenilen iş gerçekleştirilebilir. Fakat bu birbirine yakın işlevlere sahip kod bloklarının tekrarlanmasına yol açar. Birbirine çok benzeyen alanlarda bu durum sıklıkla gerçekleşir. Yeni bir modül yazmak gerektiğinde, güvenilir ve test edilmiş olduğu için eski kodun biraz değiştirilerek kullanılması yaygın bir uygulamadır.

- **Kod üretici araçların kullanımı**

Kod üreten yazılım veya araçların kullanımı ile aynı işi gören ve aynı işleve sahip yazılımlar üretilebilir. İki kod arasında sadece tanıtıcı isimleri değişkenlik gösterir, geri kalan kısımlar genelde tamamen aynıdır.

- **Kodun farklı ekiplerce yazılması**

Kaynak kodun farklı takımdaki geliştiriciler tarafından yazılması birbirine benzer fonksiyonların kodda bulunmasına yol açabilir.

- **Lisans ihlalleri**

Kaynak kodun lisansının bilerek veya bilmeyerek ihlal edilmesi de kodun klonlanmasına sebep olur. Özellikle açık kaynak kod yazılımlardan kopyalama yapılması bu durumun en bilinen örneğidir. Genelde açık kaynak lisansının kişiye her türlü imkânı sağladığı düşünülür. Bu da açık kaynak kodun istenildiği gibi kullanılmasına ve farkında olmadan intihal yapılmasına yol açar.

- **Kodların tesadüf eseri benzeşmesi**

Nasıl ki bir duyguyu yada bir düşünceyi her insan farkı biçimlerde ve üslupla ifade ediyorsa aynı şekilde aynı işleve sahip bir programı yazmak da geliştiriciden geliştiriciye farklılık gösterir. Fakat nadiren de olsa aynı işleve sahip kodların farklı geliştiriciler tarafından aynı şekilde yazıldığı durumlar da olabilir. Yalnız bu durumla sık karşılaşılmaz, bu sadece istisna bir durumdur.

2.4.2.3. Klonlama Yöntemleri

Kaynak kod intihalinin diğer intihal türlerinden farklı karakteristikleri vardır. Kaynak kod intihali yapmak kolaydır fakat tespit etmek zordur. Kaynak kod üzerinde yapılan değişiklikler sözcüksel ve anlamsal (yapısal) değişiklikler olarak iki kısma ayrılabilir (Kustanto ve Liem, 2009).

Sözdizimsel kaynak kod intihali, genellikle kodun kopyalanarak diğer koda yapıştırılması ile oluşur (Roy ve Cordy, 2009). Kopyalanmış kod ya olduğu gibi bırakılır veya klonlanan kodda yorumları silmek, kod biçimini değiştirmek, değişken adlarını, tipini, fonksiyon ve prosedür isimlerini değiştirmek, önemsiz kod ekleme çıkarma işlemleri gibi basit işlemler yapılarak gerçekleşir. Bu intihal işlemleri programlama bilgisine ihtiyaç duyulmadan bir metin editör ile yapılabilir (Kustanto ve Liem, 2009).

Yapısal intihal türünde iki kod bölümü sözdizimsel açıdan farklıdır fakat işlevsel açıdan özdeştir veya benzerlik gösterir. Bu türe anlamsal (semantik) klon da denilir (Roy ve

Cordy, 2009). Bu tür intihalde bir fonksiyon veya kod bloğu farklı komutlar ve programlama yapıları ile aynı işlevi görecek şekilde tekrar yazılır. Bu nedenle bu intihal türünün yapılması programlama bilgisi gerektirir. Yapısal anlamdaki intihal, yapısal programlama ifadelerini (kontrol yapıları, döngü yapıları, fonksiyon ve sınıf yapıları) değiştirmek veya eklemek/çıkarmak, fonksiyon/prosedür çağrılarını değiştirmek, programın işleyişini etkilemeyecek değişiklikler yapmak şeklinde örneklendirilebilir (Kustanto ve Liem, 2009).

Sözdizimsel değişiklikleri tespit etmek kolaydır. Öte yandan yapısal değişiklikleri içeren intihal türünü tespit etmek ise daha zordur.

Kaynak kodda yorumların veya değişken isimlerinin değiştirilmesi gibi sözdizimsel değişiklikler programın ayrıştırılmasını (*parsing*) etkilemez fakat kaynak kodda blokların yerinin değiştirilmesi gibi yapısal değişiklikler programın ayrıştırılmasını ve hata düzeltmesini (*debugging*) etkiler (Cosma, 2008).

Çok kullanılan ve bilinen kaynak kod intihal kalıpları ve intihali gizleme teknikleri aşağıda listelenmiştir (Jones, 2001; Weissgerber ve Diehl, 2006; Mozgovoy, 2007):

- Kaynak kodu olduğu gibi kelimesi kelimesine (tam) kopyalamak (*verbatim copying*)
- Yorumları değiştirmek
- Boşluk, tab gibi beyaz boşluk (*whitespace*) olarak bilinen kodda görünmeyen kısımları (girintileri) değiştirmek, kodlama stilini/biçimlemesini değiştirmek
- Tanıtıcıların adını değiştirmek
- Kod bloklarının yerini/sıralamasını değiştirmek
- Kod bloğu içindeki komut setlerinin/deyimlerin yerini/sıralamasını değiştirmek
- İfadelerde yer alan işlenenlerin ve işleyicilerin (operands/operators) sırasını değiştirmek
- Veri tiplerini değiştirmek
- Fazladan deyim, değişken, fonksiyon eklemek
- Kontrol ve döngü gibi temel programlama yapılarını eş değeriyle değiştirmek (*for* döngüsünü *while* döngüsüne dönüştürmek gibi)
- Fonksiyonun yapısını ve çağrılma şeklini değiştirmek

- Fonksiyonun veya değişkenin görünürlüğünü değiştirmek (her yerden erişilebilir, sınıf dışından erişilebilir, sadece sınıf içinden erişilebilir şeklindeki görünürlük özelliğini değiştirmek)

Buradaki değiştirmelerden kasıt düzenlemek, değişiklik yapmak, farklı şekilde ifade etmek, başka bir dile dönüştürmek, ekleme yapmak, silmek, sıralamasını ve yerini değiştirmek gibi işlemlerdir.

Yukarıda değinilen bu yöntemlerden en bilinenleri ve en çok kullanılanları kodda yer alan değişken ve fonksiyonları yeniden isimlendirmek, kod bloklarını sıralamak, kaynak koda düzmece kod eklemek, kopyalanan koddaki gereksiz kısımları silmek, kaynak kodu farklı komutlarla ifade etmek, kaynak kodu başka bir programlama diline çevirmek ve birden fazla programdan kaynak kod kopyalamak olarak sayılabilir. Bu yöntemler aşağıda kısaca açıklanmıştır (Freire ve diğ., 2007):

- **İsim değişikliği**

Kodda yer alan tanıtıcı isimlerinin (değişken, fonksiyon, modül, sınıf vb.) değiştirilmesi, yorumların değiştirilmesi gibi kaynak kod bileşenlerinin ismini değiştirmek olarak örneklenebilir.

- **Kod bloklarını tekrar sıralama**

Kod bloklarının, deyimlerin, fonksiyonların veya deyimlerin içinde yer alan operatörlerin ve operatörlerin etkilediği kısımların yerinin değiştirilmesidir. Bu işlem aynı kod dosyası içinde olacağı gibi, fonksiyon, modül gibi kısımların başka bir kaynak kod dosyasına alınması şeklinde de yapılabilir.

- **Kodu farklı biçimde ifade etme**

Kaynak kodda yer alan deyim, fonksiyon vb. yapıları programlama dilinin farklı komutları ve yapılarıyla ifade edilmesidir. Örnek olarak bir değişkenin değerini bir artıran x++ yerine x=x+1 değişikliği yapmak veya *if-else if* koşul yapısı yerine *switch-case* koşul yapısını kullanmak verilebilir. Bu tarz değişiklikler yapısal kaynak kod klonlaması olarak da ifade edilir. Anlam ve işlev olarak aynı fakat yapısal olarak farklı şekilde ifade edilen çalıntı kodu orijinal koddan ayırt etmek zordur.

- **Kaynak kodu farklı bir programlama diline çevirme**

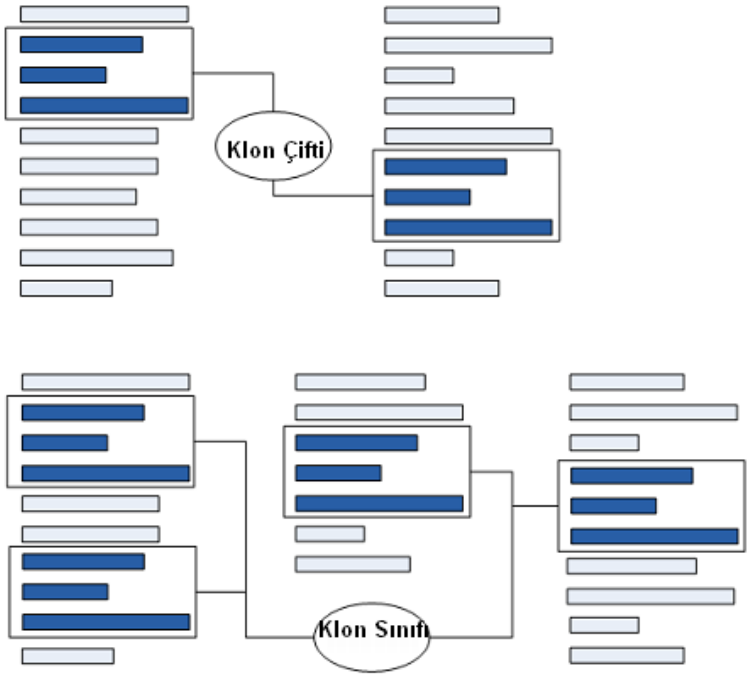
Kopyalanan kaynak kodun intihalini gizlemek amacıyla veya üretilecek olan yazılımın orijinal yazılımla farklı platformlarda olması sebebiyle, mevcut orijinal kaynak kod başka bir programlama diline çevrilir. Bu çevrim, bir programlama dilinden başka bir programlama diline yapılacağı için haliyle kaynak kodlar aynı olmayacaktır. Fakat kaynak kod, öteki programlama dilinde aynı işlevi gören komutlarla yazılır. Her iki kodun fikri, işlevi ve amacı aynıdır.

- **Kaynak koda düzmece kod ekleme/silme**

Kaynak koda programın ana amacıyla alakasız eklemeler yapmak veya hata kontrolü yapan kısımları gereksiz görüp silmek gibi işlemlerdir.

- **Karma kaynak kod kullanma**

Kaynak kodun hem çalıntı kod bölümlerini içermesi hem de çalıntı olmayan bağımsız olarak yazılmış kaynak kodları içermesi durumudur. Bir diğer şekli ise çalıntı kaynak kodun sadece bir programa ait olmaması, birden fazla programa ait kodların birleştirilerek yeni koda aktarılması durumudur. Bu genelde çoklu intihal (*multiple plagiarism*) olarak bilinir. Şekil 2.11'de iki kaynak kod dosyası arasında klonlama (klon çifti) ve birden fazla dosyadaki kaynak kodların bir programa kopyalanması temsil edilmiştir.



Şekil 2.11: Klon çifti ve klon sınıfı (Jia, 2007)'den uyarlanmıştır.

Kaynak kodlar/kod kısımları arasında aşağıdaki unsurların varlığı kaynak kodların benzerliğine veya intihaline yönelik delil olarak değerlendirilebilir (Burrows ve diğ., 2007; Cosma, 2008):

- Kaynak kod biçimlendirmesinin/sitilinin (girinti kullanımının) aynı/özdeş olması
- Sabit veya değişken isimlerinin aynı veya özdeş olması veya birbirini anımsatması
- Değişken, sabit, fonksiyon, class gibi tanıtıcıların adını değiştirmek fakat yapının ve mantığın özdeş olması
- Değişken tiplerinin özdeşliği (örneğin orijinal kodda long int sayı tipinde tanımlanan değişkenin intihal edilen kodda signed long int sayı tipinde tanımlanması)
- Program mantığının özdeş olması
- Kaynak kodların yapısal, sözdizimsel ve anlamsal açıdan özdeş olması
- Benzer kaynak kod yorumları
- Satır sayılarının aynı/veya aşağı yukarı aynı olması ve karşılıklı satırların aynı işlevselliğe sahip olması
- Sınıf değişkenlerinin benzerliği

- İki kaynak kodda benzer hataların olması
- Değişik kod satırları arasındaki benzerlik
- Sözcüksel, sözdizimsel benzerlikler
- Fonksiyon prototiplerinin olması
- Sihirli sayıların (magic numbers) özdeş olması
- Program çıktı formatlarının özdeş olması (print/printf/echo gibi ekrana biçimlendirilmiş bir şekilde çıktı üreten fonksiyon çağrılarının özdeş olması)
- Ender kullanılan programlama yapılarının tutarlı olması
- Kontrol ve döngü yapılarının türdeşleri ile değiştirilmesi
- Kaynak kod dosya isimlerinin aynı, özdeş veya birbirini anımsatıcı olması
- Kaynak kodun yer aldığı klasör yapılandırmalarının ve isimlerinin özdeş olması
- Aynı kaynak kod dosyalarında aynı işlevi gören kısımların yer alması
- Kaynak kod dosya boyutlarının aynı olması

Kaynak kod intihali küçük kod bölümlerini kopyalamaktan, çok büyük kaynak kod yığınlarını kopyalamaya kadar çeşitlilik gösterebilir (Zeidman, 2011). Klonlanan kod tek bir dosya ile sınırlı olabileceği gibi birden fazla dosyaya da kopyalama yapılabilir. Kapser ve Godfrey (2003) olası durumları şöyle saymıştır:

Kaynak kodların klonlanması; fonksiyonda yer alan kod bloklarının kopyalanması, aynı kaynak kod dosyasında hemen hemen aynı işleve sahip benzer fonksiyonların olması, fonksiyonların aynı veya farklı klasörde yer alan dosyalara kopyalanması, kaynak kod dosyasının olduğu gibi veya değiştirilerek aynı veya farklı klasörlere kopyalanması, program başlangıç ve sonlandırma dosyalarının veya fonksiyonlarının kopyalanması şeklinde gerçekleşebilir.

2.4.2.4. Klonlamanın Sağladığı Avantajlar

Klonlamanın ana avantajı yazılımın yeniden kullanımını (tekrar kullanımı da denir) (*software reusing*) basitleştirmesidir (Davey ve diğ., 1995). Yazılımın yeniden kullanımı yazılım geliştirmede sık başvurulan bir teknik ve aynı zamanda yazılım mühendisliğinde bir çalışma alanıdır.

Kapser ve Godfrey (2006) klon kodların yazılımlar üzerinde bazı yararlı etkileri olabileceğini söyleyip bunlardan birkaçını şöyle sıralamıştır:

- **Kod klonlama yazılımda tekrar kullanılabilirlik sağlar.**

Bilgisayar programcıları bir yazılımı tasarlarken ve programlarken daha önce yazılan sınıflardan, fonksiyonlardan, modüllerden, arayüz ve tasarımdan istifade ederler (Koni-N'sapu, 2001). Yazılımın tekrar kullanımı daha çok programlama evresinde gerçekleşir. Özellikle açık kaynak kod projelerinde kodun tekrar kullanımı geliştiriciyi aynı işlevli kod yazma zahmetinden kurtarır.

- **Klon kod, yazılımdaki riskleri minimize etmek amacıyla kullanılabilir.**

Kod klonlama riski minimize etmek için de yapılabilir (Jiang, 2006). Hatasız çalıştığına emin olunan kaynak kodlar, aynı işlevli başka bir yazılıma iyi adapte edilirse yeni yazılım üretmenin getirdiği riskler de bertaraf edilmiş olur. Bu, yazılım geliştirme süresini azaltır.

- **Kod klonlama yazılımın platform desteğini artırmak amacıyla yapılabilir.**

Yazılımın başka bir platformda da çalışması için kodun farklı bir programlama diline dönüşümü yapılabilir. Açık kaynak projelerinin platform çeşitliliğinin sağlanabilmesi için bunun çok büyük önemi vardır. Diğer ticarî projelerde bu intihal kapsamında değerlendirilse de kodun klonlanması bu alanda çok faydalıdır. Bu durum sadece yazılımlarda değil donanım eksenli kullanımda da çok kullanışlıdır. Örneğin mevcut bir donanım ailesine yeni bir donanım birimi eklendiğinde, o birim için yeniden sürücü yazmak yerine, mevcut sürücüde yeni donanım için ufak birkaç değişiklik yapmak yeterli olacaktır. Ayrıca aynı aileye mensup diğer donanımlarda test edilmiş olan sürücüyü tekrar test etmeye gerek kalmayacak ve bu zaman kaybının da önüne geçecektir.

- **Kod klonlama tecrübe edinmek amacıyla yapılabilir.**

Geliştiriciler yeni özellik uygulamak istediklerinde daha önce yaptıkları koddan kopyalama eğilimindedirler. Bu sayede problemi daha iyi kavramakta ve çözümü düşünmektedirler (Jiang, 2006).

- **Kod klonlama yazılımın performansının korunmasını sağlayabilir.**

Daha önce kullanılmış ve optimize edilmiş kod, aynı amaca yönelik yazılacak başka bir yazılımda performans kaygısı güdülerek kullanılabilir. Kodun daha önce kullanılmış

olması ve her iki yazılımın işlevlerinin de aynı olması kodun iyi adapte edilmesi halinde önceki yazılımın performansının yeni yazılımda da kullanılmasını sağlar.

2.4.2.5. Klonlanmış Yazılım Sistemlerindeki Problemler

Yazılım klonlama, sunduğu avantajların yanı sıra bazı problemleri de beraberinde getirmektedir. Yazılım mühendisliği süreçleri açısından incelenirse iyi analiz ve test edilmemiş kod klonlamanın yazılım üzerindeki ana etkileri yazılım kalitesinin düşmesi, yazılımın bakımını zorlaşması, yazılımın maliyetini artırması ve yazılım üretim süresinin uzaması olarak sayılabilir. Klon kod kullanımının yazılımda oluşturduğu diğer bazı olumsuz etkilere aşağıda başlıklar halinde değinilmiştir (Davey ve diğ., 1995; Kapser ve Godfrey, 2003; Mishne, 2003; Rieger, 2005; Kapser ve Godfrey, 2006; Tekin ve diğ., 2009):

- **Kod klonlama, kodun karmaşıklığını artırır.**

Kaynak kodun çeşitli yerlerinde klonlanmış kodun yer alması kodu daha karmaşık bir hale soktuğu gibi kodun okunabilirliğini ve anlaşılırlığını azaltacaktır.

- **Klonlanan kod orijinal yazılımdaki hataları içerir.**

Kodun klonlanmasıyla, orijinal koddaki yazılım hataları kopyalanan yazılıma da geçmiş olur.

- **Kodun klonlanması ile hata tekrarı artar.**

Yazılımın birçok yerine klon kodun kopyalanması, klon kodda bulunan hataların yazılımda tekrarlanacağı anlamına gelir.

- **Klon kodun uyarlanması ile orijinal kodda bulunmayan hatalar oluşabilir.**

Performansı korumak ve yazılımdaki riskleri azaltmak amacıyla klon kodun kopyalanarak yeni yazılıma uyarlanması, önceki yazılımda bulunmayan hataların oluşmasına sebep olabilir. Kopyalanan kodun sorunsuz çalıştığının bilinmesi kodun yeni yazılıma uyarlandıktan sonra sorunsuz çalışacağı anlamına gelmez.

- **Klon kod, kaynak kodun ve programın boyutunu artırır.**

  Kaynak kod kopyalaması iyi analiz edilmediği takdirde programda gereksiz ve ölü kodlar yer alacak bu da kaynak kodun ve dolayısıyla çalıştırılabilir bilgisayar programının boyutunun şişmesine sebep olacaktır.

- **Klon kod yazılımın bakımını zorlaştırır, yazılımın maliyetini artırır.**

Kaynak koddaki hataların tespit edilmesi ve düzeltilmesi, klon kodun adaptasyonunda yaşanan zorluklar, kodun karmaşıklığı gibi sebepler doğrudan yazılımın bakımını da zorlaştırmaktadır. Kodun bakımı, yazılım üretim sürecinde harcanan emeğin büyük bir bölümünü ve yazılımın toplam maliyetinin ise yaklaşık %90'ını oluşturduğu düşünüldüğünde, kodun kopyalanması yazılımın maliyetine olumsuz yönde etki edecektir (Jiang, 2009; TAIRAS, 2010).

- **Klon kod, sistem kaynaklarının gereksiz kullanımına sebep olur.**

Kodun kopyalanması ile çok sayıda gereksiz ifadeler de kopyalanmış olur. Bunlara örnek olarak koddaki gereksiz döngüler, hiç kullanılmayan değişkenlerin bellekte yaratılması verilebilir. Bu ise sistemin işlemci, bellek gibi donanımının boş yere çalışmasına yani sistem kaynaklarının israfına neden olur.

- **Klon kod programın derlenme süresini artırır, programın yavaş çalışmasına neden olur.**

Klon kodun programın çeşitli yerlerinde gereksiz tekrarı veya klon kodda kopyalanan gereksiz kısımlar programın derlenme süresinin uzamasına ve derlenen bilgisayar programının yavaş çalışmasına sebep olur.

- **Klon kod yazılımlarda adaptasyon işlemlerini zorlaştırır, yazılımı üretim sürecini olumsuz etkiler.**

Klon kod ile üretilen yazılıma yeni özellik eklenmek istendiğinde kopyalanan kodun iyi anlaşılamaması ve iyi analiz edilememesi adaptasyon işlerini zorlaştıracaktır. Bu aynı zamanda yazılım üretim süresinin uzamasına sebep olur.

- **Klon kod orijinal koda bağımlı olabilir.**

Orijinal kod, üzerinde çalışacağı platformun özellikleri dikkate alınarak yazılmış olabilir. Bu kodda donanıma veya platforma bağlı olarak değişiklik yapıldığında, klonlanan kodda da bu değişikliği yapmak gerekecektir. Bunu düzeltmek uzun zaman alabileceği gibi, dokümantasyon eksikliği veya platform gereksinimlerinin karşılanamaması nedeniyle imkânsız da olabilir. Bu anlamda klonlanan kod orijinal koda bağlıdır. Bir diğer sorun da orijinal kodun bu bağımlılıklarının bilinmemesi veya fark edilmemesi gibi durumlarda kopyalanan kod hata vermeye meyilli olacaktır.

2.4.2.6. Klonlama Tipleri

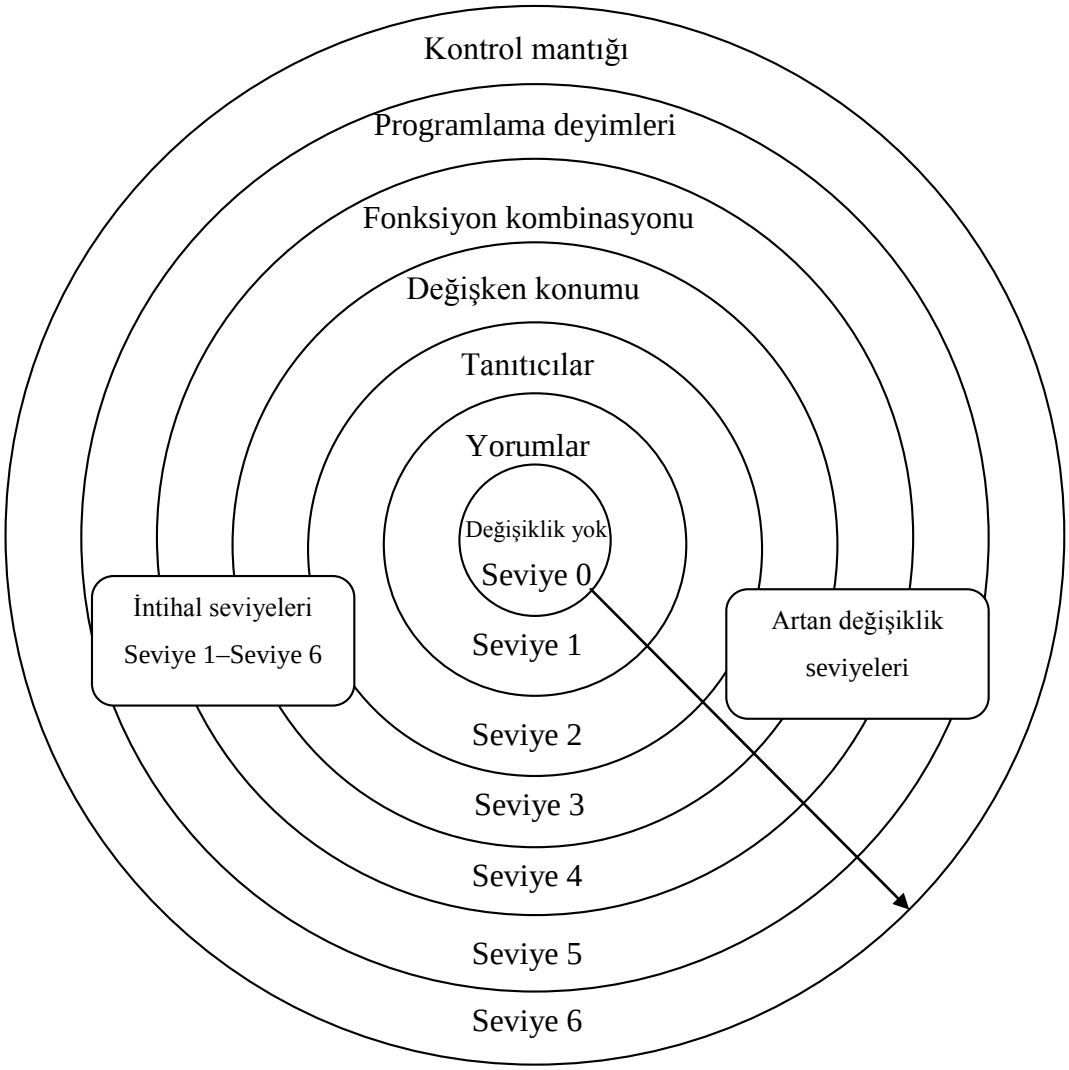
Kodun klonlanma türünün metinsel ve yapısal klonlama olarak kavramsal anlamda ikiye ayrıldığından daha önce bahsedilmişti. Literatürde kaynak kodun klonlanması teknik olarak dört tipte incelenmektedir. Bunlardan ilk üçü metinsel benzerlik içeren kod klonlamalarıdır. Dördüncü tip klonlama ise metinsel olarak farklı fakat aynı işleve sahip fonksiyonlar barındıran ve yapısal benzerlik içeren klon tipidir. Bahsedilen klon tipleri aşağıda açıklanmıştır (Davey ve diğ., 1995; Roy ve Cordy, 2009; Tekin ve diğ., 2009):

- Tip I: Boşluk, hizalama, kod biçimlendirmesi (kodlama sitili/hizalama) ve yorumları değiştirme haricinde tamamen aynı kod.
- Tip II: Tanıtıcı, literaller, değişken tipleri, boşluk, kod biçimlendirmesi ve yorumlarda farklılık haricinde sözdizimsel olarak tamamen aynı kod.
- Tip III: Kopyalanmış kodun üzerinde değişiklik yapılması (değiştirme, ekleme, silme gibi) ve tanıtıcı, literaller, değişken tipleri, boşluk, kod biçimlendirmesi ve yorumlarda ekleme değiştirme, ek fonksiyon ekleme veya küçük değişiklikler yapma.
- Tip IV: İki veya daha fazla kod bölümlerini aynı işi gerçekleştirecek şekilde başka bir yolla ifade etme. Bu tür klonlar belirlenmesi en zor olanlarıdır.

Tip I'den Tip IV'e kodda yapılan değişim artmakta ve değişim biçimi farklılık göstermektedir (Roy ve Cordy, 2007). Aynı paralelde klonları algılamak için geliştirilen programların karmaşıklık seviyeleri de Tip I'den Tip IV'e artmaktadır (Davey ve diğ., 1995).

Literatürde klon kod tipleri dışında kaynak kod intihalinin çeşitliliği üzerine de çalışmalar yapılmıştır. Faidhi ve Robinson (1987) kaynak kod intihalinin çeşitliliğini 6 seviyeye ayırmıştır. Bu seviyeler hiçbir değişikliğin yapılmadığı orijinal kaynak kodun kendisi, yorumlarda değişiklik, tanıtıcılarda değişiklik, değişkenlerde ve yerlerinde yapılan değişiklik, fonksiyon/işlev kombinasyonları/birleştirmeleri, program yapılarının değişimi ve kontrol mantığında yapılan değişiklikler olup açıklamaları aşağıda verilmeye çalışılmıştır (Parker ve Hamblen, 1989; Ji ve diğ., 2007; Sraka ve Kaucic, 2009; Zeidman, 2011):

Seviye 0: En alt seviyedir, orijinal programı temsil eder. Hiçbir değişiklik içermez.
Seviye 1: Yorumlar ve kaynak kod biçimlendirmesi (girintiler) değiştirilir. Bu kısımda yapılan değişiklikler pek fazla programlama bilgisi gerektirmez.
Seviye 2: Tanıtıcılar yeniden isimlendirilir.
Seviye 3: Değişkenler, sabitler ve yerleri değiştirilir, ekstra değişkenler tanımlanır.
Seviye 4: Program modülleri değiştirilir, ek modüller yazılır, iki modül tek bir modülde birleştirilir.
Seviye 5: Programlama ifadeleri birbirine çevrilir, eşleniğiyle yazılır, yapıları değiştirilir. Örneğin *if-else if* yapısı *switch-case* yapısına çevrilir.
Seviye 6: Karar ve kontrol mantığı değiştirilir. Bu seviye iyi bir programlama bilgisi ve hüneri ister.



Şekil 2.12: Program intihal çeşitliliği (Faidhi ve Robinson, (1987), Parker ve Hamblen (1989)'den uyarlanmıştır)

Şekil 2.12'de görüldüğü üzere seviyeler arttıkça programda yapılan değişiklikler de artmakta ve programı ilk haline göre tanımak zorlaşmaktadır. Her seviye kendinden bir önceki seviyeyi de kapsamaktadır. Seviye 4'ten sonra (Seviye 5-6) programın kaynak kod yapısında değişiklik yapıldığından intihal edilmiş programı (çalıntı kaynak kodu) tespit etmek zorlaşacaktır (Ji ve diğ., 2007).

Tablo 2.4'te kaynak kod intihalini gizlemek için kullanılan yöntemlerin bazıları listelenmiştir.

Tablo 2.4: Kaynak kod intihalini gizleme yöntemleri (Mahmoud, 2008)'den uyarlanmıştır.

Yöntem	**Orijinal Kod**	**Maskelenmiş Kod**

Kontrol ve döngü yapılarını eşleniğiyle değiştirmek	for(i=0;i<n;i++)
	{
	…
	}
		i=0;
		while(i<n)
		{
		…
		i++;
		}

	if(x == 'a')
	{ … }
	else if (x == 'b')
	{ … }
	else
	{ … }
		switch(x){
		case 'a':
		…
		break;
		case 'b':
		…
		break;
		default:
		…
		}

Koşulu uyarlamak ve koşul bloklarının sıralamasını değiştirmek	if (koşul)
	koşul doğruysa
	else
	koşul yanlışsa
		if (! koşul)
		koşul yanlışsa
		else
		koşul doğruysa

Deyimleri başka türlü ifade etmek	X < Y	! (X >= Y)
	++X	X = X + 1
	X += Y	X = X + Y
Operatörlerin sıralamasını değiştirmek	X – Y + Z	Z + X – Y
Operatörleri yaymak	X * (Y + Z)	X * Y + X * Z
	X && (Y || Z)	X && Y || X && Z

Deyimleri bölmek/birleştirmek	x+= y = a + b + c, z = n = foo();	y = a + b + c;
		x += y;
		n = foo();
		z = n;

	x = getSomeValue(); y = x – z;	y = (x = getSomeValue()) – z;

	x = (a + b) * c;	x = a;
		x += b;
		x *= c;

Deyimlerin sıralamasını değiştirmek	x +=PI;
	y = atan (0.9);
	z = x – y;
		y = atan (0.9);
		x +=PI;
		z = x – y;

2.5. KLON KOD TESPİT TEKNİKLERİ

İntihalin ve klon kodun tespiti amacıyla çeşitli yaklaşımlar geliştirilmiştir. Bunların en popülerleri metin tabanlı, simge tabanlı, metrik tabanlı, soyut sentaks ağacı tabanlı ve program bağımlılık grafı tabanlı tekniklerdir (Freire ve diğ., 2007; Jafar, 2007; Jia, 2007).

İntihal tespiti yapmak için her bir kaynak kodun orijinal kodla karşılaştırılması gerekmektedir. Temelde intihal tespit süreci iki aşamadan oluşmaktadır. İlk aşamada verilen program setinin orta seviyede temsili oluşturulur. İkinci aşamada ise her bir klon çifti arasındaki benzerlik bu temsil kullanılarak belirlenir. Programda intihal tespiti için, orta seviyede temsil üreten bir yöntem ve benzerlik değerlendiren algoritma gereklidir. Benzerlik algoritması benzerlik oranının belirlenmesinde kullanılır ve karşılaştırılan iki metin/kod arasındaki intihal edilmiş kısımları bulur (Ji ve diğ., 2007).

Bilgisayarın klon kodları veya metinleri tespit etmesi ancak benzer dokümanların karşılaştırılıp değerlendirilmesiyle mümkündür. Bu nedenle intihal kavramı benzerlik kavramıyla eşanlamlı olarak kullanılır. Benzerlik oranı dosyaların diğer dosyalarla karşılaştırılması ile hesaplanır (Mozgovoy, 2007). Bilgisayarlar intihal tespitinde hangi dosyanın orijinal hangisinin çalıntı olduğunu bilemez. Aynı şekilde kurallara uygun atıf yapılıp yapılmadığını da bilemez. Bunlar intihal tespit sistemlerinin çıkardığı sonuçların yorumlanması ile anlaşılır. Bu nedenle "intihal tespit sistemi" intihalin varlığına ilişkin kesin sonucu vermez (Mozgovoy, 2007).

Literatürde karşılaştırma teknikleri metinsel (*textual*), sözcüksel (*lexical*), sözdizimsel (*syntactic*), anlamsal (*semantic*) ve karışık (*hybrid*) teknikler olarak sınıflandırılmıştır (Roy ve diğ., 2009). Metin tabanlı yaklaşımlar metin tabanlı teknikleri, sözcüksel yaklaşımlar simge tabanlı teknikleri, sözdizimsel yaklaşımlar ağaç tabanlı ve metrik tabanlı teknikleri, anlamsal yaklaşımlar Program Bağımlılık Grafı tekniğini içermektedir. Karma yaklaşımlar ise bu tekniklerin bir arada kullanılmasını içerir.

2.5.1. Metin Tabanlı Teknikler

Doğal diller ve programlama dillerinde intihali tespit için geliştirilmiş en eski ve basit tekniklerden biri metin tabanlı yaklaşımdır (Jia, 2007). Bu teknik kelimeleri ve harfleri dikkate alır ve bunlar satır satır karşılaştırılır.

Bu teknik, düşük seviye kaynak kod intihallerini tespit edebilir (Hamilton, 2008). Kaynak kodda çok basit değişiklikler, biçim ve boşluk değişikliği, yorumları değiştirme gibi işlemler metin tabanlı tekniklerin tespit edebildiği değişikliklere örnek olarak verilebilir. Bu teknikte yorum satırları ve boşluklar sistem tarafından dikkate alınmaz (Komondoor, 2003).

Metin tabanlı teknikler ile metinsel değişiklikler tespit edilebilir. Bu teknikler anlamsal analiz yapamadığı için yapısal değişiklikleri tespit edemez (Jia, 2007). Bu nedenle yapısal değişiklik içeren intihal tiplerini tespit etmekte kullanışsızdır (Mishne, 2003). Bu teknik daha çok alt seviye intihal tiplerini (Tip I - "kopyala yapıştır") ve metin dokümanları arasındaki benzerliği tespit etmekte kullanılır (Tekin ve diğ., 2009).

Program kaynak kodundaki intihali tespit için uygun olmasa da metin benzerliklerini bulmak için birçok dize karşılaştırma algoritması bulunmaktadır. Bunlara kaynak kod benzerliklerinde de uygulanan *Levenshtein distance* ve *Smith-Waterman algoritması* örnek olarak verilebilir (Hamilton, 2008). Sözcüksel analiz veya ayrıştırma gerekli olmadığından bu teknik birçok programlama diline uygulanabilir (BASIT, 2006).

Bu yaklaşımı kullanan klon tespit araçlarına örnek olarak CPD (Copy/Paste Detector) ve Simian verilebilir (Mishne, 2003). Bu programlar tezin sonraki bölümünde ayrıntılı olarak incelenmiştir.

2.5.2. Simge Tabanlı Teknikler

Simgeleştirme tekniğinde programın kaynak kodundaki her bir komut, değişkenler, sabitler vb. simgeleştirilerek bir dosyada saklanmakta ve karşılaştırma bu simge (token) dosyaları üzerinden yapılmaktadır (Jafar, 2007). İntihal edilen kaynak kodda değişken tipi ve ismi değiştirilse bile alacağı simgeler aynı kalacaktır. Bu nedenle kaynak kodda

isim değişikliği yaparak intihali gizleme yöntemi bu teknikte işe yaramamaktadır (Burrows ve diğ., 2007; Hamilton, 2008).

Aşağıda örnek bir C programı (Örnek Program 2.1) ve bu programın simgeleştirilmiş hali Tablo 2.5'te verilmiştir (Burrows ve diğ., 2007).

Örnek Program 2.1

```c
#include <stdio.h>
int main (void) {

    int var;
    for (var=0; var<5; var++) {
        printf("%d\n", var);
    }
    return 0;
}
```

Tablo 2.5: Örnek Program 2.1'in Simgeleştirilmesi (Burrows ve diğ., 2007)

No	Yapı	Simge	No	Yapı	Simge
1	int	S	16	ALPHANUM	N
2	main	N	17	+	D
3	(	A	18	+	D
4	)	B	19	)	B
5	{	j	20	{	j
6	int	S	21	ALPHANUM	N
7	ALPHANUM	N	22	(	A
8	for	R	23	STRING	5
9	(	A	24	,	E
10	ALPHANUM	N	25	ALPHANUM	N
11	=	K	26	)	B
12	ALPHANUM	N	27	}	l
13	ALPHANUM	N	28	return	g
14	<	J	29	ALPHANUM	N
15	ALPHANUM	N	30	}	l

Örnek Program 2.1 simgeleştirilmiş dizimi ise aşağıda gösterilmiştir:

SNABjSNRANKNNJNNDDBjNA5ENBlgNl

Tablo 2.5'te görüldüğü gibi sayı tipinde int değişkeni tanımlamasına S simgesi, değişken isim atamasına ise N simgesi verilmiştir. Değişken tiplerine ve değişken isimlerine verilen simge aynı olacağı için bunlarda değişiklik yaparak kaynak kod intihalini gizlemek bu teknikte işe yaramayacaktır.

Simgelerin benzerlik karşılaştırmasında *suffix tree* ve *n-gram* algoritmaları kullanılır (Jia, 2007; Burrows ve diğ., 2007). n-gram algoritmaları simgeleri olduğu gibi değil verilen n sayısına göre gruplandırır. Her bir simge dizisinin uzunluğu verilen n sayısı kadar olur ve bir simge dizisinin ikinci simgesi bir sonraki simgenin ilk harfi olmaktadır. Böylece simgeler devam eden bir yapıyla sona erer. n-gram algoritması kullanıldığında kaynak kodların yeri değiştirilse dahi benzerlik içeren kısımlar devam eden yapıda ele alınacağı için kopyalanmış kodun tespiti mümkün olmaktadır (Burrows ve diğ., 2007). Örneğin yukarıda simgeleştirilmiş hali verilen programa simgeleştirme tekniğinde n-gram algoritması kullanılırsa ve n değerine 4 verilirse programın yeni simgeleştirilmiş dizimi aşağıdaki gibi olacaktır (Burrows ve diğ., 2007):

Simgelerin 4-gram tekniği ile oluşturulması :

```
SNAB NABj ABjS BjSN jSNR SNRA NRAN RANK ANKN
NKNN KNNJ NNJN NJNN JNND NNDD NDDB DDBj DBjN
BjNA jNA5 NA5E A5EN 5ENB ENBl NBlg BlgN lgNl
```

Bu teknikte her bir satırın simgesel temsilinin çıkartılması gerekmektedir. Bu nedenle karşılaştırma metin tabanlı tekniğe göre daha uzun sürer (Jia, 2007). Simge tabanlı yaklaşım çeşitli dönüşümlerle kodlama stilindeki farklılıkları elimine eder. Bu yönüyle metin tabanlı tekniğe göre göreceli olarak daha güvenilir sonuçlar üretir.

Bu teknikte programın tamamen ayrıştırılıp simge dosyasının oluşturulması için o dile ait ayrıştırıcı gerekmektedir (BASIT, 2006).

Kaynak kodlardaki yorum benzerliği intihalde kanıt olarak kullanılabilir. Fakat bu teknikte yorumlar göz ardı edildiği için onlara simge oluşturulmaz (Burrows ve diğ., 2007). Oysa klon kodlarda yorumların değiştirilmeden bırakıldığı durumlar da olabilir. Bu nedenle yorumların göz ardı edilmesi bu tekniğin zayıf yönleri arasında sayılabilir.

Bu teknik sözdizimsel klonları belirlemede metin tabanlı tekniğe göre daha başarılıdır. Ancak yapısal klonları belirlemede zayıf kaldığından daha çok Tip 1 ve Tip 2 klonların tespitinde kullanılmaktadır (Tekin ve diğ., 2009). CCFinder bu tekniği kullanan araçlara örnek olarak verilebilir (Jiang, 2009).

2.5.3. Metrik Tabanlı Teknikler

Yazılım metrikleri tekniği programdaki istatistiksel değerleri ele alan ve kaynak kodlar yerine bu değerlerle oluşturulan metrik gruplarını karşılaştıran bir yaklaşımdır (Jafar, 2007). Bu metrik gruplarında satır sayıları (bildirim yapılan satır sayısı, yorum satırlarının sayısı, çalıştırılabilir programın satır sayısı vb.), fonksiyon parametrelerinde kullanılan global değişken sayısı, maksimum iç içe blokların sayısı, değişken ve fonksiyon sayıları, döngü sayıları, koşul yapılarının sayısı vb. gibi metrikler kullanılır (Kapser ve Godfrey, 2003). Bu metriklerin hangilerinin kullanılacağına dair genel bir kabul yoktur. Metriklerin seçimi karşılaştırma sonucunu önemli ölçüde etkilemez (Kapser ve Godfrey, 2003). Fakat uygun metriklerin seçimi intihal tespitinde başarımı artırır. Benzer metrikler benzer kaynak kod bölümleri olduğu anlamına gelir (Hamilton, 2008).

Bu teknik, metin tabanlı karşılaştırma yaklaşımına göre daha iyi olsa da programda yer alan istatistikî verilere dayandığı için kaynak kod intihalini tespit etmede en iyi seçenek değildir. Örneğin intihal yapılmış bir programda intihali gizlemek için fazladan değişken veya fonksiyon eklenmiş olabilir. Bu tekniğin çalışma prensibi değişken, koşul, fonksiyon sayıları gibi metrikleri karşılaştırmak olduğundan benzerlik tespitinde doğru ve güvenilir sonuçlar vermeyebilir (Hamilton, 2008; Tekin ve diğ., 2009).

Bu yaklaşım kaynak kodlar arasındaki benzerlik ölçüsünü verir. Kullanıcı en yüksek benzerlik oranına sahip blok çiftlerine bakarak bunların klon kod olup olmadığına karar verir (Komondoor, 2003). Başka bir ifade ile bu teknikte sonuçların değerlendirilmesi için dışarıdan insan müdahalesine ihtiyaç duyulmaktadır (Tekin ve diğ., 2009). Bu teknik yüksek seviye klon tiplerinde kopyalanmış fonksiyonları tespit etmek için kullanılır (Jia, 2007).

2.5.4. Soyut Sentaks Ağacı Tekniği

Soyut Sentaks Ağacı (*Abstract Syntax Trees-AST*) tekniğinde kaynak kodda yer alan yapılar, fonksiyonlar, parametreler vb. her şeyin birbiriyle olan ilişkisi hiyerarşik bir şekilde ağaç yapısında olduğu gibi dallanarak gösterilir. İki kaynak kodun benzerlik ölçümü, bu dalların ve alt dalların karşılaştırması ile yapılır. AST tekniği yapısal karşılaştırma sağladığı için metrik veya dize tabanlı tekniklere göre daha doğru bir karşılaştırma yapar.

Aşağıda örnek bir Java programı (Örnek Program 2.2) ve bu programa bir koşul eklenerek klonlama yapılmış program (Örnek Program 2.3) verilmiştir. Bu iki programın bu metoda göre oluşturulan ağaç grafı ise Şekil 2.13'te verilmiştir.

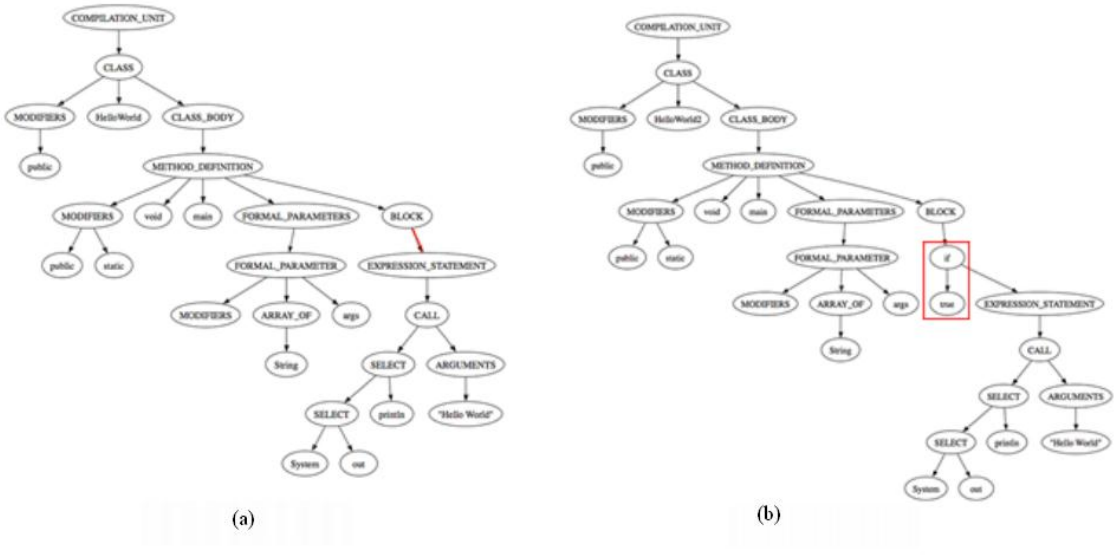
Örnek Program 2.2

```
public class HelloWorld {
    public static void main ( String [ ] args ) {
        System.out.println ( "Hello World" ) ;
    }

}
```

Örnek Program 2.3

```
public class HelloWorld2 {
    public static void main ( String [ ] args ) {
        if ( true )
            System.out.println ("Hello World" ) ;
    }
}
```



Şekil 2.13: Örnek Program 5.2 (a) ve 5.3'ün (b) AST yaklaşımına göre graf gösterimi (Hamilton, 2008)

Şekil 2.13 incelendiğinde Örnek Program 2.3'ün grafında (b), Örnek Program 2.2'nin grafından (a) farklı olarak kırmızı dikdörtgen içinde fazlalık olan kısım gösterilmiştir. Bu, Örnek Program 2.2'ye eklenen *if* koşuludur. Geri kalan kısımlar birbiriyle tamamen aynıdır.

Bu yaklaşım sözdizimsel olduğu için ifadelerin yer değişimine ve değişkenlerin isim değişikliğine duyarlıdır (Komondoor, 2003). Bu teknik daha çok yapısal değişiklik içeren klonları tespit etmede kullanılır. Bu teknikte programın ayrıştırılması uzun sürdüğü için karşılaştırma diğer yöntemlere göre daha fazla zaman alır (TAIRAS, 2010).

2.5.5. Program Bağımlılık Grafı Tekniği

Program bağımlılık grafı tekniği (*Program Dependency Graphs-PDG*), programın veri akışının ve kontrol bağımlılıklarının graf olarak temsilidir (Hamilton, 2008). Bu teknikte program yapıları düğümlerle, veri ve kontrol bağımlılıkları da çizgilerle temsil edilir. Klonlar eş biçimli (izomorfik) grafların karşılaştırılmasıyla belirlenebilir (Jafar, 2007). İki graf arasındaki benzerlik, aynı dallanma yapısında benzer kenarlığa ve köşelere sahip özelliklere bakılarak tanımlanır.

PDG tekniği AST tekniğine benzer şekilde graf karşılaştırma algoritması kullanır. Karşılaştırmada alt dallanmaların sayısı, grafların birbirine uzaklığı gibi parametreler kullanılır (Hamilton, 2008).

PDG tekniği programlama dilinin yapısına bağımlıdır ve veri yapıları boyutlandırılabilir değildir (BASIT, 2006).

2.5.6. Hibrit Yöntemler

Benzerlik karşılaştırmasında her tekniğin üstün özelliklerinden faydalanılarak benzerlik oranını ve benzeşen kısımları daha iyi tespit edebilmek amacıyla yukarıda sayılan tekniklerin birlikte kullanıldığı yaklaşımlar da geliştirilmiştir (Tekin ve diğ., 2009). Roy ve diğ. (2009) sözdizimsel ve AST tabanlı anlamsal tekniklerin benzerlik karşılaştırmalarında bir arada kullanıldığını ifade etmiştir.

2.5.7. Tekniklerin Karşılaştırılması

Karşılaştırmada kullanılan her bir tekniğin bir diğerine göre üstün ve zayıf olduğu noktalar vardır. Örneğin bazı teknikler karşılaştırma işlemini diğerlerine göre çok daha hızlı yapar fakat kaynak koddaki yapısal değişiklikleri tespit edemez. Bazı tekniklerle kodun karşılaştırılması daha uzun sürer fakat daha doğru benzerlik oranları verir ve yapısal değişiklikleri tespit eder (TAIRAS, 2010). Örneğin bağımlılık grafı tabanlı tekniklerin klon tiplerini tespiti diğerlerine göre daha kapsamlıdır fakat yavaş çalışırlar. Yazı ve simge tabanlı teknikler daha düşük seviye klon tiplerini tespit edebilir. Metrik tabanlı tekniklerin başarımı ise diğerlerine göre daha düşüktür (Tekin ve diğ., 2009).

3. MALZEME VE YÖNTEM

Malzeme ve yöntem kısmında kaynak kod intihalini ve klon kodları tespit etmek için geliştirilmiş olan 9 farklı yazılım incelenmiş ve kullanılmıştır. Bu yazılımları kullanmak için Windows7 64bit, Windows XP 32bit, Ubuntu 11.10 (Oneiric Ocelot) 64bit işletim sistemleri ve sanal işletim sistemi yönetim yazılımı Oracle VirtualBox v.4.1.6 kurulmuş ve kullanılmıştır. Ayrıca kullanılan yazılımların gereksinim duyduğu Eclipse, GNU C derleyicisi, Python derleyicisi, Perl derleyicisi, Java Runtime Environment (JRE), Java Development Kit (JDK) ve Notepad++ v.5.9.3 gibi yazılım, kütüphane, derleyici gibi diğer araçlar da kurulmuştur.

Tezde kaynak kod benzerlik ve intihal tespit yazılımlarına ek olarak mahkemelere intikal etmiş iki farklı kaynak kod intihal davası da incelenmiştir. Bu davalarda şüphelenilen kaynak kod intihalini tespit için kaynak kod intihalini tespit eden yazılımların kullanılıp kullanılmadığı ve kullanılanların hangileri olduğu araştırılmıştır. Yalnız bu kısımda incelenen davalarda söz konusu firmaların ve yazılımların mahremiyetleri ve marka değerleri ve kişilerin özel hayatları göz önüne alınarak davalarda adı geçen kişi, şirket ve yazılım isimleri yerine A, B gibi kodlama yapılmış, davaya ilişkin sadece Dava Dosya Numarası ve davanın görüldüğü mahkeme adı verilmiştir.

İncelenen davaların mahkemeye intikal seyri özetlenmiş, özellikle de kaynak kod intihalini tespit etmeleri amacıyla görevlendirilen teknik bilirkişilerin incelemeleri üzerinde durulmuştur. Davaların sonuçları yukarıda sayılan sebeplerden ötürü açıklanmamış sadece bilirkişilerin kullandıkları kaynak kod intihal tespit yazılımlarının neler olduğu veya kullanmadılarsa benzerlik veya intihal tespitini nasıl yaptıkları ve bilirkişi raporlarına ne yazdıkları incelenmiştir.

3.1. KAYNAK KOD BENZERLİĞİ VE KLON KOD TESPİT ARAÇLARI

Tezin önceki bölümlerinde değinildiği gibi kaynak kod, normal metinlere göre çok daha kurallı bir yapıya sahiptir. Bu nedenle kaynak kodlar arasında benzerlik karşılaştırması yapmak ve var olan benzerliği tespit etmek normal metinlere göre daha güçtür. Bu nedenle benzerlik ölçümüyle ilgili daha güvenilir sonuçlar alabilmek için özel olarak kaynak kodlardaki benzerliği tespit eden araçların kullanılması gerekmektedir. Kaynak kod benzerliğinin tespiti amacıyla tezin önceki bölümünde değinilen yaklaşımlar temel alınarak çeşitli kaynak kod benzerlik ve klon tespit araçları geliştirilmiştir. "Kaynak Kod Benzerlik Tespit Araçları" başlığında ayrıntılı olarak incelenene klon kod ve intihal tespiti yapan araçlardan MOSS ve JPlag gibi intihal tespit yazılımları benzerlik oranı vermekte, CPD ve Duplo gibi klon kod tespit yazılımları ise sadece kopyalanan satırları göstermektedir. Yine bu araçlardan Plaggie gibi araçlar sadece bir programlama dilindeki intihali tespit etmek amacıyla geliştirilmişken CCFinder, Simian gibi bazı araçlar ise birden fazla programlama dilinde intihal tespiti yapabilmektedir.

Tezin bu kısmında mahkemelerden benzerlik tespiti yapabileceğimiz iki farklı program edinme imkânımız olmadığından *7zip* dosya sıkıştırma yazılımının 9.10, 9.21 ve 9.22 versiyonlarında benzerlik tespiti yapılmıştır. Sadece iki kaynak kod dosyasının karşılaştırma örneklerinde ise yine aynı yazılıma ait XzEnc.c isimli C kaynak kodu, dosya adının sonuna karşılaştırıldığı versiyonun numarası eklenerek XzEnc_910.c veya XzEnc_922.c şeklinde gösterilmiştir.

3.1.1. Kaynak Kod Benzerlik Tespit Araçları

Kaynak kod benzerliği tespitinde kullanılan en popüler araçlar MOSS, JPlag, CCFinder, YAP3, Sherlock, SIM, Simian ve Plaggie gibi kod karşılaştırma araçlarıdır (Mozgovoy, 2007). Bu araçlardan tezde ayrıntılı olarak incelenenlerin (MOSS, JPlag, CCFinder, Sherlock, SIM, Simian, CPD, Duplo, Plaggie) örnek kullanımları ve ekran görüntüleri verilmiştir. Bununla birlikte lisans sorunu, yazılımın denem sürümü olması ve deneme sürümünün kullanımının açıklanmaya değer olmaması, yazılımın artık dağıtılmaması veya diğer yazılımların kullanıldığı mevcut platformlarda çalıştırılabilen versiyonlarının bulunmayışı veya ücretli bir yazılım olmaları gibi sebeplerle tezde ayrıntılı olarak incelenemeyen Plague, YAP ve CodeMatch yazılımlarının sadece çalışma prensipleri açıklanmaya çalışılmıştır.

Tablo 3.1: İncelenen ve klon kod ve kaynak kod intihal tespit yazılımları

Araç	Tezde kullanılma durumu
MOSS	✓
JPlag	✓
CCFinder	✓
Sherlock	✓
SIM	✓
Simian	✓
CPD	✓
Duplo	✓
Plaggie	✓
Plague	✗
YAP	✗
CodeMatch	✗

3.1.1.1. MOSS

1994 yılında Stanford Üniversitesi'nde Alex Aiken tarafından geliştirilmeye başlanmış olan ve Web tabanlı kaynak kod karşılaştırması yapan MOSS (Measure of Software Similarity), kaynak kod benzerliği tespitinde kullanılan en popüler araçlardan biridir (Hage ve diğ., 2010). MOSS'un kullanımı üyelik gerektirmekte olup, üyelik işlemi ücretsizdir. MOSS benzerlik tespit aracının karşılaştırılacak kaynak kodları kendi sunucusuna gönderen güncel betiği http://theory.stanford.edu/~aiken/moss/ adresinden indirilebilir. MOSS klon tespit araçları arasında en fazla dilde karşılaştırma yapan araçlardan biridir. MOSS C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, HCL2, Prolog, Python ve PL/Sql dillerinde kaynak kod benzerliği tespiti yapabilmektedir (Aiken, 2011).

Simgeleştirme metoduna dayalı yapısal karşılaştırma yapabilen MOSS, kaynak kod karşılaştırmasında Winnowing algoritması kullanmaktadır (Burrows ve diğ., 2007; Hage ve diğ., 2010). MOSS kaynak kod benzerliği tespiti yaparken önce koddaki beyaz boşlukları ve yorum satırlarını silmekte daha sonra kelimeleri n-gram tekniği ile

bölmekte ve bölünen kısımlara hash uygulayarak benzersiz bir sayı vermektedir (Hage ve diğ., 2010). Bu teknikte bölünen bir kelimenin ikinci harfi daha sonraki bölümün ilk harfi olmaktadır. Böylece kelimenin devam eden kısımları da özetlenerek (*hashing*) dokümanın parmak izi (*fingerprint*) çıkarılmış olur. Karşılaştırma da oluşturulan bu parmak izine göre yapılmaktadır (Zeidman, 2004).

MOSS'un kullanımı için sistemde Perl derleyicisinin kurulu olması gerekmektedir. MOSS, 7690. port üzerinden Perl Socket bileşeni ile karşılaştırılacak kaynak kod dosyalarını sunucuya göndermekte ve cevaplar da aynı şekilde gelmektedir. Bu nedenle bu portun bilgisayarda açık olması gerekmektedir.

MOSS karşılaştırma sonucunu kendi sunucusunda HTML sayfası olarak oluşturmakta ve http://moss.stanford.edu/results/karşılaştırma_numarası şeklinde karşılaştırma sonuçlarının yer aldığı sonuç linkini kullanıcıya göstermektedir (Şekil 3.1). Karşılaştırma sonuçları ise 14 gün sonra sunucudan silinmektedir.

Konsoldan komut tabanlı olarak çalıştırılan MOSS çeşitli parametreler almaktadır (Aiken, 2011). MOSS'un kullanımı ve alacağı önemli parametreler aşağıda örneklerle açıklanmıştır:

-l parametresi: MOSS yazılımında karşılaştırılacak kaynak kod dili varsayılan olarak C dilidir. C diline ait komutlar karşılaştırılacaksa bunu *–l* parametresi ile belirtmeye gerek yoktur. Fakat farklı bir dilde karşılaştırma yapılacaksa *–l* parametresi ile dil belirtilmelidir. Bu parametrenin alacağı değerler c, cc, java, ml, pascal, ada, lisp, scheme, haskell, fortran, ascii, vhdl, perl, matlab, python, mips, prolog, spice, vb, csharp, modula2, a8086, javascript, plsql ve verilog şeklindedir.

Örnek kullanım:

```
moss.pl -l java dosya1.java dosya2.java
```

-b parametresi: Temel dosya olarak belirlenecek dosya **-b basefile1** ile gösterilebilir. Bu dosya karşılaştırmaya dâhil edilmeyecek, diğer dosyaların benzerlik tespiti bu dosyaya göre yapılacaktır. Örneğin bulunulan klasördeki tüm C++ kodlarının

`skeleton.cc` koduna göre karşılaştırılması isteniyorsa MOSS parametreleri aşağıdaki gibi kullanılabilir.

Örnek kullanım:

```
moss.pl -l cc -b skeleton.cc *.cc
```

-m parametresi: MOSS uygulamasının karşılaştırma hassasiyetini belirlemek için atanır. Örneğin 2 program karşılaştırılacaksa m parametresine 2 değeri verilebilir. Çok büyük programlarda bu seçenek artırılabilir. Bu parametrenin varsayılan değeri 10'dur. Örneğin –m parametresine 2 verildiğinde MOSS iki dosya arasındaki benzerliği birbirine göre sırasıyla %84 ve %60 olarak hesaplamıştır. Bu parametre değerine 1000000 verildiğinde ise dosyaların birbirine benzerlik oranı sırasıyla %85 ve %61 olarak hesaplanmıştır.

Örnek kullanım:

```
moss.pl -m 2 XzEnc_910.c XzEnc_922.c
moss.pl -m 1000000 XzEnc_910.c XzEnc_922.c
```

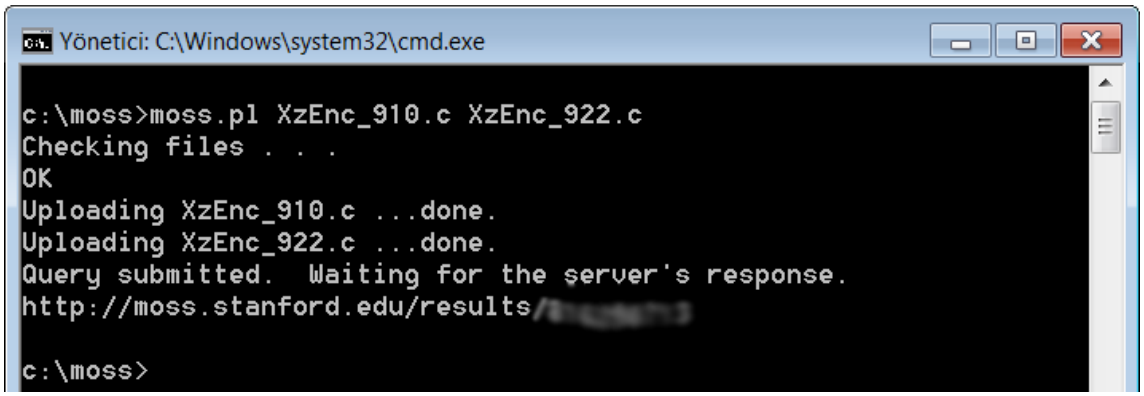
-n parametresi: Sonuçlarda gösterilecek dosya sayısını belirler. Büyük programların tüm dosyalarının sonuçlarda gözükmesi için parametre değeri artırılabilir. Varsayılan değer 250'dir.

MOSS benzerlik aracının kaynak kod intihalini tespit için kullanımı aşağıda örneklerle açıklanmıştır:

- İki C kaynak kod dosyasında karşılaştırma yapan örnek MOSS kullanımı Şekil 3.1'de gösterilmiştir.

Örnek kullanım:

```
moss.pl XzEnc_910.c XzEnc_922.c
```



Şekil 3.1: MOSS ile iki C kaynak kod dosyasının benzerlik tespitinin yapılması

Şekil 3.1'de de görüldüğü gibi `XzEnc_910.c` ve `XzEnc_922.c` isimli iki C dosyasının karşılaştırma sonuçlarının linki kullanıcıya gösterilmiştir. Linkin bağlantısına girildiğinde Şekil 3.2'de görüldüğü gibi karşılaştırmaya ait bilgiler gösterilmektedir. Bunlar karşılaştırma yapılan tarih, karşılaştırmada kullanılan parametreler, karşılaştırılan iki dosyanın ismi, dosyaların birbirine benzerlik oranı ve benzerlik hesabında kullanılan satır sayısı gibi bilgilerdir.

Moss Results

Mon Dec 5 01:27:21 PST 2011

Options -l c -m 10

[How to Read the Results | Tips | FAQ | Contact | Submission Scripts | Credits]

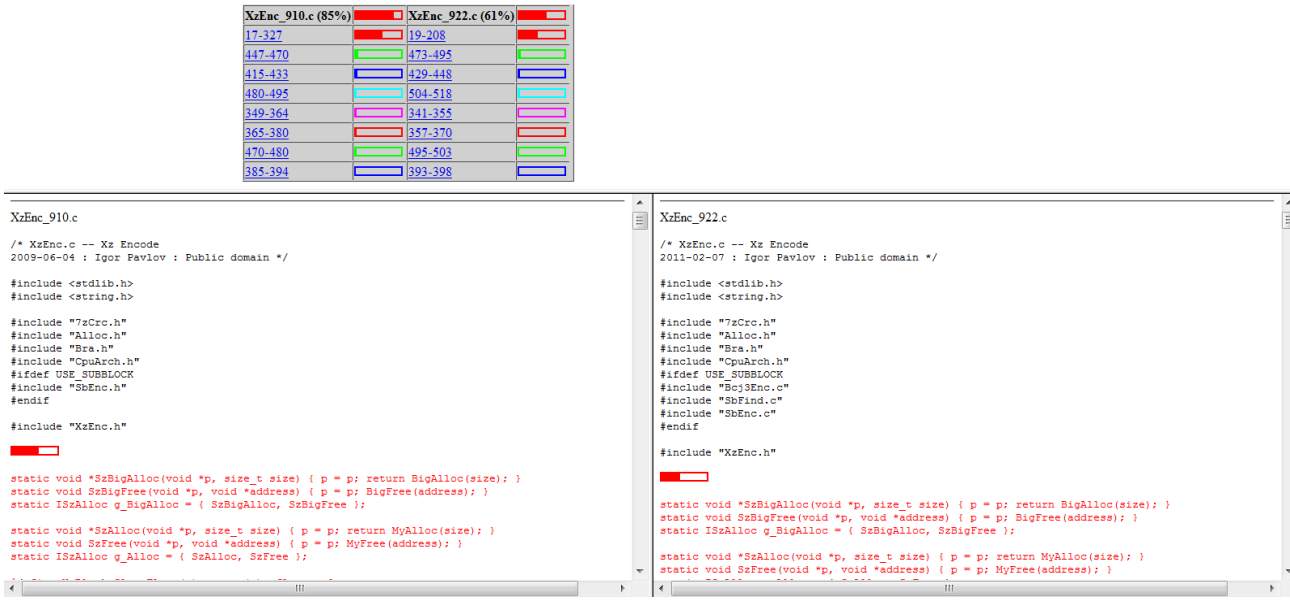
File 1	File 2	Lines Matched
XzEnc_910.c (85%)	XzEnc_922.c (61%)	423

Any errors encountered during this query are listed below.

Şekil 3.2: Karşılaştırma sonucunun MOSS sunucusunda gösterilmesi

Şekil 3.2'de de görüldüğü gibi 7zip yazılımının 9.22 versiyonunun `XzEnc_922.c` isimli C dosyası 9.10 versiyonunun `XzEnc_910.c` isimli C dosyasına %61 oranında benzerlik göstermektedir. 9.10 versiyonu ise 9.22 versiyonuna %85 oranında benzemektedir. Buradan 9.22 versiyonunun `XzEnc_922.c` kaynak kod dosyasında eklemeler veya değişiklikler yapıldığı sonucu çıkarılabilir. Sayfadaki dosya isimlerinin linkine tıklandığında ise klon kod satırları Şekil 3.3'te renklendirilerek gösterilmiştir.

Şekilden de görüleceği gibi `XzEnc_922.c` dosyasının 19-208 arası satırları `XzEnc_910.c` dosyasının 17-327 arası satırlarından klonlanmıştır. Diğer klonlanan satırlar da benzerlik oranlarına büyükten küçüğe listelenmektedir.

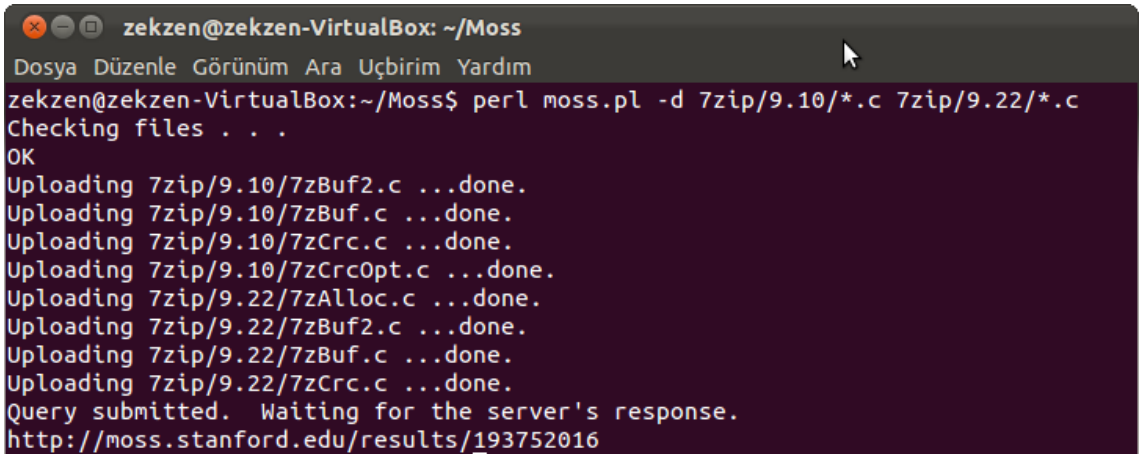


Şekil 3.3: MOSS sunucusunda klonlanan satırların gösterimi

- Klasör tabanlı kaynak kod karşılaştırması yapmak için ise –d parametresi ile karşılaştırılacak klasörler arada bir boşluk bırakılarak yazılmalıdır (Şekil 3.4).

Örnek kullanım:

```
moss.pl -d 7zip/9.10/*.c 7zip/9.22/*.c
```



Şekil 3.4: MOSS'un iki klasördeki C kodlarının karşılaştırması için örnek kullanımı

MOSS kaynak kod karşılaştırmasında kullanılan en önemli ve en popüler araçlardan biridir. Güvenilir sonuçlar vermesi, benzerlik sonuçlarını kullanıcının kolay anlayacağı bir biçimde sunması, ücretsiz kullanımı ve hızlı karşılaştırma yapması avantajları arasında sayılabilir. Bunların yanında üyelik aşamasının uzun sürebilmesi, kullanımı için ek çaba ve deneme gerektirmesi, grafik arayüz olmadan konsoldan kullanımı ve açık kaynak olmaması ise dezavantajları olarak gösterilebilir. MOSS bu alanda geliştirilen ilk yazılımlar arasında olduğundan, kendinden sonra geliştirilen birçok klon kod tespit yazılımına ilham kaynağı olmuştur.

3.1.1.2. JPlag

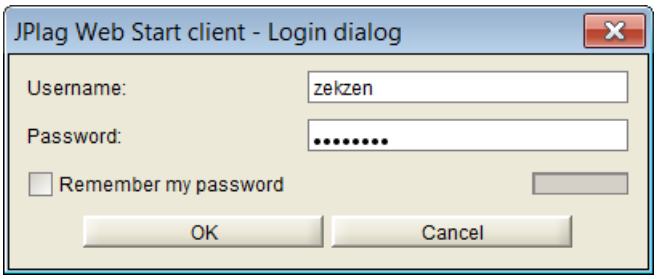
JPlag, *Greedy String Tiling* algoritmasını kullanarak simgeleştirme tekniğine dayalı karşılaştırma yapan kaynak kod intihal tespit aracıdır (Ji ve diğ. 2007; Chilowicz ve diğ. 2009). Karlsruhe Üniversitesi bünyesinde geliştirilen JPlag, MOSS uygulamasındaki gibi kaynak kod karşılaştırma işlemini kullanıcının bilgisayarında değil, Web tabanlı servis şeklinde kendi sunucusunda online olarak yapmaktadır. Karşılaştırılacak kaynak kodlar JPlag'ın sunucusuna gönderilmekte, karşılaştırma sonuçları ise kullanıcının bilgisayarına gelmektedir. JPlag bir Java uygulaması olduğundan çalışabilmesi için bilgisayarda JRE 1.5 ve üzeri kurulu olması gerekmektedir. Sunucuya kaynak kodları göndermeyi sağlayan Java Web Start uygulaması https://www.ipd.uni-karlsruhe.de/JPlag adresinden indirilebilir. JPlag kullanımı ücretsiz bir araç olup kullanımı için sitesinden ücretsiz kayıt olmak gerekmektedir.

JPlag, karşılaştırılacak metnin baytlarını değil, karşılaştırılacak programlama dilinin sözdizimini ve programın yapısını dikkate alarak yapısal karşılaştırma yapar (Elo ve Hasu, 2003; Burrows ve diğ., 2007). Yani karşılaştırılacak programlama dili başta seçilmekte ve o dilin özellikleri dikkate alınarak karşılaştırma yapılmaktadır. JPlag, sunucuya gönderilen her bir kaynak kod dosyasını, ait olduğu programlama dilinin gramerini dikkate alarak inceler ve simge dizilerine dönüştürür. Bu simge dizileri de *Greedy String Tiling* algoritması kullanılarak birbirleriyle karşılaştırılır ve benzerlik ölçümü hesaplanır (Kustanto ve Liem, 2009). JPlag C, C++, Java, C#, Scheme ve doğal dillerde benzerlik tespiti yapabilmektedir (Prechelt ve diğ., 2000).

Konsoldan değil grafik arayüzden kullanılan JPlag, karşılaştırma sonuçlarını HTML Web sayfası olarak sunmaktadır. Karşılaştırılan kaynak kodlar arasındaki benzerlik

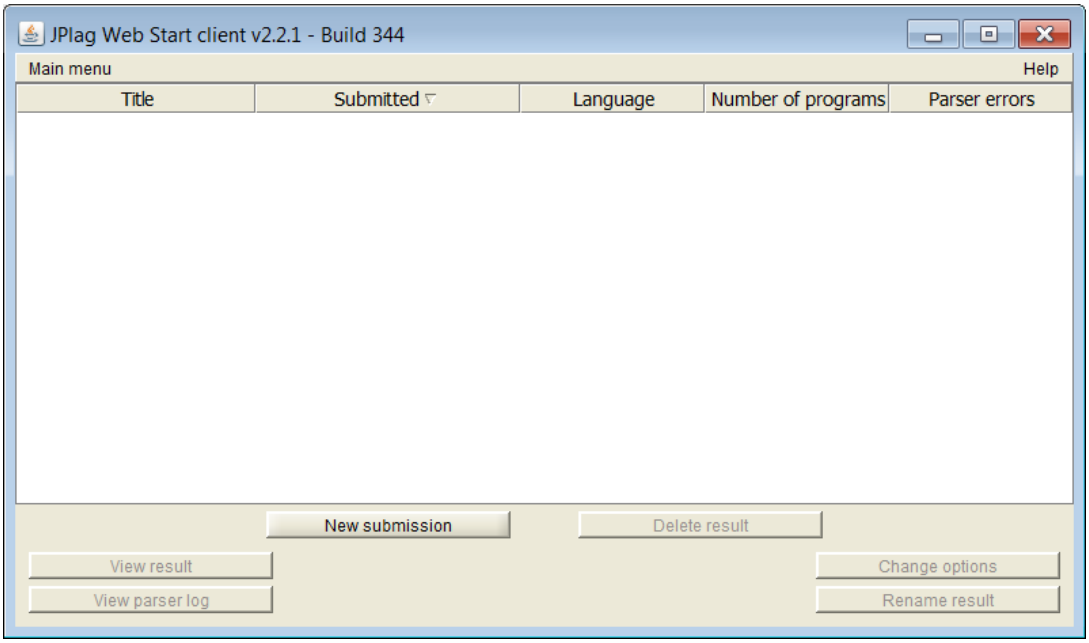
yüzde olarak verilmekte ve benzerlik bulunan kısımlar yan yana listelenmektedir. Kodda yer alan benzerlikler farklı font ve renklerle işaretlenmektedir. Bu özellik kullanıcının sonuçları daha iyi analiz etmesini sağlar (Kustanto ve Liem, 2009).

JPlag, indirilen *start.jnlp* isimli Java Web Start uygulamasının çift tıklanması ile çalıştırılmaktadır. İlk çalıştırmada sunucudan çalışması için gereken dosyalar indirilmektedir. İndirme işleminin bitmesinden sonra sunucuya bağlanmak için kullanıcı adı ve şifrenin istendiği ekran çıkmaktadır (Şekil 3.5).



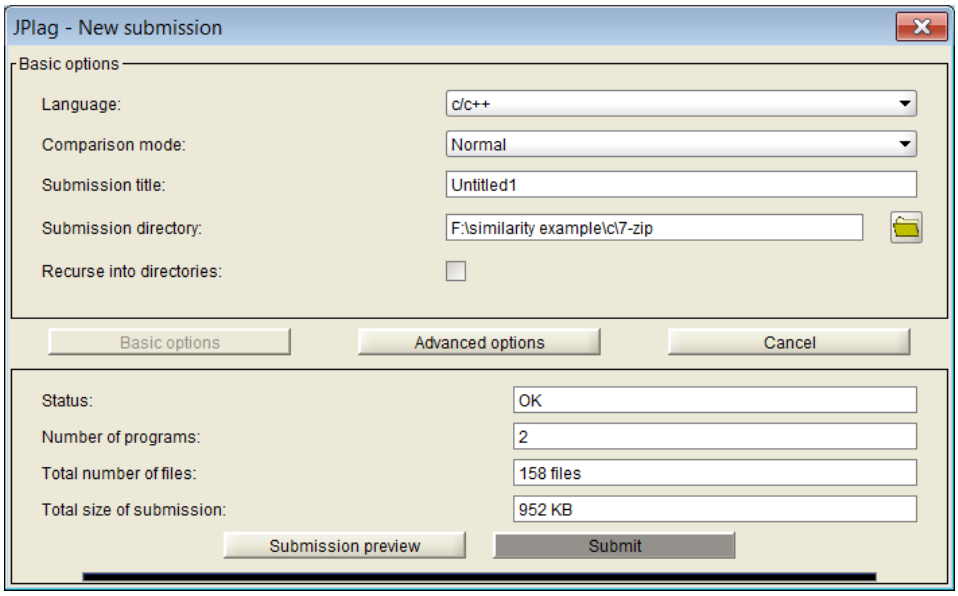
Şekil 3.5: JPlag sunucusuna bağlanma ekranı

JPlag'ın sitesinden kayıt olunarak edinilmiş olan kullanıcı adı ve şifre ile sisteme girildikten sonra JPlag'ın ana ekranı gelecektir (Şekil 3.6).



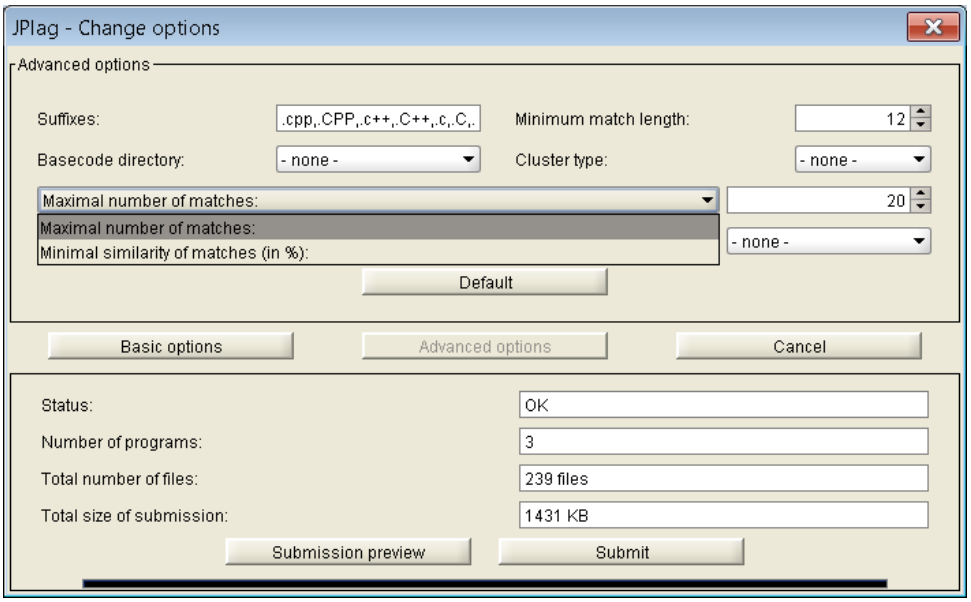
Şekil 3.6: JPlag ana ekranı

Ana ekranda *New Submission* butonuna tıklanarak karşılaştırılacak kodun programlama dilinin seçimi, kod dosyalarının bulunduğu klasörün seçimi ve alt klasörlerin dikkate alınıp alınmayacağıyla ilgili ayarların yapıldığı ekran gelecektir (Şekil 3.7).



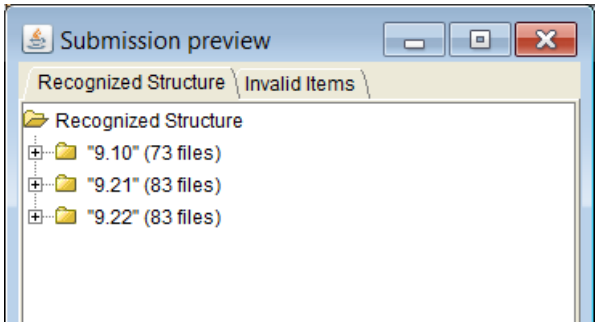
Şekil 3.7: JPlag karşılaştırılacak kaynak kod dosyalarını veya klasörü seçme ekranı

Karşılaştırılacak kodların programlama dili ve klasörü seçildikten sonra *Advanced options* butonuna tıklanarak karşılaştırmayla ilgili gelişmiş ayarlar yapılabilir. Burada karşılaştırmada dikkate alınacak dosya uzantıları (*Suffix)*, ölçümde dikkate alınacak minimum karakter uzunluğu (*Minimum match length*), maksimum ölçüm sayısı (*Maximal number of matches*), istenilen oranın üzerinde benzerlik gösteren kodları listelemek için minimum benzerlik yüzde oranı (*Minimal similarity of matches in %)* ve temel kod klasörünün belirlenebileceği (*Basecode directory)* gibi ayarlar yapılabilmektedir (Şekil 3.8).



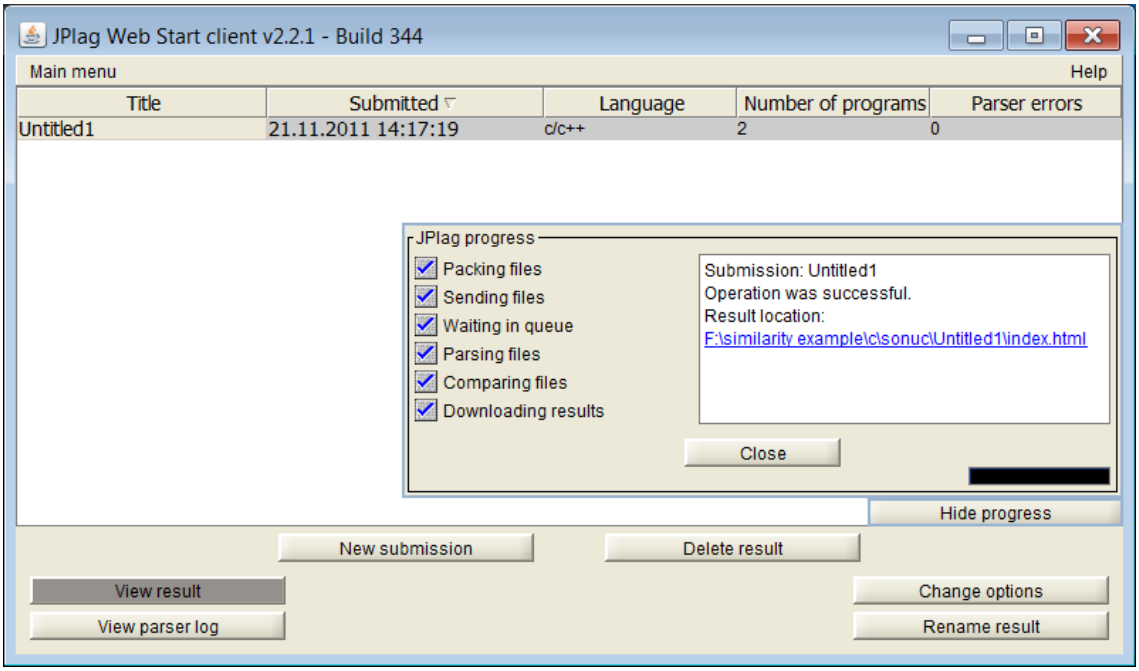
Şekil 3.8: Karşılaştırmayla ilgili gelişmiş ayar yapılmasını sağlayan JPlag ekranı

Sunucuya gönderilecek dosyaların kaç tane olduğu ve hangi klasörlerde bulunduğu *Submission preview* butonuna tıklanarak görülebilir (Şekil 3.9).



Şekil 3.9: Sunucuya gönderilecek dosyaları gösteren JPlag ekranı

İstenilen ayarların yapılmasından sonra *Submit* butonuna tıklanarak karşılaştırılacak kodların JPlag sunucusuna gönderilmesi ve benzerlik ölçümünün yapılması sağlanır. Bu aşamada Şekil 3.10 ekranı çıkarak işlemin hangi aşamada olduğu kullanıcıya bildirilecektir.



Şekil 3.10: Kaynak kodların benzerlik tespiti için JPlag sunucusuna gönderilmesi

Gönderim işleminin başarıyla yapılmasının ardından JPlag karşılaştırma sonuçlarını programda *Main menu* ayarlarında belirlenmiş klasöre aktaracaktır.

Dosyaların sunucudan bilgisayara yüklenmesinin ardından bilgi ekranında yer alan sonuçların kopyalandığı klasörün yolunu içeren linke (*Result location*) veya *View result* butonuna tıklanarak karşılaştırma sonuçları görüntülenebilir. Şekil 3.11'de 7zip programının 9.10, 9.21 ve 9.22 versiyonlarının C kodları arasındaki benzerlik oranları görülmektedir.

JPlag Search Results

Title:	Untitled1
Directory:	F:\similarity example\c\7-zip
Programs:	9.10 - 9.21 - 9.22
Language:	C/C++ Scanner [basic markup]
Submissions:	3
Matches displayed:	3 (Treshold: 83.9%) (average similarity) 3 (Treshold: 96.0%) (maximum similarity)
Date:	2011-11-21
Minimum Match Length (sensitivity):	12
Suffixes:	.cpp, .CPP, .c++, .C++, .c, .C, .h, .H, .hpp, .HPP

Distribution:

90% - 100%	1	##################################
80% - 90%	2	##
70% - 80%	0	.
60% - 70%	0	.
50% - 60%	0	.
40% - 50%	0	.
30% - 40%	0	.
20% - 30%	0	.
10% - 20%	0	.
0% - 10%	0	.

Matches sorted by average similarity (What is this?):

9.22	->	9.21 (99.3%)	9.10 (83.9%)
9.21	->	9.10 (83.9%)	

Matches sorted by maximum similarity (What is this?):

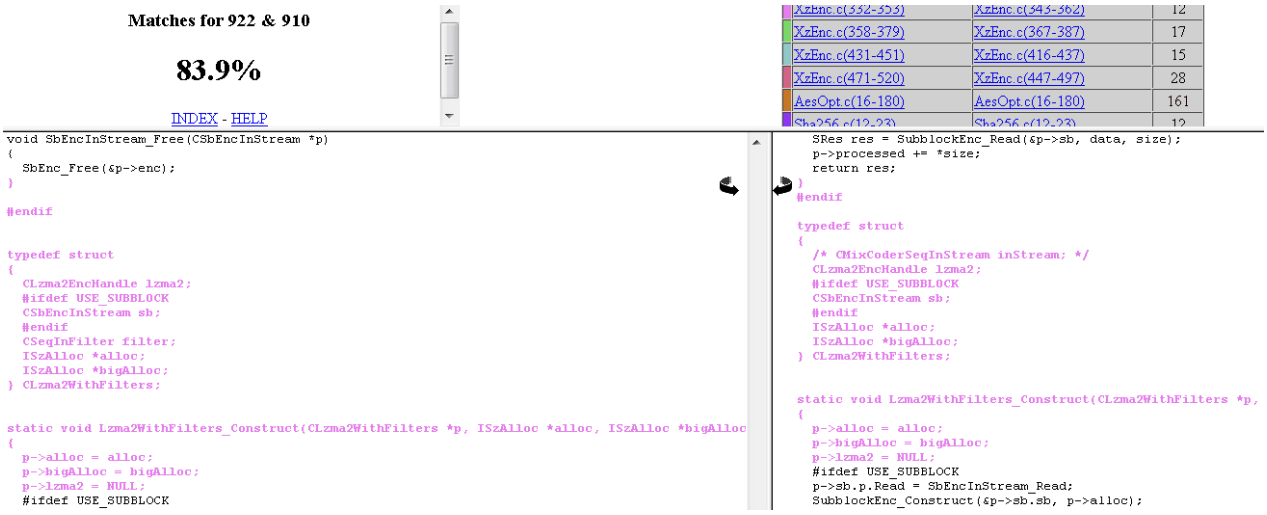
9.22	->	9.21 (99.0%)	9.10 (96.0%)
9.21	->	9.10 (96.0%)	

Şekil 3.11: JPlag karşılaştırma sonuçlarının gösterimi

Programlarda versiyonlama (sürümleme - sürüm numarası verme) resmi bir kural olmasa da genelde sürümlerde yapılan değişikliğin boyutuna ve yapısına göre yapılır. Örneğin x.yz şeklinde devam eden program versiyonlarında x, moderate versiyon numarasını, y major versiyon numarasını, z ise minor versiyon numarasını gösterir. Yapısal anlamda ve çok büyük değişim gösteren program versiyonlarında x moderate numarası artırılırken, ufak değişiklikler içeren güncellemelerde minor numarası olan z, nispeten daha büyük değişiklik içeren güncellemelerde ise major numarası olan y artırılarak versiyonlama yapılır (Fisher, 2011).

7zip programında sürüm numarasına bakılarak ufak değişiklikler içerdiği düşünülen 9.21 ve 9.22 versiyonları arasındaki benzerlik oranı %99.3 iken versiyon numaralarına

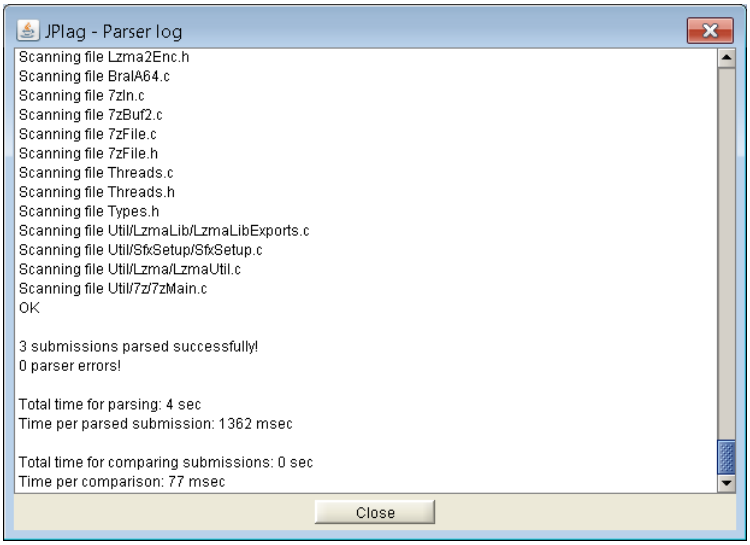
bakılarak nispeten daha az benzerlik bulunacağı tahmin edilen 9.22 ile 9.10 versiyonları arasındaki benzerlik oranı ise %83.9 olarak hesaplanmıştır. Bu oranların hemen altında da kaynak kod dosyaları arasındaki maksimum benzerlik oranları listelenmiştir. Bu oranların üzerinde yer alan linklere tıklanarak karşılaştırılan klasörler/programlar arasındaki benzerlik dosya dosya görülebilir (Şekil 3.12).



Şekil 3.12: JPlag karşılaştırma sonuçlarının gösterimi

Şekil 3.12'den de görülebileceği gibi ekranın sağ üst kısmında yer alan dosya adlarına tıklandığında karşılaştırılan kodlar alt kısımda listelenecektir. Burada sağa ve sola doğru dönük oklara tıklandığında, ilgili kod satırları yan yana getirilecektir. Şekil 3.12'den de görüleceği üzere kodlar farklı renklendirmeye sahiptir. Değişen, eklenen, çıkarılan veya aynen kopyalanan kod satırları farklı renklendirmelerle kullanıcının daha rahat anlayacağı bir formda sunulmaktadır.

Karşılaştırma sonrası *View parser log* butonuna tıklanarak işlenmeye alınan kaynak kod dosyalarının başarı durumu, işlenme ve karşılaştırma işlemine ait toplam süreye de bakılabilir (Şekil 3.13).



Şekil 3.13: Karşılaştırılan dosyaların işlenme durumunu gösteren JPlag ekranı

JPlag kaynak kod benzerlik tespit yazılımının ücretsiz olması, kullanım kolaylığı sağlayan grafik arayüzü, sonuçları kullanıcının anlayacağı şekilde sunması, karşılaştırmaya ilişkin sunduğu ayarlar ve güvenilir sonuçlar vermesi ile kaynak kod benzerlik tespitinde MOSS ile birlikte en fazla kullanılan araçlardandır.

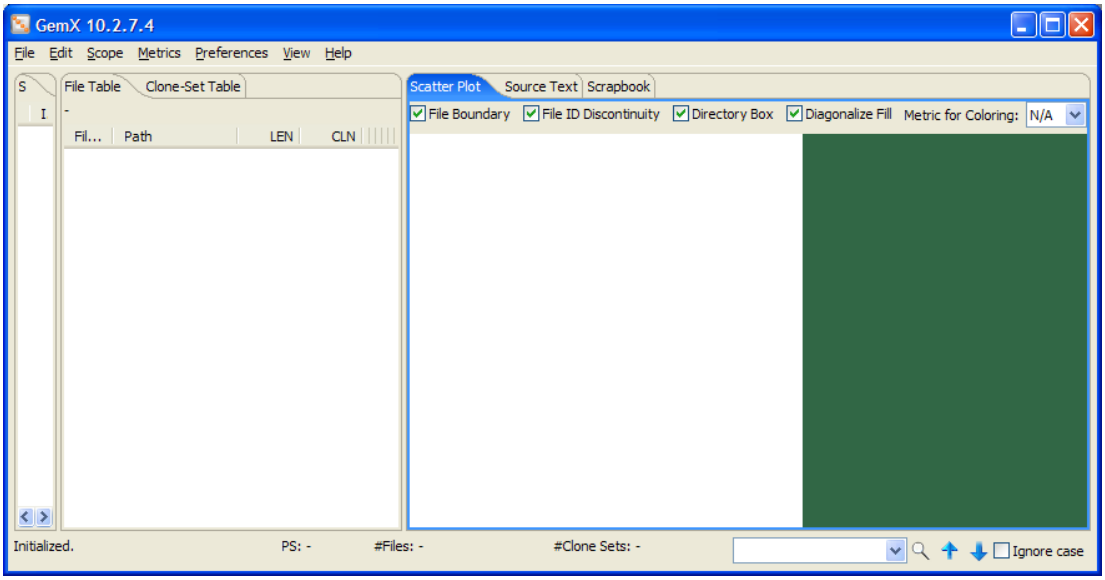
3.1.1.3. CCFinder

Simgeleştirme metoduyla *suffix-tree* algoritmasını kullanarak karşılaştırma yapan CCFinder (Code Clone Finder), karşılaştırma sonuçlarını saçılma grafiği (*scatter plot*) üzerinde görselleştirerek sunmaktadır (Kamiya ve diğ., 2002). Java, C/C++, C#, Visual Basic (VB) ve COBOL dillerinde karşılaştırma yapabilen CCFinder, farklı kod satırlarındaki farklı isimlendirmelere sahip klonları bulabilmektedir (Kamiya ve diğ., 2002; AIST, 2010). Toshihiro Kamiya tarafından yazılan ve ücretsiz kullanılabilen CCFinder, MIT Lisansıyla dağıtılmakta, konsoldan ve grafik arayüzden kullanılmakta ve Windows ve Linux işletim sistemlerinde çalışabilmektedir. http://www.CCFinder.net/CCFinderxos.html adresinden program ve kaynak kod olarak indirilebilen CCFinder'i kullanabilmek için sistemde 1 GB ve üzeri bellek, Sun JavaVM. 1.5 veya üzeri ve Python 2.6 ve üzeri (3.x ile çalışmamaktadır) derleyicinin kurulu olması gerekmektedir.

Büyük boyutlu yazılım sistemlerindeki kaynak kod benzerliğini tespit edebilen CCFinder, karşılaştırılacak kaynak kodu simgelere çevirir ve karşılaştırmayı bu

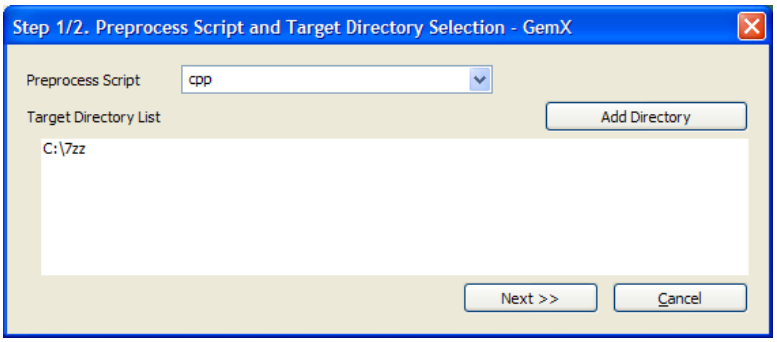
simgeler üzerinden yapar. CCFinder benzerlik sonuçlarını çizim üzerinde noktalama yöntemi (*dotplotting*) kullanarak görselleştirir (Burd ve Bailey, 2002).

CCFinder'in kurulu olduğu dizinde *bin* klasöründeki *gemx.bat* toplu iş dosyasına çift tıklanarak çalıştırılabilen CCFinder'in grafiksel ana ekranı Şekil 3.14'te gösterilmiştir.



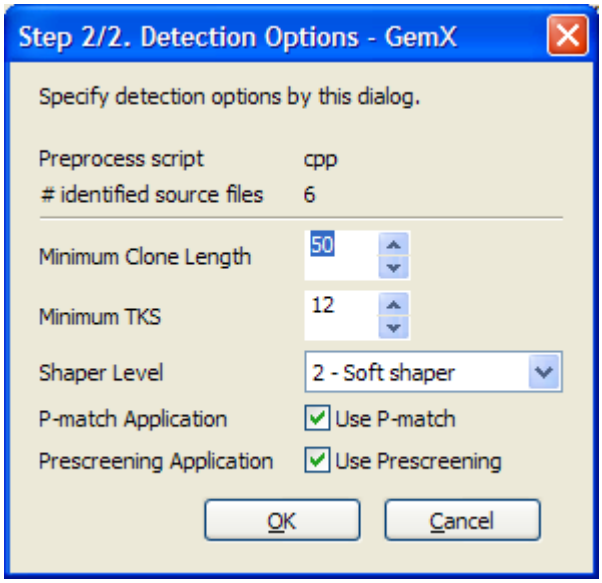
Şekil 3.14: CCFinder ana ekranı

Yeni bir karşılaştırma yapmak için menüden *File->Dedect Clones* seçildiğinde karşılaştırılacak kaynak kodların bulunduğu klasörün ve programlama dilinin seçiminin yapılacağı ekran gelecektir (Şekil 3.15).



Şekil 3.15: CCFinder karşılaştırılacak kaynak kodun programlama dilinin ve klasörünün seçimi

Gereken klasör ve dil seçimi yapıldıktan sonra *Next>>* butonuna tıklanarak karşılaştırmada kullanılacak ayarları gösteren ekran gelecektir (Şekil 3.16).



Şekil 3.16: CCFinder kod karşılaştırma ayarlarının yapıldığı ekran

Bu ekranda bulunan ayarlar aşağıda açıklanmıştır:

Minimum Clone Length: Kod karşılaştırmasında dikkate alınacak minimum klon kod uzunluğu ayarıdır.

Minimum TKS: Bu ayar ile bir simge dizisinde kaç simge bulunacağı yani simge dizisinin uzunluğu belirlenir. CCFinder, simgeleştirme metodu kullandığından her dosyanın kendine ait simge dizilerinden oluşan bir dosyası olacaktır. Örneğin CCFinder'in bir kaynak kod dosyasına aşağıdaki gibi 12 simgeden oluşan bir simge dizisi oluşturduğunu varsayalım:

"a b c x y 1 a b c 2 x y"

Başka bir dosyada da "a b c" ile devam bir simge dizisi olduğunu ve karşılaştırmada kullanılacak minimum TKS ayarının 3 verildiğini varsayalım. CCFinder bu klonu tespit edecektir. Fakat "x y", "a b", ve "b c" şeklindeki klonları simge dizilerinin uzunluğu 2 olduğu için tespit edemeyecektir (Ma ve Woo, 2008).

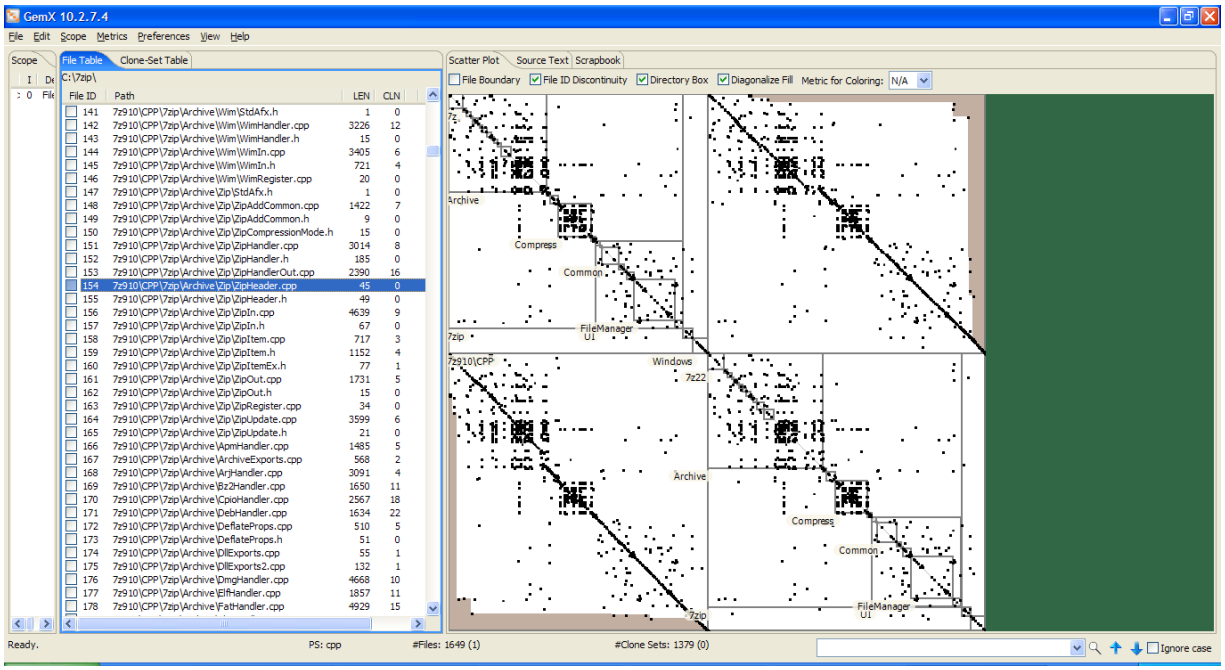
Shaper Level: Klon kodların birbirine ve içinde bulundukları klasör yapısına göre görselleştiren CCFinder'de *scatter plot* haritasında oluşturulacak şekil seviyesi *Shaper*

Level ayarı ile belirlenebilir. Bu ayarın sunduğu seçenekler *Hard shaper, Soft shaper, Easy shaper ve Without shaper* olmak üzere 4 tanedir. *Hard shaper* ayarında sadece bir kod bloğunun ihtiva ettiği bir simge dizisi klon adayı olarak kabul edilir. Soft shaper ayarında, harici bir blok sınırına bölünmemiş bir simge dizisi klon adayı kabul edilmektedir. Easy shaper ayarında keyfi bir simge dizisi bir klon adayı olarak kabul edilir, ancak bölünmemiş simge dizisi de uzunluk ölçümünde kullanılır. Without shaper ayarında blok sınırları ihmal edilir yani keyfi herhangi bir simge dizisi bir klon adayı olabilir.

P-match application: Bu ayarın seçimi ile fonksiyon ve değişken isimlerine ayrı ayrı özel simge verilip verilmeyeceği belirlenir. Örneğin bu ayar seçilmezse kodda yer alan "return x + y" ve "return a + a", "return $ + $" formuna çevrilerek isimlendirmedeki farklılıklar ihmal edilecektir. Eğer bu ayar aktif edilirse "return x + y", "return $1 + $2" olarak "return a + a" ise "return $1 + $1," olarak ele alınarak, aralarındaki benzerlik klon çifti olarak saptanamayacaktır (Ma ve Woo, 2008).

Prescreening application: Bu seçenek çok fazla klon çifti içeren karşılaştırmalarda kolaylık sağlamaktadır (Kamiya, 2008).

İstenilen ayarların yapılmasının ardından karşılaştırma sonrası Şekil 3.17'deki gibi CCFinder ekranı oluşacaktır.



Şekil 3.17: CCFinder karşılaştırma sonrası sonuç ekranı

Karşılaştırma sonrası CCFinder programının sol orta bloğunda yer alan *File Table* sekmesinde karşılaştırılan kaynak kod dosyaları yer almaktadır (Şekil 3.17). Bu sekmede karşılaştırılan dosyaların kimlik numarası (*File ID*), klasör konumu (*Path*), içerdiği simge uzunluğu (*LEN*) ve o kaynak kod dosyasında bulunan klon kod sayısı (*CLN*) yer almaktadır. Sekmenin en üstünde ise dosyaların yer aldığı ortak klasör yazmaktadır.

Şekil 3.17'de orta kısmın sağında yer alan *Scatter Plot* sekmesinde karşılaştırma sonrası tespit edilen klon kodlar köşegenle kesilmiş kareler içerisinde gösterilmektedir. Bu kareler üzerinde yer alan köşegenin bir tarafı karşılaştırılan iki klasörden birinde bulunan kaynak kodları temsil etmekte, köşegenin diğer yarısı da karşılaştırılan diğer klasördeki kaynak kod dosyalarını temsil etmektedir. Bu dosyalar köşegene göre birbirine ne kadar simetrikse benzerlik o oranda çoktur.

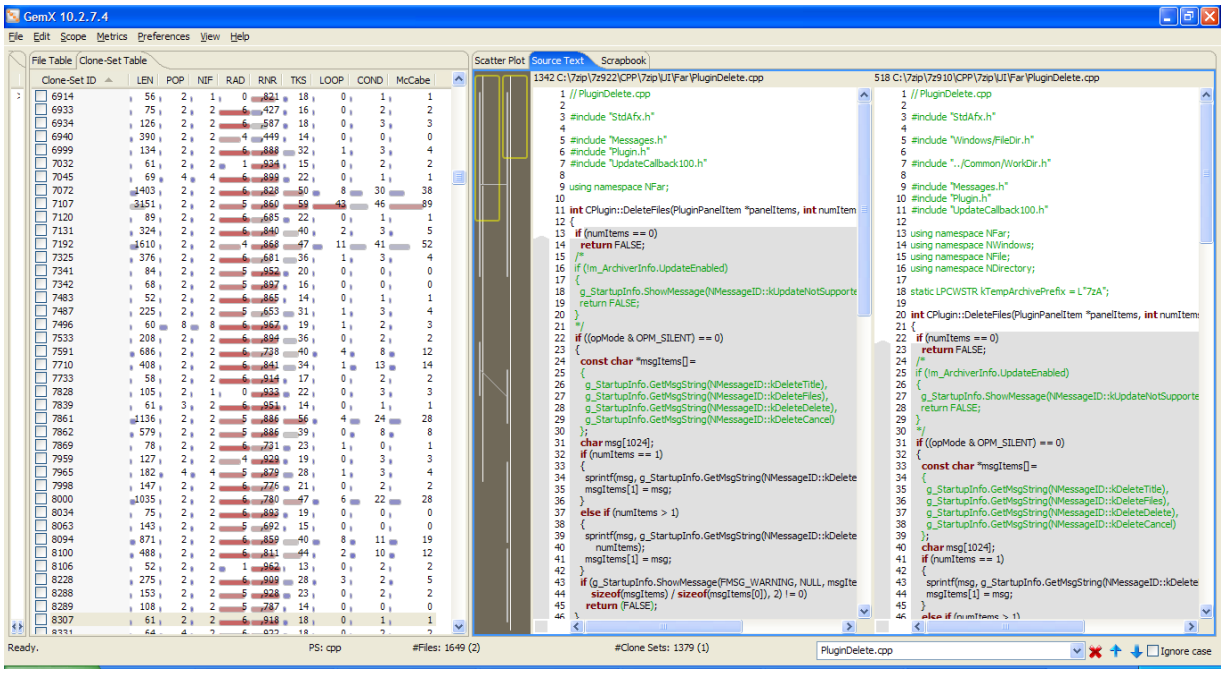
Scatter Plot sekmesinde yer alan köşegen, karşılaştırılmak için CCFinder'a gönderilen klasörleri, köşegene paralel konumlanan siyah çizgiler ve noktalar ise klon kodlarını temsil etmektedir. Yatay ve dikey gri çizgiler ise kaynak kodun sınırlarını yani

klasördeki konumunu göstermektedir. Yatay ve dikey eksende ortogonal izdüşümler benzer kod parçalarını temsil etmektedir.

Karşılaştırmada tespit edilen klonlar ise *Clone-Set Table* sekmesinde listelenmektedir (Şekil 3.18). Bu sekmede yer alan *Clone-Set ID* klonun ID numarasını, *LEN* klon kodun içerdiği simge uzunluğunu, *POP* (*Population*) klon koddaki kod parçalarının sayısını gösterir. POP ne kadar büyükse benzer kod kısımlarının birçok yerde bulunduğu anlamına gelmektedir. *NIF* bir veya daha fazla klon kod parçası içeren kaynak kod sayısını temsil eder. *RAD* (*Radius*) ise şöyle tarif edilir: Bir C klon sınıfı ve de her birinin içeriğinde C kodundan bölümler barındıran F dosyalar dizisi olsun. Eğer C klon sınıfının tüm kod parçaları tek bir dosyada ise RAD (C) = 0'dır. RAD değeri ne kadar büyükse klon kodlar tüm sisteme yayılmış demektir. *RNR* (*ratio of non-repeated tokens*) ise klon kodda tekrarlanmayan kısımlarının oluşturduğu simgelerin tüm koda yüzde olarak oranıdır. *TKS* (*token set size*) klon kod kısımlarının simge dizisinin büyüklüğüdür. *LOOP, COND* kod bölümünde sırasıyla döngü ve koşul dallanmalarının sayısıdır. *McCabe* ise bu ikisinin toplamıdır (Kamiya ve diğ., 2002; Kamiya, 2008). Bu metriklere ayrıca menüden *Metrics->Show Clone-Set Metrics* ile de ulaşılabilir (Şekil 3.20).

CCFinder ana ekranında ortanın sağında yer alan *Source Text* sekmesinde ise seçilen iki kaynak kod dosyasındaki klon kod satırları arka planı gri renkli olarak gösterilmektedir (Şekil 3.18). Sekmenin solunda yer alan ve dikine inen ince gri çizgiler klon kodların bulunduğu satırları göstermektedir. Bu çizgilerin hemen üzerinde yer alan iki adet sarı dikdörtgen bu çizgilerin olduğu yere indirildiğinde klon kodlar *Source Text* sekmesinde yan yana gelecektir. İki bölmeye ayrılmış bu sekmede karşılaştırılan kaynak kod dosyalarının üzerinde kaynak kod dosyasının ID numarası ve bulunduğu klasördeki yolu yazmaktadır.

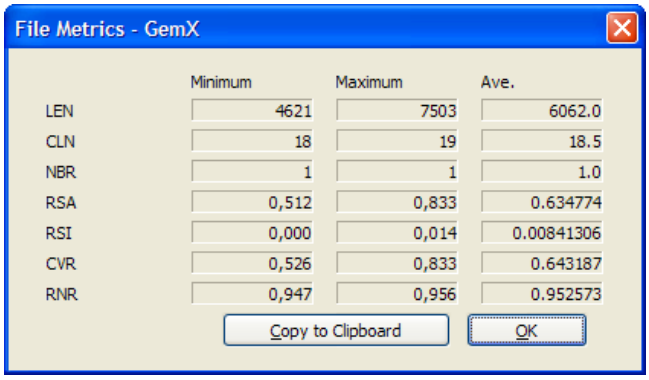
Scrapbook sekmesi ise, *Source Text* sekmesinde ilgilenilen kaynak kodların kopyalandığı kısımdır.



Şekil 3.18: CCFinder Source Text sekmesinde iki kaynak kod dosyasındaki klon kodların gösterilmesi

CCFinder durum çubuğunda seçilen dilin uzantısı, karşılaştırılan toplam dosya sayısı ve bulunan klon dizilerinin sayısı yer almaktadır. Durum çubuğunun sağında yer alan arama kutusuna bir dosya ismi veya ID numarası yazılarak kaynak kod dosyası aratılabilir.

CCFinder karşılaştırmaya ilişkin kaynak kodlarda yer alan dosya metriklerinin bilgisini de vermektedir (Şekil 3.19). Bu metrik bilgilerine menüden *Metrics -> Show File Metrics* ile ulaşılabilir.



Şekil 3.19: CCFinder dosya metrik bilgilerinin gösterildiği ekran

File Metrics ekranında çıkan metrik bilgileri aşağıda açıklanmıştır (Kamiya, 2008; Ma ve Woo, 2008):

NBR (neighbor): Bir f dosyasındaki kodun bir veya daha fazla bölümünü içeren kaynak kod dosya sayısıdır. Büyük NBR değeri benzer dosyaların çok sayıda olduğu anlamına gelmektedir. NBR değeri kaç dosyanın benzer olduğunu ifade eder (Ma ve Woo, 2007).

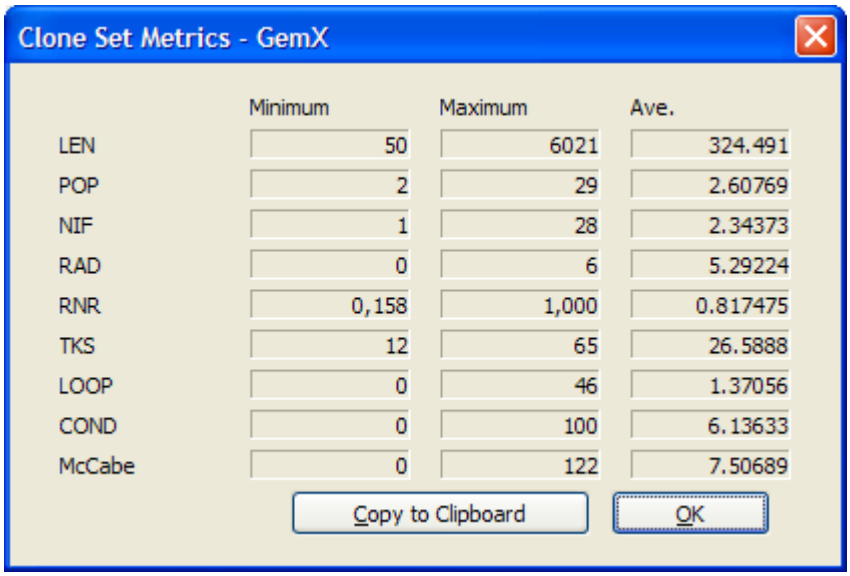
RSA (*ratio of similarity between another files*): Karşılaştırılan iki kaynak kod dosyasının oluşturduğu simgelerin birbirine benzerliğinin yüzde olarak oranıdır. Bu oran ne denli büyükse benzerlik de o oranda çok demektir. Örneğin bu oranın %100 olması dosyalardan birinin diğerinden olduğu gibi kopyalandığı anlamına gelmektedir. RSA değeri benzerliğin nasıl (ne oranda) olduğunu ifade etmektedir (Ma ve Woo, 2007).

RSI (*ratio of similarity within the file*): Bir dosyadaki klon kod simge oranıdır. Bu değer %100'e ne kadar yakınsa o dosyanın benzer fonksiyon veya metotlar içerdiği söylenebilir.

CVR (*coverage*): Herhangi bir kod klonunun kapsadığı simgelerin yüzde olarak oranıdır.

RNR (*ratio of non-repeated code*): Tekrarlanmayan kodun oranını verir. Değeri sıfıra yakın ise, kodda basit ifadeler ve/veya tanımların tekrarlandığı anlamına gelmektedir.

CCFinder klon kodun metrik bilgilerini de vermektedir. Şekil 3.20'de *Clone Set Metrics* ekranında klon metrikleri gösterilmiştir. Burada yer alan önemli metrikler aşağıda açıklanmıştır (Kamiya ve diğ., 2002; Kamiya, 2008).



Şekil 3.20: CCFinder klon metrik bilgilerini gösteren ekran

LEN: Klon kodun içerdiği simge uzunluğudur.

POP: Kodda klonlanmış bölümlerin sayısıdır.

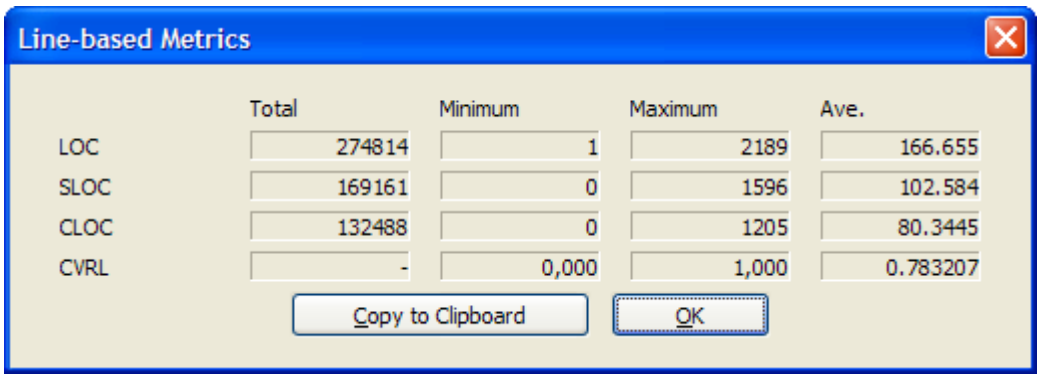
NIF: Bir veya daha fazla klon kod içeren kaynak kod dosyalarının sayısıdır.

RAD: Klasördeki kaynak kodlarda bulunan klon kod oranıdır. Tüm klonlar bir dosyada ise bu değer sıfırdır.

RNR: Klon kod dosyasında tekrarlanmayan kodların simgelerinin oranıdır.

TKS: Klon kod olarak algılanan simgelerin büyüklüğüdür.

CCFinder aynı zamanda klon kodun satır bazında metrik bilgilerini de *Line-based Metrics* ekranında vermektedir (Şekil 3.21).



Şekil 3.21: CCFinder klon kodun satır tabanlı metrik bilgilerini gösteren ekran

Burada gösterilen metrik bilgileri aşağıda açıklanmıştır (Kamiya ve diğ., 2002; Kamiya, 2008):

LOC (*Lines of Code*): İşlenmemiş ham kod satır sayısıdır.

SLOC: Boş satır veya yorum satırları hariç geçerli simge içeren satır sayısıdır.

CLOC: Klon kod bölümünün en az bir simge içeren satır sayısıdır. Klonlanan satır sayısını verir.

CVRL: Klon kod bölümünün simge içeren satırlarının geçerli simge içeren satırlara oranıdır. CVRL = CLOC / SLOC ile hesaplanır.

CCFinder karşılaştırma sonuçlarını kendi okuyabileceği bir formatta kaydedebilmektedir. Ayrıca karşılaştırılan kaynak kod dosyalarının isimlerini de liste olarak kaydedebilmektedir. Bunun yanında karşılaştırma ekranında yer alan Scatter Plot haritalarının PNG formatında kaydedilmesine izin vermektedir.

3.1.1.4. Sherlock

Warwick Üniversitesi Bilgisayar Bilimleri departmanında geliştirilmiş olan Sherlock, kaynak kodlarda simgeleştirme yöntemini kullanarak metinlerde ise string eşleme yöntemini kullanarak benzerlik tespiti yapan açık kaynak bir araçtır (Joy ve Luck, 1999; Goel ve Rao, 2005). http://sydney.edu.au/engineering/it/~scilect/sherlock/ adresinden C kaynak kodu indirilebilen Sherlock, Linux ve Windows işletim sistemlerinde derlenebilmektedir. Sherlock C, C++, Java, Pascal, HTML ve metin dosyaları arasındaki benzerlik oranlarını vermektedir (Goel ve Rao, 2005). Tar uzantılı sıkıştırılmış dosyaları da işleyebilen Sherlock, 2002 yılında BOSS (The BOSS Online Submission System) kurs yönetim yazılımına uyarlanarak öğrenci ödevleri arasındaki benzerliği tespit etmek amacıyla da kullanılmaktadır (Ahtiainen ve diğ., 2006). Sherlock, kaynak kodda kelimelerin dizilimine bakarak hash algoritması ile dijital sayı dizileri üretmekte ve bunları karşılaştırmaktadır (Joy ve Luck, 1999).

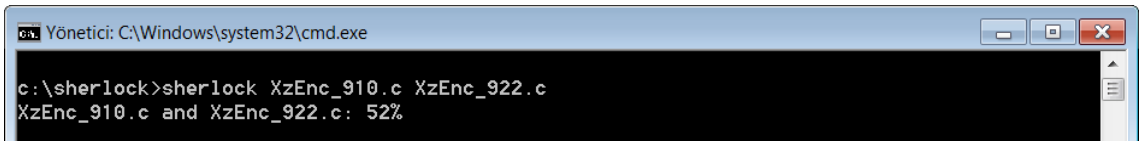
C kodu indirilen Sherlock konsoldan çalışmakta, BOSS yazılımına entegre olan Sherlock ise grafiksel arayüzden kullanılmaktadır. Sherlock hem iki kaynak kod dosyası arasında hem de iki klasörde bulunan dosyalar arasında karşılaştırma yapabilmektedir. Sherlock'u kullanabilmek için indirilen `sherlock.c` C kaynak kodunun derlenmesi gerekmektedir. Derleme sonrası çalıştırılabilir *sherlock* uygulaması oluşmaktadır.

Sherlock temel olarak aşağıdaki yapıda kullanılmaktadır:
sherlock [*–parametre değer*] *karşılaştırılacak_dosya1 karşılaştırılacak_dosya2*

Sherlock programının karşılaştırmayla ilgili alacağı parametreler örneklerle aşağıda açıklanmıştır:

- İki C kaynak kodu arasındaki benzerliği bulmak için `sherlock` komutundan sonra karşılaştırılacak dosyaların arada bir boşluk bırakılarak yazılması gerekmektedir. Şekil 3.22'de de görüleceği gibi Sherlock iki kaynak kod arasındaki benzerlik oranını yüzde olarak vermektedir.

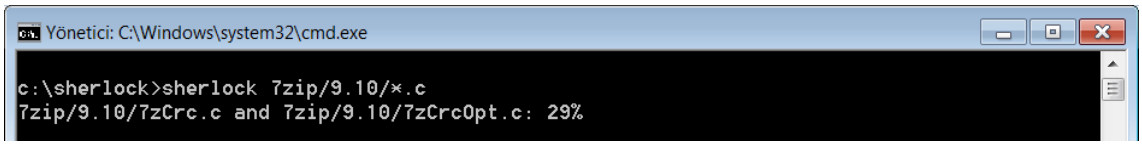
Örnek kullanım:
`sherlock XzEnc_910.c XzEnc_922.c`



Şekil 3.22: İki C kaynak kodunun Sherlock ile karşılaştırılması örneği

- Aynı klasördeki bütün C kaynak kodları arasındaki benzerliği bulmak için ise joker karakter olan * kullanılmalıdır (Şekil 3.23).

Örnek kullanım:
`sherlock 7zip/9.10/*.c`

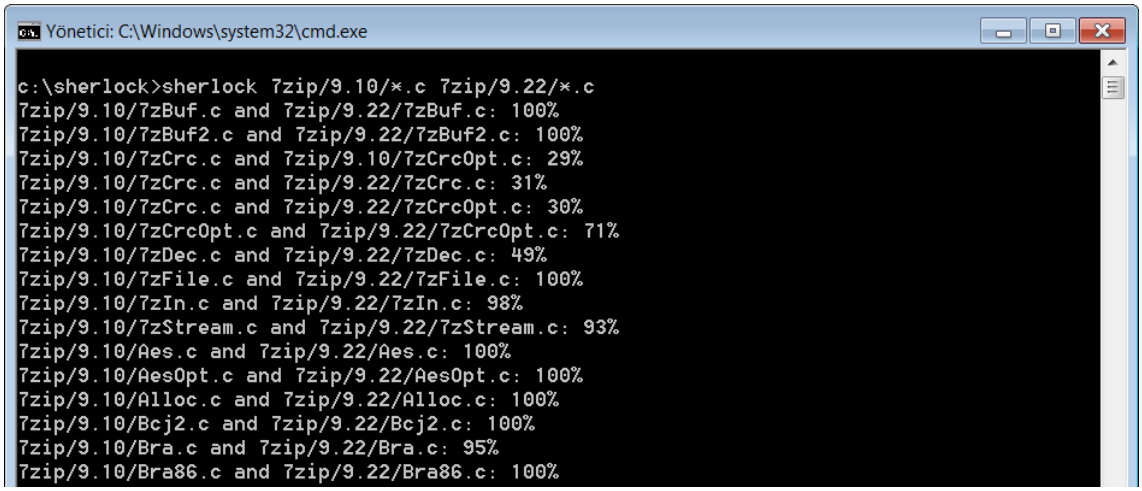


Şekil 3.23: Verilen klasördeki kaynak kodların Sherlock ile karşılaştırılması

- İki klasör arasındaki C kaynak kodlarının benzerlik oranını bulmak için karşılaştırılacak klasörler ve içerisindeki dosyalar yazılmalıdır. Karşılaştırma sonucu Şekil 3.24'te verilmiştir.

Örnek kullanım:

```
sherlock 7zip/9.10/*.c 7zip/9.22/*.c
```

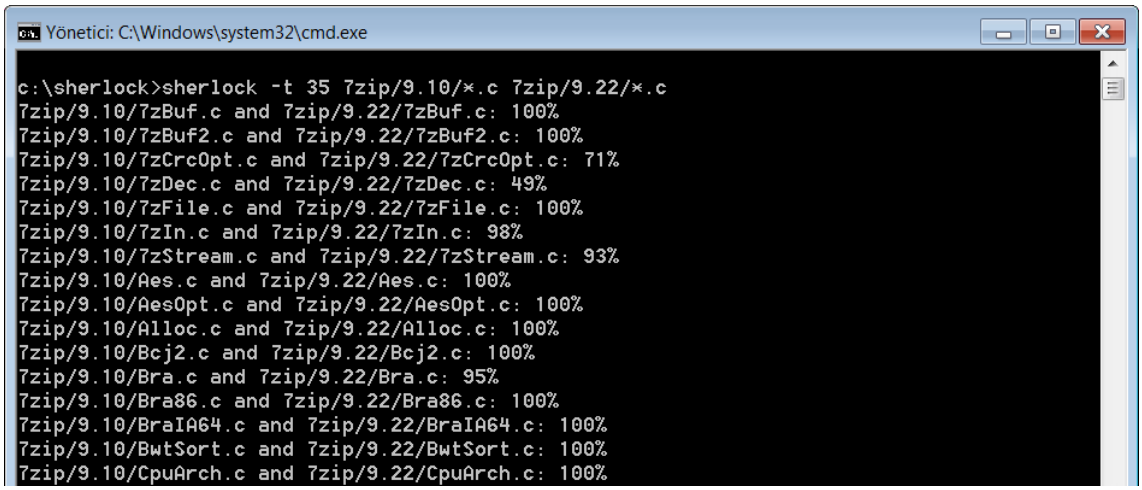


Şekil 3.24: İki klasördeki kaynak kodların Sherlock ile karşılaştırılması

- Belli bir oranın üzerindeki benzerlikleri görüntülemek için –t (*threshold*) parametresi ile oran verilmelidir. Örneğin %35'un üzerinde benzerlik gösteren dosyaları görüntülemek için aşağıdaki örnek kullanımın sonucu Şekil 3.25'te gösterilmiştir. Normal kullanımda iki klasörde bulunan ve %31 benzeşen 7zCrc.c dosyası bu durumda listede gözükmeyecektir.

Örnek kullanım:

```
sherlock -t 35 7zip/9.10/*.c 7zip/9.22/*.c
```

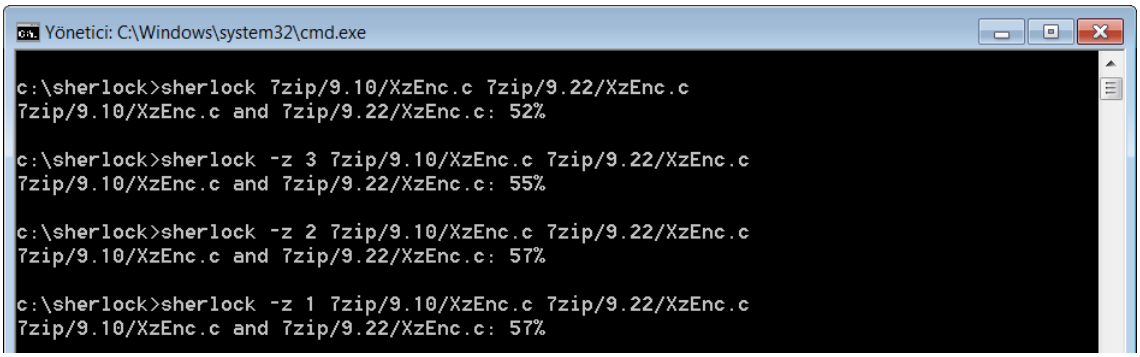


Şekil 3.25: Sherlock'un -t parametresi ile kullanımı

- Daha hassas bir karşılaştırma yapmak için –z (*zerobits*) parametresi ile birlikte dikkate alınacak minimum karakter uzunluğu verilebilir. Bu parametrenin varsayılan değeri 4'tür fakat daha ayrıntılı karşılaştırma için düşürülebilir. Bu parametre kullanılmadan ve parametre değeri değiştirilerek yapılan karşılaştırma sonuçları Şekil 3.26'da gösterilmiştir. Parametre düşürüldükçe daha hassas karşılaştırılma yapılmakta ve benzerlik oranı artmaktadır.

Örnek kullanım:

```
sherlock -z 3 7zip/9.10/XzEnc.c 7zip/9.22/XzEnc.c
```

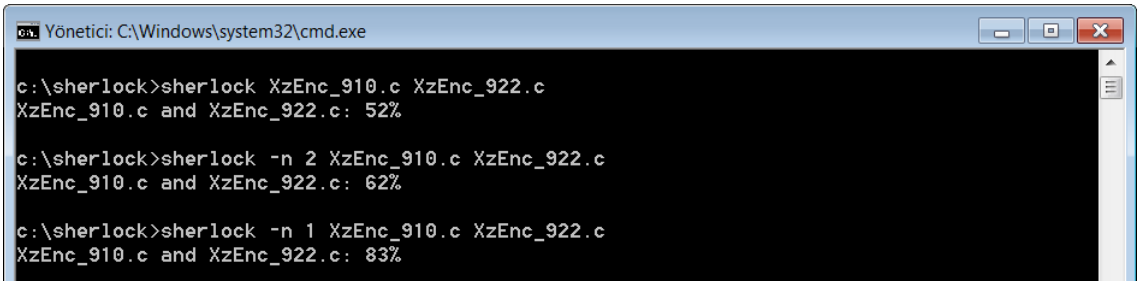


Şekil 3.26: Sherlock'un -z parametresi ile kullanımı

- Simge oluşturmak için kullanılacak olan kelime sayısı –n (*number_of_words*) parametresi ile belirlenebilir. Varsayılan değeri 3 olan bu parametre düşürüldükçe daha hassas karşılaştırılma yapılmakta ve benzerlik oranı artmaktadır (Şekil 3.27).

Örnek kullanım:

```
sherlock -n 3 7zip/9.10/XzEnc.c 7zip/9.22/XzEnc.c
```

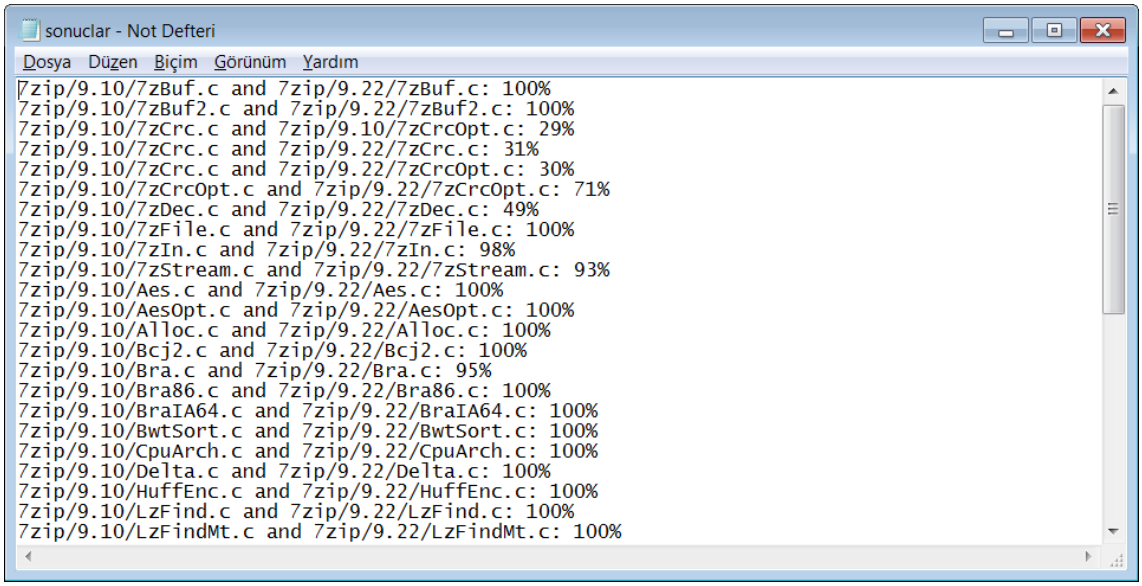


Şekil 3.27: Sherlock'un -n parametresi ile kullanımı

- Karşılaştırma sonuçlarının bir dosyaya aktarılması için –o (*outfile*) parametresi verilerek aktarılacak dosyanın adı yazılmalıdır. Aşağıdaki örnek kullanımın sonucu Şekil 3.28'de gösterilmiştir.

Örnek kullanım:

```
sherlock -o sonuclar.txt 7zip/9.10/*.c 7zip/9.22/*.c
```



Şekil 3.28: Sherlock karşılaştırma sonuçlarının aktarıldığı dosyanın içeriği

Sherlock'un hızlı olması, ek parametreler alması ve karşılaştırma sonuçlarını bir dosyaya aktarabilmesi avantajları arasında sayılabilir. Fakat karşılaştırma sonuçlarını görselleştirmemesi ve konsoldan kullanılıyor olması Sherlock'un eksileri arasındadır.

3.1.1.5. SIM

SIM C, Java, Pascal, Modula-2, Lisp, Miranda ve metin dosyalarında benzerlik tespiti yapan bir araçtır (Grune, 2008). ftp://ftp.cs.vu.nl/pub/dick/similarity_tester/ adresinden indirilebilen ve ücretsiz bir yazılım olan SIM karışık karşılaştırma tekniği kullanan araçlardan biridir. Benzerlik karşılaştırması yaparken metin tabanlı karşılaştırma yöntemi ve ağaç tabanlı yaklaşımı kullanır (Mozgovoy, 2006). Karşılaştırılacak her bir kaynak kod önce sözcüksel analiz yöntemi ile sayılardan oluşan simgelere dönüştürülmekte ve ardından iki kaynak kod dosyasında yer alan kod bloklarının simgeleri karşılaştırılarak benzerlik sonucu üretilmektedir (Arwin ve Tahaghoghi, 2006; Hage ve diğ., 2010). SIM isim değişikliklerini ve program bloklarının yer değişimini fark edebilmekte, boşlukları ise dikkate almamaktadır (Arwin ve Tahaghoghi, 2006).

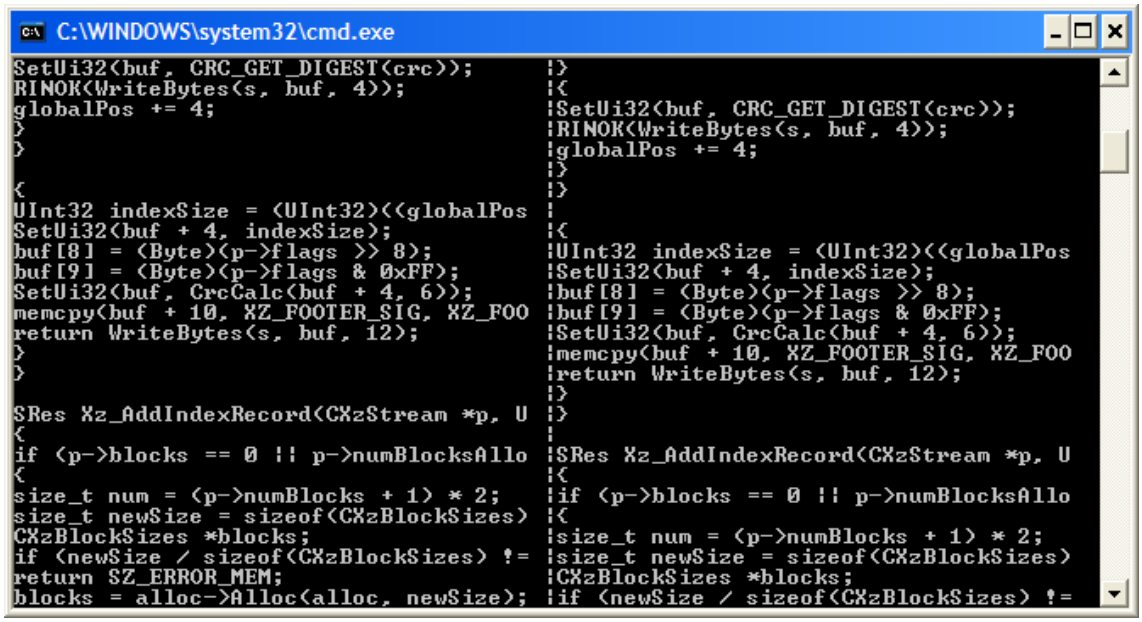
SIM, sadece 32-bit Windows üzerinde konsoldan çalıştırılabilmekte ve grafik arayüz desteği sunmamaktadır. Sonuçların çıktısını ekrana veya bir dosyaya aktarılabilen SIM çalışma sırasında çeşitli parametreler alabilmektedir. Aşağıda SIM uygulamasının kullanımına ilişkin birkaç örnek verilmiştir.

- İki C dosya arasındaki benzerliği bulmak için konsoldan `sim_c dosya1 dosya2` yazmak yeterlidir. SIM aynı zamanda klasör tabanlı karşılaştırma da yapabilmektedir (Şekil 3.29).

Örnek kullanım: `sim_c XzEnc_910.c XzEnc_922.c`

Şekil 3.29: SIM ile iki C kaynak kodu dosyasının karşılaştırılması

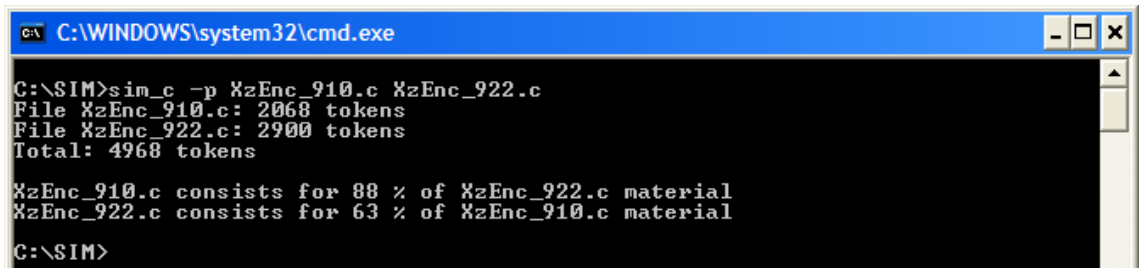
SIM varsayılan olarak benzeşen kod satırlarını ekranda göstermektedir (Şekil 3.30).



Şekil 3.30: SIM ile karşılaştırma sonuçlarının gösterimi

- Karşılaştırılan kodların benzerlik oranını almak için *–p (percentages)* parametresi kullanılmalıdır (Şekil 3.31).

Örnek kullanım: `sim_c –p XzEnc_910.c XzEnc_922.c`



Şekil 3.31: SIM ile karşılaştırılan kodların benzerlik oranının alınması

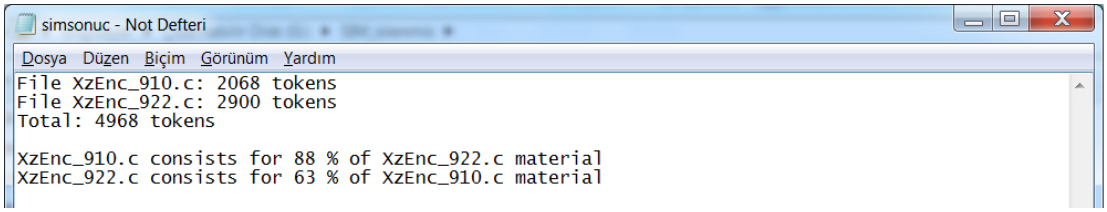
Şekil 3.31'de görüleceği gibi SIM, 7zip programının 9.22 sürümüne ait XzEnc_922.c dosyasının 9.10 sürümüne ait `XzEnc_910.c` dosyasına %63 oranında benzediği, `XzEnc_910.c` dosyasının ise `XzEnc_922.c` dosyasına %88 oranda benzediği hesaplamıştır.

- Karşılaştırmada ekranda çıkan bilgiler ve benzerlik sonuçları istenirse bir metin dosyasına *–o dosya_adı* parametresi verilerek aktarılabilmektedir. Aşağıdaki örnek kullanımda sadece "benzerlik sonuçları" `simsonuc.txt` dosyasına aktarılmaktadır.

Örnek kullanım:

```
sim_c -p -o simsonuc.txt XzEnc_910.c XzEnc_922.c
```

Benzerlik sonuçlarının raporlandığı `simsonuc.txt` dosyasının içeriği Şekil 3.32'de gösterilmiştir.



Şekil 3.32: SIM ile yapılan karşılaştırma sonuçlarının aktarıldığı dosyanın içeriği

SIM kaynak kod benzerlik tespit aracının ücretsiz oluşu ve karşılaştırma sonuçlarını başka bir dosyada raporlayabilmesi avantajlarından biridir. Grafik arayüz desteği sunmaması dezavantajlarından biri olsa da kaynak kod benzerliğinde güvenilir sonuçlar vermektedir.

3.1.1.6. Simian

Java, C, Objective-C, C++, C#, JavaScript (ECMAScript), COBOL, ABAP, Ruby, Lisp, SQL, Visual Basic, Groovy dillerinde tam olarak JSP, ASP, HTML, XML dillerinde de kısmî olarak karşılaştırma yapan Simian (Similarity Analyser) metin tabanlı karşılaştırma yapan kaynak kod benzerlik tespit aracıdır (Harris, 2011; Pate ve diğ., 2011). http://www.harukizaemon.com/simian/get_it_now.html adresinden indirilebilen Simian kişisel kullanım için 15 gün ücretsiz lisans desteği sunmaktadır. Windows, MacOS ve Linux altında konsol tabanlı çalışabilen Simian, benzerlik sonuçlarını bir dosyaya aktarabilmektedir. Simian, karşılaştırılan kodların benzerlik oranlarını vermemekte, sadece karşılaştırılan kaynak kod dosyalarında birbirinden kopyalanmış satırları göstermektedir (Mishne ve De Rijke, 2004). Çalışması için sistemde JRE 1.4

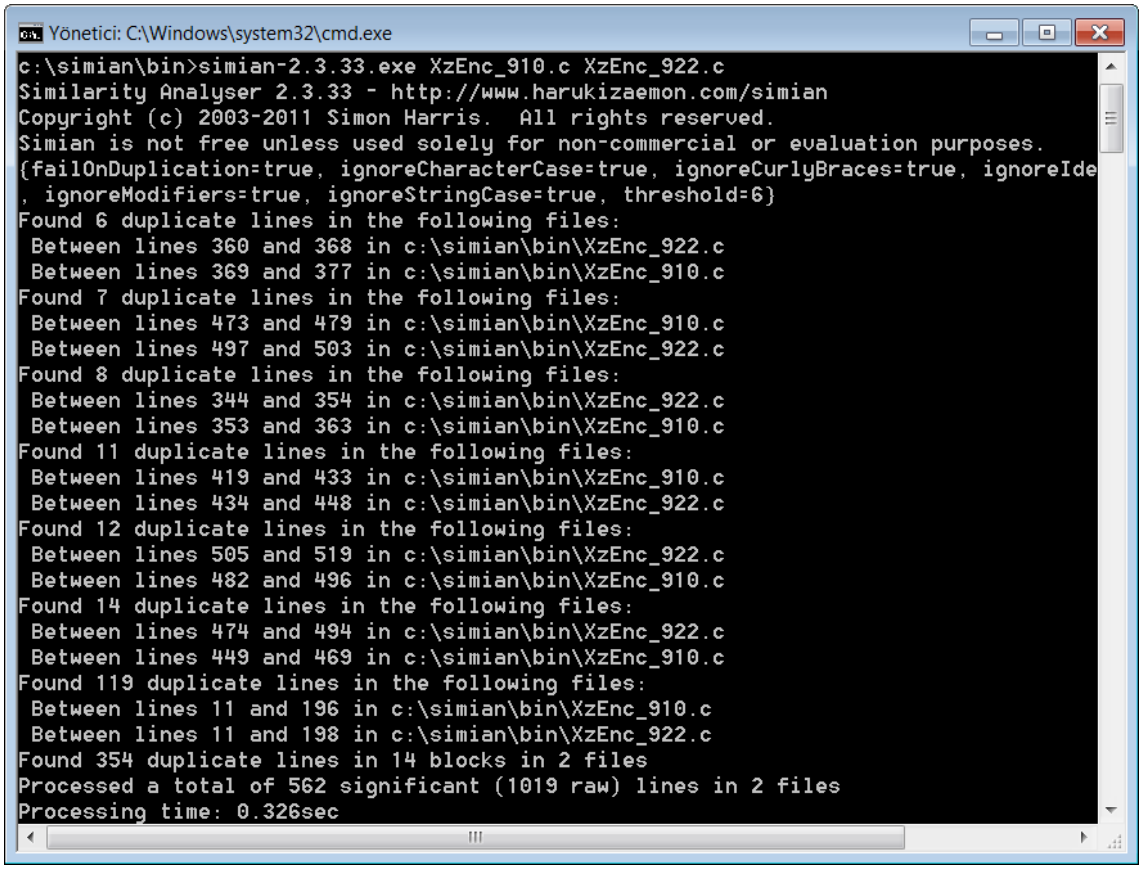
veya üzeri ve .Net 1.1 ve üzeri kurulu olması gereken Simian metin tabanlı karşılaştırma yöntemini kullanmaktadır (Roy ve diğ., 2009). Simian sadece dosya bazlı değil klasör tabanlı karşılaştırma da yapabilmektedir.

Simian ile yapılan örnek karşılaştırmalar ve ekran görüntüleri aşağıda verilmiştir:

- İki C kaynak kod dosyasının karşılaştırılmasına ilişkin örnek Simian kullanımı ve karşılaştırma sonrası oluşan ekran görüntüsü Şekil 3.33'te gösterilmiştir.

Örnek kullanım:

```
simian XzEnc_910.c XzEnc_922.c
```



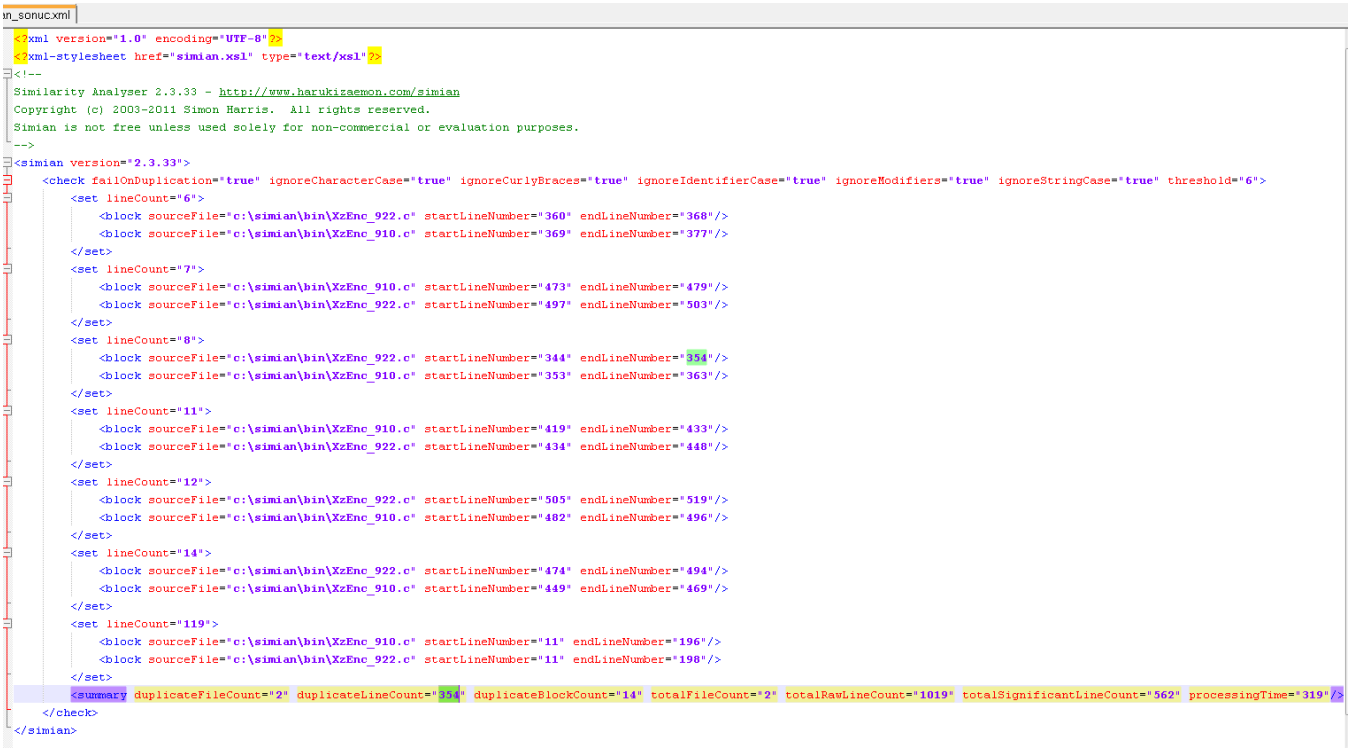
Şekil 3.33: Simian ile iki C kaynak kod dosyasının karşılaştırılması

Şekil 3.33'te görüleceği gibi, Simian iki dosya arasında yaptığı kod karşılaştırmasında toplamda 14 blok içinde 354 kod satırının klonlandığını tespit etmiştir. Simian karşılaştırmada bir benzerlik oranı vermemekte fakat hangi satırların klonlandığını tespit edebilmektedir.

Simian, karşılaştırma sonuçlarını metin dosyası (plain), xml, emacs, vs (visual studio) ve yaml formatında *-formatter* parametresi ile bir dosyaya aktarabilmektedir. Şekil 3.34'de benzerlik sonuçlarının aktarıldığı dosyanın içeriği gösterilmiştir.

Örnek kullanım:

```
simian -formatter=xml:c:\simian\bin\simian_sonuc.xml
XzEnc_910.c XzEnc_922.c
```



Şekil 3.34: Simian karşılaştırma sonuçlarının aktarıldığı XML dosyasının içeriği

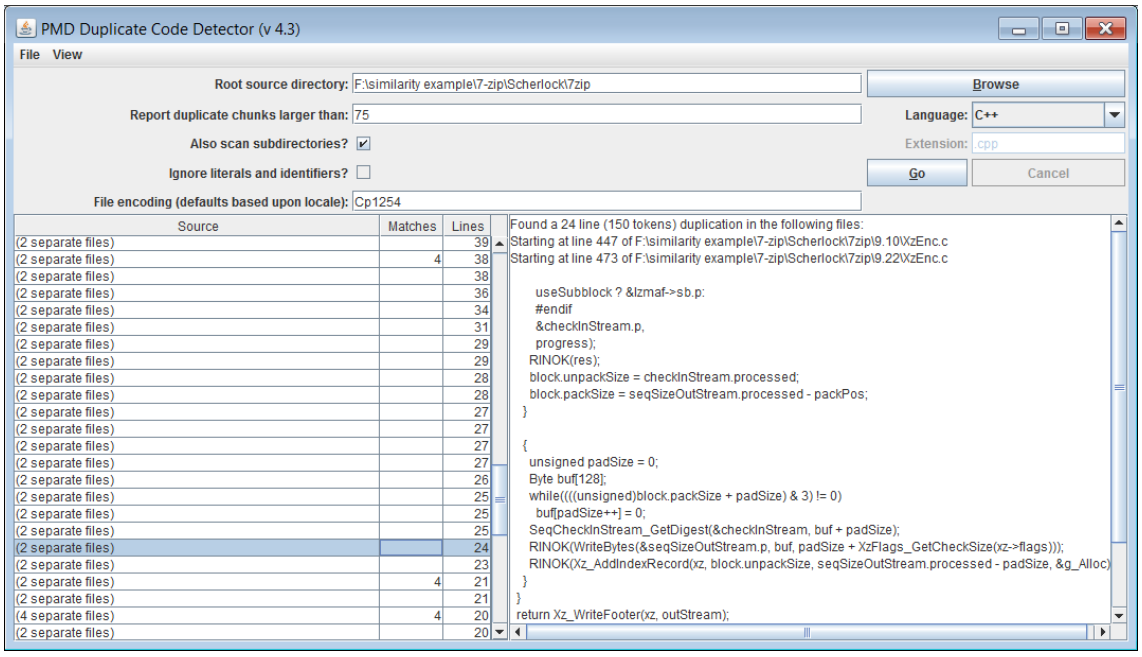
Karşılaştırılan kaynak kodlar arasında benzerlik ölçümü yapmayan Simian, benzerliğinden şüphelenilen koda "kopyala-yapıştır" ile eklenen kod bloklarının yerini ve sayısını vermektedir.

3.1.1.7. CPD

http://pmd.sourceforge.net/cpd.html adresinden indirilebilen veya bilgisayarda JDK 1.4 veya üzeri kurulu ise http://pmd.sourceforge.net/cpd.jnlp adresindeki Java Web Starter uygulamasının indirilmesi ile de çalıştırılabilen PMD projesine bütünleşik CPD (Copy/Paste Dedector), simge tabanlı klon kod tespiti yapabilen ve son versiyonu "*Karp-Rabin"* metin eşleme algoritmasını kullanan açık kaynak bir yazılımdır (Roy ve

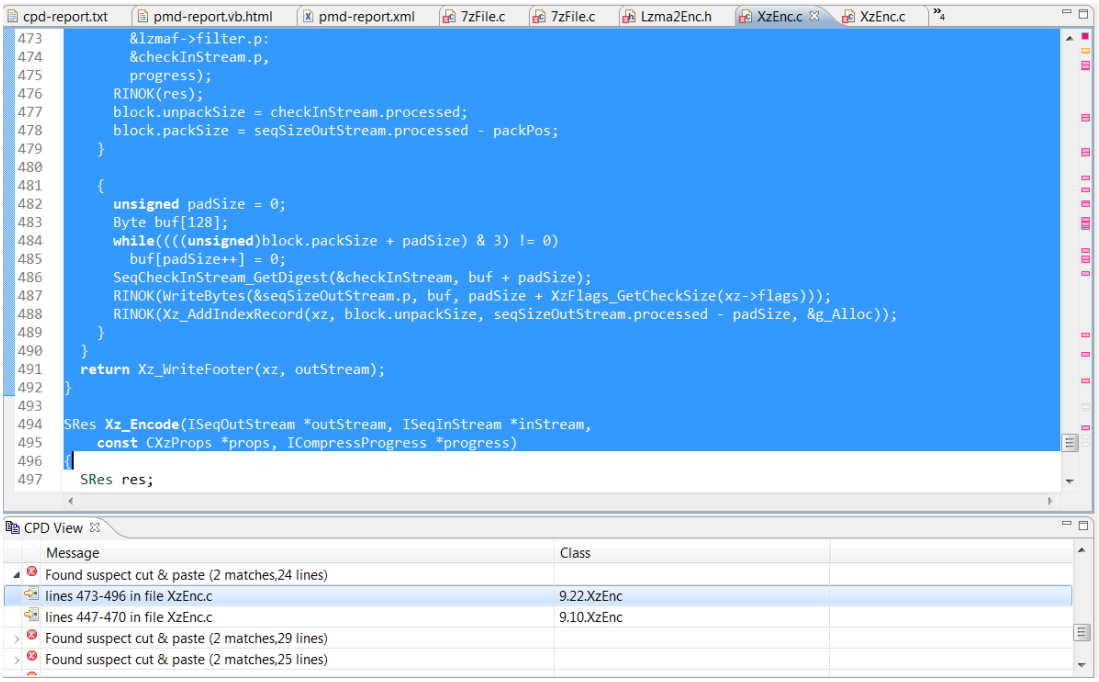
diğ., 2009; InfoEther, 2011). CPD C, C++, C#, Java, JSP, Ruby, Fortran, PHP ve seçilen uzantıya sahip kaynak kod dosyalarında karşılaştırma yapabilmektedir.

Grafik arayüz ve konsoldan kullanılabilen CPD, kaç satırın klonlandığını ve bunların hangi aralıkta olduğu bilgisini vermekte; benzerlik ölçümü yapmamakta sadece kaynak kod dosyalarında birbirinden kopyalanmış kaynak kod bölümlerini göstererek klon kod tespiti yapmaktadır (Mishne ve De Rijke, 2004). BSD lisansı ile yayımlanan CPD aralarında Eclipse, NetBeans, JDeveloper, JEdit, BlueJ ve IntelliJ IDEA gibi birçok IDE'ye eklenti olarak da kurulabilmektedir. Karşılaştırılacak kodların bulunduğu klasörün, kodun dilinin seçilmesinin ardından karşılaştırma sonrası oluşan CPD ekran görüntüsü Şekil 3.35'de verilmiştir.



Şekil 3.35: CPD klon kod tespit aracı

CPD klon kod tespit yazılımının Eclipse IDE'sinde kullanım ekranı ise Şekil 3.36'daki gibidir. Şekilden de görüleceği gibi alt tarafta yer alan *CPD View* sekmesinde listelenen benzerlik içeren dosyalardan birine tıklandığında ilgili dosyada benzeyen satırlar arka planı mavi renkli seçili olarak gösterilmektedir.



Şekil 3.36: Eclipse geliştirme platformunda CPD ile tespit edilen klonların gösterimi

Benzerlik sonucu vermeyen fakat hızlı bir şekilde klonlanmış kod bloklarını tespit edebilen CPD, karşılaştırma sonuçlarını Text, XML ve CSV dosya formatlarına aktarabilmektedir.

3.1.1.8. Duplo

http://duplo.sourceforge.net/ adresinden kaynak kod veya program olarak indirilebilen ve açık kaynak bir yazılım olan Duplo C, C++, C#, Java ve VB.NET dillerinde kopyalanmış kod bloklarını tespit etmektedir. Duplo, kaynak kodlarda benzerlik ölçümü vermemekte sadece kodlarda tekrarlanan kod bloklarını tespit etmektedir. GNU lisansıyla yayımlanan ve geliştirilmesi 2006 yılında durmuş olan Duplo'nun temel kullanımı aşağıdaki gibidir:

```
duplo [OPTIONS] [INTPUT_FILELIST_FILE] [OUTPUT_FILE]
```

Örnek kullanımda OPTIONS ile Duplo'nun alacağı parametreler ve değerleri belirtilmekte, INTPUT_FILELIST_FILE ile karşılaştıracağı kaynak kod dosyalarının yolunun ve isimlerinin yer aldığı dosyalar/klasörler verilmekte, OUTPUT_FILE ile de sonuçların aktarılacağı dosyanın yazılması gerekmektedir.

Duplo'nun aldığı parametrelerden kaynak kodların benzerlik sonucuna etki eden iki parametre bulunmaktadır. Bunlardan biri olan *–ml* parametresi ile kaynak kodlar arasında birbirine benzeyen minimum kaç satırın klon olarak algılanabileceği belirlenir. Varsayılan değeri 4 olan bu parametre düşürülürse daha hassas klon kod tespiti yapılabilir. Bir diğer parametre *–mc* ise klon tespitinde bir satırda dikkate alınacak minimum karakter sayısını belirler. Bu parametrenin varsayılan değeri ise 3'tür.

Duplo, karşılaştırılacak kodları bir dosyadan okumaktadır. Bunun için karşılaştırılacak kaynak kodların isimlerinin ve yolunun bir dosya içerisine eklenmesi gerekmektedir. Örneğin Windows'ta içerisinde bulunulan klasördeki tüm .c uzantılı C kodlarının ismini karsilastirilacaklar.txt isimli bir dosyaya almak için karşılaştırılacak dosyaların olduğu klasöre konsoldan gelip şu komutu yazmak yeterlidir:

Windows:

```
dir /s /b /a-d *.c > karsilastirilacaklar.txt
```

Unix:

```
find -name "*.c" > karsilastirilacaklar.txt
```

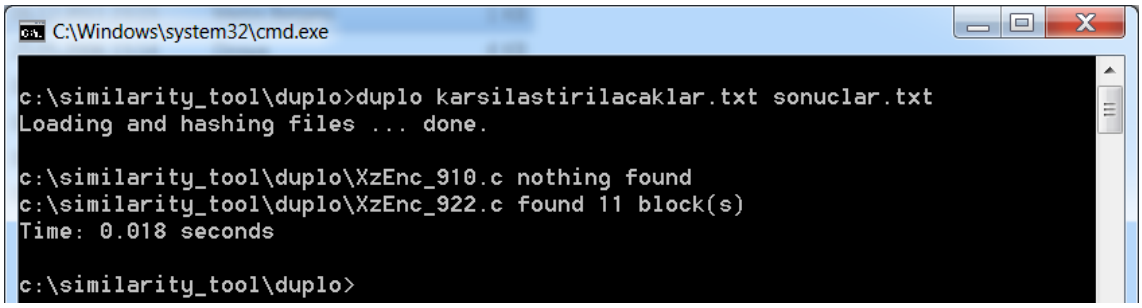
Karşılaştırılacak dosyaların yolu ve isimleri elle veya yukarıda açıklandığı gibi konsol komutlarıyla bir dosyaya aktarıldıktan sonra OUTPUT_FILE parametresi ile sonuçların aktarılacağı dosya ismi verilmelidir.

Duplo'nun kullanımına ait bir örnek aşağıda açıklanmıştır:

- `karsilastirilacaklar.txt` dosyasında isimleri yer alan kaynak kod dosyalarını karşılaştırmak ve karşılaştırma sonuçlarını `sonuclar.txt` isimli dosyaya aktarmak için Duplo'nun örnek kullanımı ve ekran görüntüsü Şekil 3.37'deki gibi olacaktır.

Örnek kullanım:

```
duplo karsilastirilacaklar.txt sonuclar.txt
```



Şekil 3.37: Duplo ile kaynak kod karşılaştırma yapılması

Duplo, `karsilastirilacaklar.txt` dosyada isimleri bulunan `XzEnc_910.c` ve `XzEnc_922.c` kaynak kod dosyalarında benzerlik tespiti yapmış ve `XzEnc_922.c` dosyasında 11 kod bloğunun klonlanmış olduğunu tespit etmiştir. Sonuçların aktarıldığı `sonuclar.txt` dosyasına kopyalanan kod satırları, hangi satırların kopyalandığı, işlenen ve kopyalanan toplam satır sayısı ve karşılaştırma işleminin ne kadar sürdüğü aktarılmıştır. Şekil 3.38'de sonuçların aktarıldığı dosyanın bir kısmı gösterilmiştir.

```
c:\similarity_tool\duplo\XzEnc_922.c(473)
c:\similarity_tool\duplo\XzEnc_910.c(448)
        &checkInStream.p,
        progress);
      RINOK(res);
      block.unpackSize = checkInStream.processed;
      block.packSize = seqSizeOutStream.processed - packPos;
      unsigned padSize = 0;
      Byte buf[128];
      while((((unsigned)block.packSize + padSize) & 3) != 0)
        buf[padSize++] = 0;
      SeqCheckInStream_GetDigest(&checkInStream, buf + padSize);
      RINOK(WriteBytes(&seqSizeOutStream.p, buf, padSize + XzFlags_GetCheckSize(xz->flags)));
      RINOK(Xz_AddIndexRecord(xz, block.unpackSize, seqSizeOutStream.processed - padSize, &g_Alloc));
  return Xz_WriteFooter(xz, outStream);
SRes Xz_Encode(ISeqOutStream *outStream, ISeqInStream *inStream,

Configuration:
  Number of files: 2
  Minimal block size: 4
  Minimal characters in line: 3
  Ignore preprocessor directives: 0
  Ignore same filenames: 0

Results:
  Lines of code: 589
  Duplicate lines of code: 201
  Total 11 duplicate block(s) found.
```

Şekil 3.38: Duplo karşılaştırma sonuçlarının görüntülenmesi

3.1.1.9. Plaggie

Sadece Java kodlarında intihal tespiti yapabilen Plaggie, Aleksi Ahtiainen tarafından geliştirilen GPL lisansına sahip açık kaynak benzerlik tespit yazılımıdır. Öğrencilere verilen programlama ödevlerindeki kopya/klon kodları tespit etmek için geliştirilmiş olan Plaggie, http://www.cs.hut.fi/Software/Plaggie/download.html adresinden indirilebilmektedir (Ahtiainen ve diğ., 2006). JPlag temel alınarak geliştirilen Plaggie, Running-Karp-Rabin Greedy-String-Tiling (RKR-GST) algoritmasını kullanarak simgeleştirme tekniğine dayalı karşılaştırma yapmaktadır (Mozgovoy, 2007; Cosma, 2008).

Plaggie'den türemiş ve derlemeye gereksinim duymadan çalıştırılabilen Plaggie2 isminde bir başka proje daha bulunmaktadır. Tezin bu kısmında Plaggie2'nin kullanımı açıklanmış olmakla birlikte Plaggie2'nin Plaggie'den bir farkı olmadığı, ondan türediği ve aynı kullanıma sahip olduğu için bundan sonra Plaggie2'den Plaggie diye bahsedilecektir.

svn checkout http://plaggie2.googlecode.com/svn/trunk/ plaggie2-read-only ile edinilebilen Plaggie2, konsoldan veya Eclipse gibi bir Java geliştirme ortamında çalıştırılabilmektedir. Temel kullanımı aşağıdaki gibi olan Plaggie, karşılaştırma işlemini kurulduğu bilgisayarda yapmakta ve benzerlik sonuçlarını HTML sayfası şeklinde kullanıcıya sunmaktadır.

Temel kullanımı:

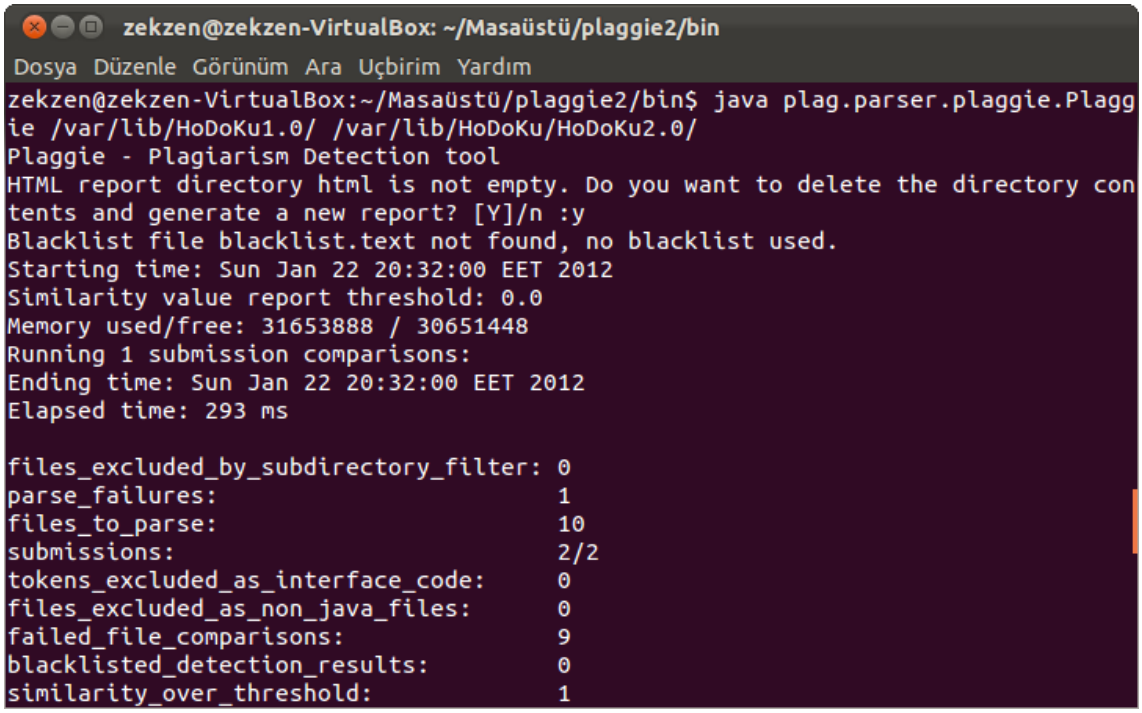
```
java plag.parser.plaggie.Plaggie [-s<number>] [-nohtml]
directory1|file1 [directory2|file2]
```

Plaggie sadece Java kaynak kodları karşılaştırabildiğinden, diğer araçların kullanımını açıklamak için C ve C++ ile kodlanmış 7zip sıkıştırma programı yerine Java ile kodlanmış açık kaynak bir yazılım olan ve http://sourceforge.net/projects/hodoku/ adresinden indirilebilen "Sudoku for Java – HoDoKu" isimli Sudoku oyununun 1.0 ve 2.0 versiyonlarının Java kodları karşılaştırılmıştır.

Konsoldan Plaggie'nin kullanılması için Plaggie'nin kaynak kodlarının olduğu dizinde *bin* klasörüne gelinip karşılaştırılacak klasörlerin belirtilmesi yeterlidir (Şekil 3.39).

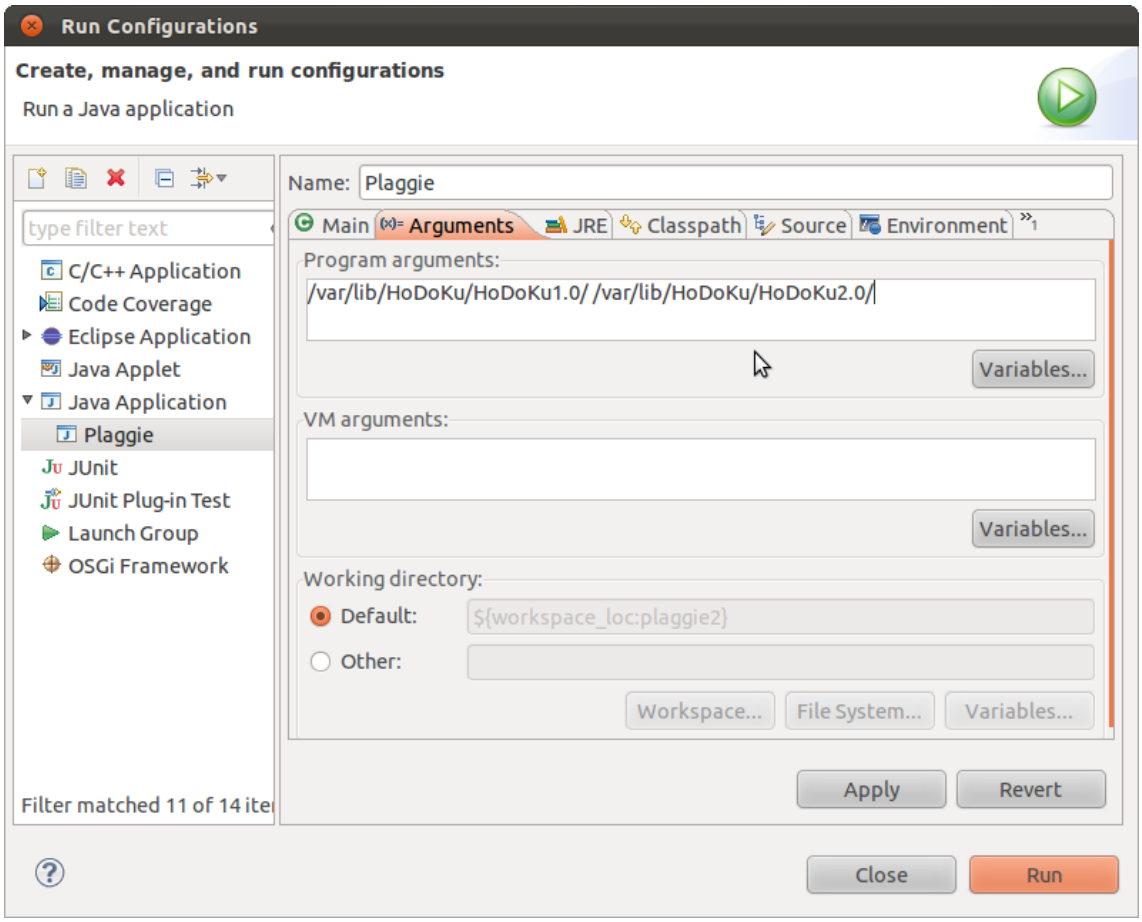
Örnek kullanım:

```
java plag.parser.plaggie.Plaggie /var/lib/HoDoKu/HoDoKu1.0/
/var/lib/HoDoKu/HoDoKu2.0/
```

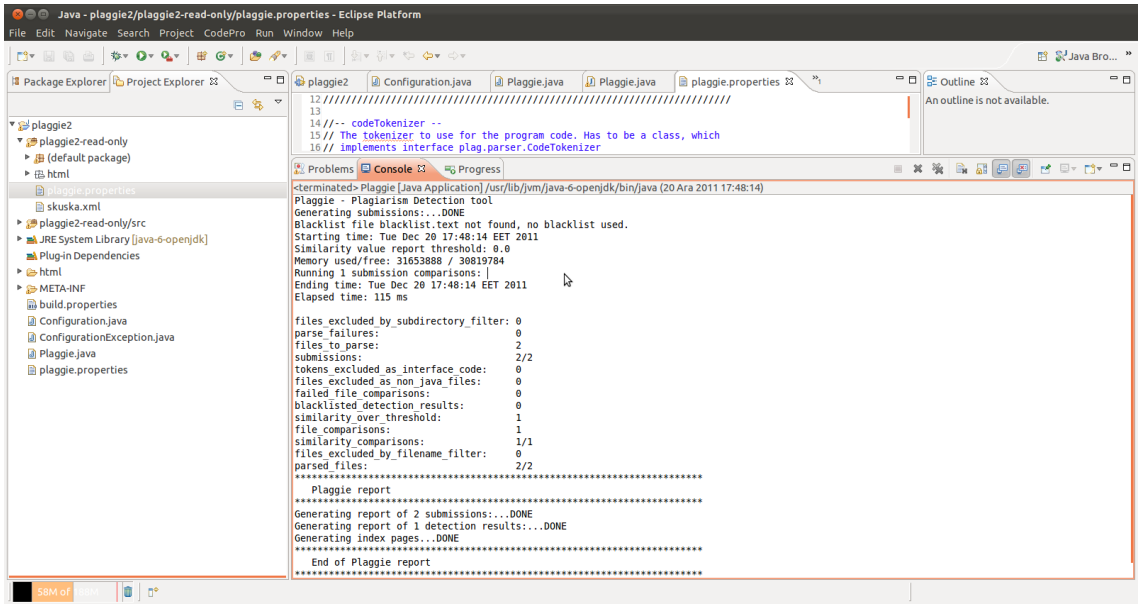


Şekil 3.39: Konsoldan Plaggie'nin kullanılması

Eclipse üzerinde Plaggie'nin kullanılması için yeni bir Java projesi oluşturulması ve Plaggie kodlarının Eclipse'e içe aktarım yapılması (import) gerekmektedir. Aktarılan kodların çalıştırılması için *Run->Run Configurations*... menüsü kullanılarak açılan pencerede yeni bir "Java Application" ayarı oluşturulur. Oluşturulan ayarların *Arguments* sekmesinde ise karşılaştırılacak kodların yolu ve adı verilmelidir (Şekil 3.40). Daha sonra Plaggie çalıştırıldığında Eclipse'in alt tarafında yer alan *Console* sekmesinde Plaggie'nin karşılaştırmaya ilişkin çıktıları yer alacaktır (Şekil 3.41).



Şekil 3.40: Plaggie'nin karşılaştıracağı iki Java dosyasının Eclipse üzerinde parametre olarak verilmesi



Şekil 3.41: Plaggie intihal tespit aracının Eclipse üzerinde çalıştırılması

Plaggie karşılaştırma sonucunu Plaggie uygulamasının bulunduğu klasörde *html* klasörüne aktarmaktadır. Klasörde yer alan *index.html* isimli sayfa tarayıcıda açıldığında karşılaştırmaya ilişkin ayarların, istatistiklerin ve benzerlik oranlarının verildiği Şekil 3.42'deki ekran çıkacaktır.

Plaggie ile HoDoKu oyununun 1.0 ve 2.0 versiyonlarının Sudoku.java isimli Java kodları karşılaştırılmıştır. Şekil 3.42'de istatistik tablosunda karşılaştırılan kodların benzerlik dağılımının yüzdesi ve her bir kodun diğerine göre benzerliğinin yüzdesi verilmektedir. Örnek karşılaştırmada Şekil 3.42'de görüldüğü gibi her iki kodun benzerlik ortalaması %79.7 hesaplanmıştır. HoDoKu 2.0 versiyonuna ait `Sudoku.java` kaynak kodunun 1.0 versiyonuna ait `Sudoku.java` kaynak koduna benzerliği %82.6 iken, 1.0 versiyonuna ait kaynak kodun 2.0 versiyonununkine benzerliği ise %96.5 olarak ölçülmüştür.

Plaggie - Plagiarism Detection Report

[Configuration] [Statistics] [Results]

Configuration

Minimum match length used	5
Recurse subdirectories	true
Html report directory	html
Minimum submission similarity to report	0.0
Maximum number of detection results to report	500
Minimum file similarity to report	0.05
Code tokenizer used	plag.parser.java2.JavaTokenizer
Filename filter used	plag.parser.java.JavaFilenameFilter
Excluded files	ExistingCode1.java,ExistingCode2.java
Excluded subdirectories	webstore,webstore/server,webstore/remote,webstore/servlets
Excluded code in template files	
Interface code excluded	true

Statistics

Distribution of similarities, Average: % 79,7

% 100	0	
% 90	0	
% 80	0	
% 70	1	##
% 60	0	
% 50	0	
% 40	0	
% 30	0	
% 20	0	
% 10	0	
% 0	0	

Distribution of similarities A, Average: % 82,6

% 100	0	
% 90	0	
% 80	1	##
% 70	0	
% 60	0	
% 50	0	
% 40	0	
% 30	0	
% 20	0	
% 10	0	
% 0	0	

Distribution of similarities B, Average: % 96,5

% 100	0	
% 90	1	##
% 80	0	
% 70	0	
% 60	0	
% 50	0	
% 40	0	
% 30	0	
% 20	0	
% 10	0	
% 0	0	

Distribution of maximum file similarities, Average: % 79,7

If there are a lot of high similarity values (i.e. > 20%) in this distribution, there is a reason to suspect, that some code provided to the students with the exercise definition or somewhere else is not excluded from the detection. You can use the code exclusion parameters in the configuration file to exclude code from the detection algorithm.

% 100	0	
% 90	0	
% 80	0	
% 70	1	##
% 60	0	
% 50	0	
% 40	0	
% 30	0	
% 20	0	
% 10	0	
% 0	0	

Files per submission

Exception while printing statistics: java.lang.NegativeArraySizeException at plag.parser.Stats.printHtmlDistribution(Stats.java:218) at plag.parser.plaggie.Plaggie$1.printHtmlReport(Plaggie.java:451) at plag.parser.report.SimpleHtmlSubmissionReportGenerator.generateIndexPage(SimpleHtmlSubmissionReportGenerator.java:736) at plag.parser.report.SimpleHtmlSubmissionReportGenerator.generateReport(SimpleHtmlSubmissionReportGenerator.java:1050) at plag.parser.plaggie.Plaggie.generateReport(Plaggie.java:467) at plag.parser.plaggie.Plaggie.main(Plaggie.java:725)

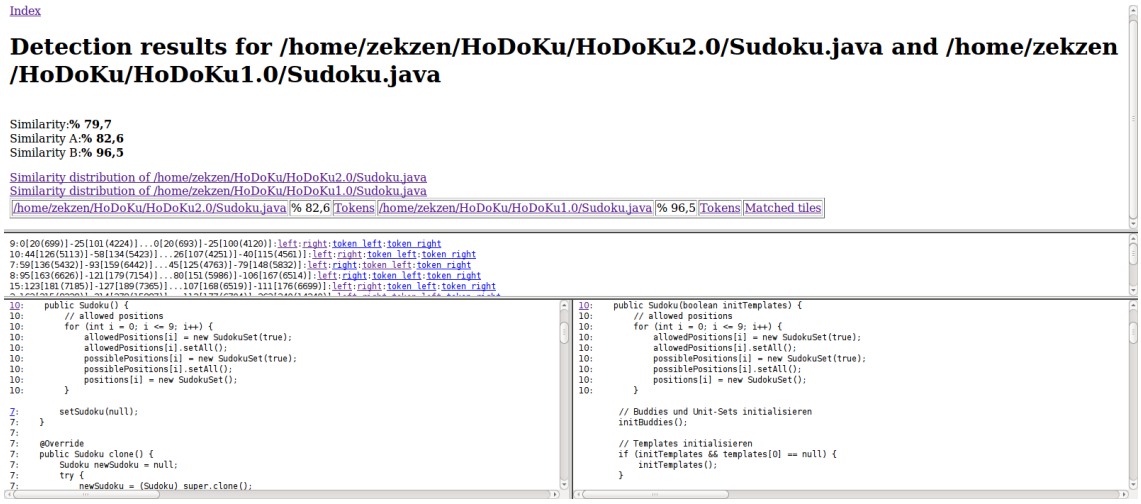
Results

Change sorting by clicking the links on the header row.
Blacklisted student id's are highlighted.

Student A	Student B	Similarity	Similarity A	Similarity B	Maximum file similarity	Student A similarity distribution average	Student B similarity distribution average
/home/zekzen/HoDoKu/HoDoKu2.0/Sudoku.java	/home/zekzen/HoDoKu/HoDoKu1.0/Sudoku.java	% 79,7	% 82,6	% 96,5	% 79,7	% 82,6	% 96,5

Şekil 3.42: Plaggie karşılaştırma sonuçlarının gösterildiği anasayfa

Karşılaştırma sonuçlarının gösterildiği tabloda yer alan kaynak kodların linkine tıklandığında ise her iki kaynak kod arasındaki benzerlik olan kısımlar MOSS ve JPlag benzerlik tespit araçlarında olduğu gibi yan yana iki çerçeve içinde gösterilmektedir (Şekil 3.43). Orta çerçevede yer alan benzerlik tespit edilen kod bloklarının numaralandırıldığı listede *left* ve *right* linklerine tıklandığında benzeşen kod blokları alt çerçevede yan yana gelecektir.



Şekil 3.43: Plaggie benzerlik aracının benzerlik tespit edilen kod bloklarını gösterdiği ekran

Plaggie sadece Java kaynak kodlarında benzerlik tespiti yapabilen ve başarılı sonuçlar verebilen bir intihal tespit aracıdır. Sonuçları kullanıcının anlayacağı şekilde sunması, açık kaynak olması ve kolay kullanımı artıları olarak sayılabilir.

3.1.1.10. Plague

Plague yapısal tabanlı klon tespit araçlarının ilklerindendir. Çalışması birkaç adımda gerçekleşir. İlk adımda her bir kodun yapısal profili çıkartılır ve daha sonra her bir profil *Heckel algoritması* kullanılarak karşılaştırılır. Plague benzerlik sonuçlarını bir liste formunda döndürmekte ve C kaynak kodlarındaki intihali tespit edebilmektedir (Kustanto ve Liem, 2009). Plague'yi edinme imkanımız olmadığından tezde ayrıntılı olarak incelenememiştir.

3.1.1.11. YAP

YAP, Plague tabanlı bir sistem olup gelişmiş özelliklere sahiptir. İlk versiyonu 1992 yılında çıkarılan YAP'ın YAP2 ve YAP3 şeklinde versiyonları da bulunmaktadır. YAP'ın tüm versiyonları iki aşamalı olarak çalışır. İlk aşamada her bir kaynak kodun

simgesel dosyası yaratılır. İkinci aşamada ise her bir simge dosyasının karşılaştırılması yapılır. Karşılaştırma sonuçları 0-100 arasında bir değerle skorlanır. YAP benzerlik tespit aracında minimum benzerlik değerini kullanıcı isterse kendisi belirleyebilmektedir. Her bir simge dosyası çiftinin benzerlik skoru, belirlenen minimum benzerlik değerinin üzerinde çıkarsa ilgili çift intihal yapıldığından şüpheli bir durum olarak değerlendirilmektedir. YAP benzerlik sonuçlarını bir metin dosyasında sunmaktadır (Kustanto ve Liem, 2009).

YAP'ın üç versiyonu arasındaki en önemli fark karşılaştırma aşamasında kullanılan algoritmada yatmaktadır. YAP1 *LCS* algoritması kullanmaktadır. Bu algoritmada kod yapılarının sırasının değiştirilmesiyle intihal gizlenebilmekte ve YAP1 bu tarz tekniklerle yapılan intihali tespit edememektedir. Bu sorunu çözmek için *Heckel* algoritması kullanan YAP2 geliştirilmiştir. Ancak, bu algoritma da uzun altdizgelere (karakter dizilerine) öncelik vermemektedir. YAP3 ise *Running Karp-Rabin Greedy String Tiling* algoritması kullanmaktadır (Kustanto ve Liem, 2009).

Yap benzerlik tespit aracı, yeterli kurulum ve kullanma dokümanına sahip olmadığı için derlenebilmiş fakat karşılaştırmada hata vermesinden dolayı kullanılamamıştır.

3.1.1.12. CodeMatch

CodeMatch kaynak kod benzerliğini karşılaştıran ticarî bir yazılımdır. CodeMatch BASIC, C, C++, C#, Delphi, Flash ActionScript, Java, JavaScript, MASM, Pascal, Perl, PHP, PowerBuilder, Ruby, SQL, Verilog ve VHDL gibi programlama dillerinde benzerlik tespiti yapabilmektedir (ICS, 2011).

3.1.2. Kaynak Kod Benzerlik Tespitinde Kullanılan Araçların Karşılaştırılması

Klon tespit araçlarının başarımı kullandığı algoritmayla iç içedir. Algoritmanın zayıflığı benzerlik sonuçlarını da etkilemektedir. Yazılımların başarımını etkileyen faktörlerin ve tercih edilme sebepleri olarak benzerlik olan kısımları başarılı ve doğru bir şekilde bulabilme, araçların kullanım kolaylığı, doküman ve servis desteği, benzerlik olan kısımların araştırıcının anlayacağı şekilde görselleştirilmesi, karşılaştırma yapabildiği dil sayısı ve araçların kullanım ücretleri sayılabilir (Freire ve diğ., 2007).

- **Kullanılabilirlik**

Kimi sistem sadece ikilik formatta, kimisi ise kaynak kodu şeklinde kimi de her ikisi ile birlikte dağıtılmaktadır. Kaynak koduyla verilen araçlar tercih edilmekle birlikle sistemde derleme sırasında problemler yaşanabilmekte ve benzerlik tespit yazılımı kullanılamamaktadır. Program olarak sunulan araçlar ise kaynak kod olmadığından daha etkili bir kullanım sağlama amacıyla değiştirilememektedir.

- **Kurulum**

Sistemler yerleşik-yerel (*lokal*) veya uzaktan kullanım (*remote*) tipinde olabilir. Uzaktan erişimli sistemler Web tabanlıdır ve karşılaştırılacak dosyaların sunucuya yüklenmesi ile çalışır. Yerleşik sistemler ise çalışmak için özel gereksinimlere ihtiyaç duyabilir. Örneğin CCFinder Python 2.5x ve üzeri Python derleyicisine, Sherlock C derleyicisine ihtiyaç duymaktadır.

- **Dokümantasyon ve destek**

Teknik dokümanın varlığı ve destek seviyesi bu tarz sistemler için önemlidir. Sistemi işletecek olan programcı veya mevcut kullanıcı kullanım sırasında problem yaşayabilir ve desteğe ihtiyaç duyabilir. Teknik dokümantasyon kullanıcıya verimlilik ve etkili kullanım sağlar. Benzerlik tespit aracının kurulumunun ve kullanımını ayrıntılarıyla ve örneklerle açıklanması programın başarımı üzerine doğrudan etkilidir. Kullanıcı, kullandığı intihal tespit aracının dokümanını yazılımla birlikte edinebilmelidir. Fakat bu kısımda incelenen çoğu intihal ve klon kod tespit yazılımı doküman ve teknik destek sunmamakta veya çok az sunmaktadır. Genellikle araçların kurulumu ve kullanımı tek sayfalık metin belgesi ile açıklanmakta, ayrıntılı kullanıma ilişkin doküman ihtiyacı akademik makaleler ile sağlanmaktadır.

- **Benzerlik sonuçlarını görselleştirme**

Benzerlik sonuçlarının grafiksel arayüzde kullanıcının kavrayabileceği şekilde gösterilmesi yazılımların kullanışlılığını artıracaktır. Sherlock, SIM, Simian gibi araçlar konsol tabanlı veya bir metin dosyasında sonuçları görüntülemekte iken, MOSS, JPlag, ve Plaggie intihal tespit araçları sonuçları HTML sayfasında ve benzerlik gösteren kod satırlarını yan yana bölmelerde ve farklı renklendirmelerle kullanıcıya sunmaktadır.

CCFinder benzerlik tespit aracı ise kod benzerliğini *saçılma grafiği* üzerinde kaynak kodları noktalarla ve birbirlerine olan simetrisine göre göstermektedir.

- **Benzerlik oranı sunma**

Benzerlik tespit araçlarından MOSS ve JPlag gibi araçlar karşılaştırmada benzerlik ölçümü yapmakta, karşılaştırılan kodların birbirlerine benzerliğini yüzde olarak vermekte ve benzeyen/klonlanan satırları göstermektedir. Duplo ve CPD gibi araçlar ise benzerlik ölçümü yapmayıp sadece klonlanan satırları göstermektedir.

- **Kullanım kolaylığı**

Benzerlik/klon kod tespit araçlarının kullanım kolaylığı da başarımı etkileyen faktörler arasındadır. Özellikle konsoldan kullanılan araçların alacağı parametreler ve bunların ne şekilde yazılacağı kullanımı zorlaştırıcı faktörlerden biridir. Daha önce değinildiği gibi örnek kullanıma ilişkin eksik doküman araçların kullanımını güçleştirmektedir. Öte yandan konsoldan değil de grafiksel arabirimle kullanılan JPlag ve CCFinder araçları son kullanıcı açısından çok daha kullanışlı bulunmaktadır.

- **Algoritma ve yöntem**

Klon tespit araçlarının kullandığı algoritma ve yöntem, araçların başarılı benzerlik sonuçları üretmesiyle doğrudan ilgilidir. Araçlarda genel olarak simgeleştirme yöntemi kullanılmakla birlikte karışık yöntemler ve algoritmaların kullanılan sisteme göre uyarlanması klon tespit araçlarının çok daha başarılı sonuçlar vermesini sağlamaktadır.
Çoğu araç birden fazla programlama dilinde kaynak kod benzerliği tespiti yapabilmektedir. Bununla birlikte bazı programlama dilleri için belli algoritmaların kullanılması gerekebilir (Freire ve diğ., 2007). Her dil için tekdüze yöntemi kullanmak yerine o dilin gramerini dikkate alarak geliştirilen MOSS ve JPlag gibi araçların benzerlik ölçümü çok daha güvenilir olacaktır.

Tablo 3.2'de tezde incelenen klon kod/benzerlik tespit araçları çeşitli açılardan karşılaştırılmıştır.

Tablo 3.2: Kaynak kod benzerlik tespit yazılımlarının karşılaştırılması
(Goel ve Rao, 2005; Mozgovoy, 2007; Ji ve diğ., 2007; Roy ve Cordy, 2007; Hage ve diğ., 2010).

	MOSS	**JPlag**	**CCFinder**	**Sherlock**
Geliştirilmeye başlama tarihi	1994	1996	1999	1994
Karşılaştırma yapabildiği diller	C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS Assembly 8086, HCL2	Java, C#, C, C++, Scheme ve doğal dil metinleri	Java, C/C++, C#, Visual Basic (VB) ve COBOL	C, Java, Pascal, HTML ve metin dosyaları
Karşılaştırma tekniği	Fingerprint	Simgeleştirme	Simgeleştirme	Simgeleştirme/String Eşleme
Kullandığı algoritma	Winnowing	Greedy String Tiling	Suffix-tree	Greedy String Tiling
Ücret	Ücretsiz	Ücretsiz	Ücretsiz	Ücretsiz
Üyelik	Kullanıcı adı	Kullanıcı adı ve e-mail	Hayır	Hayır
Açık kaynak	Hayır	Hayır	Evet	Evet
Çalıştığı işletim sistemi	32-64 bit Windows, Linux	32-64 bit Windows, Linux	32 bit Windows, Linux	32-64 bit Windows, Linux
Kullanım şekli	Web servisi	Web servisi	Lokal	Lokal
Kullanım arayüzü	Konsol	Grafiksel arayüz	Grafiksel arayüz ve konsol	Konsol
Gereksinimler	Güncel Perl betiği / Perl Derleyicisi	Başlatıcı Java uygulaması / Web tarayıcı, JRE 1.5 veya üstü	Program veya kaynak kod / Python 2.6 Derleyici ve Silverlight Runtime	Kaynak kod / C derleyicisi
Karşılaştırılacak kodu seçim şekli	Dosya, klasör, alt klasörler	Dosya, klasör, alt klasörler	Dosya, klasör, alt klasörler	Dosya, klasör, alt klasörler ve .tar uzantılı dosyalar

	SIM	Simian	CPD	Duplo	Plaggie
Geliştirilmeye başlama tarihi	1989	1997	2003	1994	2006
Karşılaştırma yapabildiği diller	C, Java, Pascal, Modula-2, Lisp, Miranda ve metin dosyalarında	Java, C, Objective-C, C++, C#, JavaScript (ECMAScript), COBOL, ABAP, Ruby, Lisp, SQL, Visual Basic, Groovy, JSP, ASP, HTML, XML	C, C++, C#, Java, JSP, Ruby, Fortran, PHP ve seçilen uzantıya sahip kaynak kod dosyalarında	C, C++, C#, Java ve VB.NET	Java
Karşılaştırma tekniği	Metin tabanlı	Metin tabanlı	Simgeleştirme	Belirtilmemiş	Simgeleştirme
Kullandığı algoritma	String alignment	Belirtilmemiş	Karp-Rabin	Belirtilmemiş	RKR-GST
Ücret	Ücretsiz	15 günlük kişisel kullanım ücretsiz. Ücretli lisansları da var.	Ücretsiz	Ücretsiz	Ücretsiz
Üyelik	Hayır	Lisansına göre	Hayır	Hayır	Hayır
Açık kaynak	Evet	Lisansına göre	Evet	Evet	Evet
Çalıştığı işletim sistemi	32 bit Windows	Windows, Linux	Windows, Linux	Windows, Linux	Windows, Linux
Kullanım şekli	Lokal	Lokal	Lokal	Lokal	Lokal
Kullanım arayüzü	Konsol	Konsol	Grafiksel arayüz ve konsol	Konsol	Konsol veya Eclipse üzerinde
Gereksinimler	Kaynak kod veya program	.Net1.1 veya üzeri ve JRE 1.4 veya üzeri	JDK 1.4 veya üzeri	Kaynak kod / C derleyicisi	Kaynak kod, JRE 1.3.1 veya üzeri
Karşılaştırılacak kodu seçim şekli	Dosya, klasör, alt klasörler	Dosya, klasör, alt klasörler	Klasör, alt klasörler	Karşılaştırılacak kodların adını metin dosyasından okumakta	Dosya, klasör, alt klasörler

3.2. KAYNAK KOD İNTİHALİ DAVALARI

Kaynak kodların intihal edildiği gerekçesiyle mahkemelere taşınan iki dava aşağıda incelenmiştir. Davalarda bilirkişilerin kaynak kod intihalini nasıl tespit ettikleri ve kullandıkları yazılımların ne olduğu araştırılmıştır.

3.2.1. Örnek Dava 1

Kaynak kodların elde dilerek bir başka program üretildiği iddiası açılan bu davada teknik bilirkişi raporları üzerinde durulmuş, mahkemenin sonucu ve hukukî incelemeler çalışmanın kapsamı dışında değerlendirildiği için alınmamıştır (Yasaman, 2006:167-178).

- **Teknik Bilirkişi İncelemesi**

Mahkeme: TC. İstanbul 1. Fikri ve Sınai Haklar Hukuk Mahkemesi
Dosya No: 2003/896
Davacı: A Yazılım Firması
Davalı: B Kurye Firması

Dava geçmişi: Davacı A yazılım firması 1999 yılında davalı B kurye şirketine bir kurye programı hazırlamıştır. Davacı, davalı kurye firmasının bu yazılımı kendilerinin izni olmaksızın "kendine aitmişçesine" CD, disket gibi ortamlarda çoğaltarak kullandığını iddia etmektedir. Ayrıca davacı davalı tarafın bu programı tersine mühendislik yöntemleri ile çözüp programda kısmi değişiklikler yaptığını ve bunu da diğer şubelerine dağıttığını ve bu durumun 3 yıl sürdüğünü iddia etmiştir.

Bilirkişilere verilen görev: Mahkemenin gg.aa.yyyy tarihli ara kararı ile her iki tarafın iddiaları dikkate alınarak dava konusu programların karşılaştırılması yapılarak davalı tarafın programının davacıya ait programda kullanılıp kullanılmadığı, programın ilk kullanım durumu ve davalılar tarafından yapıldığı iddia olunan ilavelerin karşılaştırılarak rapor hazırlanması görevi tevdi edilmiştir.

Teknik İnceleme:

1. Davaya konu olan bilgisayar programının hem davacı tarafta (A yazılım firması) bulunan kaynak kodları, hem de davalı tarafta (B kurye firması) bulunan kaynak kodları CD'lere konularak ayrı ayrı alındı ve inceleme yapılacak bilgisayara yüklendi.
2. Ayrıca programın yapısı ile işleyişini gösteren ve davalı taraf tarafından hazırlanmış olan çeşitli dokümanlar alındı.
3. Kaynak kodlar ve dokümanlar bilirkişi tarafından incelendi.

Yöntemin tarifi:

1. Her bilgisayar programı önce programcı (veya bir grup programcı) tarafından belirli bir programlama diliyle yazılır. Bildiğimiz yazı (metin) şeklinde olan bu yazıların tümü programın **kaynak kodu** olarak adlandırılır. Bir bilgisayar programının bilgisayarda çalışabilmesi için, bu kaynak kodunun bilgisayarın anlayacağı bir dile dönüştürülmesi gerekir. Bu işlem ile elde edilen ürün de nesne kodu olarak adlandırılır.
2. Bir programın yazıldığı programlama dilini bilen herkes, eğer o programın kaynak kodlarına ulaşabilirse, bu kodu kolaylıkla değiştirebilir – programa istediği özellikleri ekleyebilir, istemediği özellikleri çıkarabilir vs.. bir kişinin elinde sadece nesne kodu varsa ve kaynak kodu yoksa, nesne kodunu kaynak koda dönüştürmek son derece zordur. Elde kaynak kodu olmadan bir program pratikte değiştirilemez. Bu nedenle programın kaynak kodları, genellikle programı yazanlar tarafından gizli tutulur, nesne kodları ise programı kullanan herkese dağıtılır.
3. Bir programın başka bir programdan kopyalanıp kopyalanmadığını anlamak genellikle kolaydır. İki programın kaynak kodları karşılaştırılır. Belirli bir dereceden fazla benzerlik varsa, taraflar birinin diğerinden, veya ikisinin birden üçüncü bir kaynaktan (kendileri yazmayıp) yazılmış programı kopyaladığı hükmüne varılabilir. Bununla beraber ortada kopyalama durumu varsa, programların kaynak kodlarına ve bilgisayar ortamında mevcut diğer verilere bakarak, programı kimin kimden kopyaladığını anlamak genellikle zordur.

Yapılan tespitler:

1. Hem davacı hem de davalı taraf, Microsoft FoxPro programlama diliyle yazılmış bir kurye takip programı kullanmaktadır.
2. Her iki programın kaynak kodları hem ayrı ayrı incelenmiş, hem de birbirleriyle karşılaştırılmıştır. Her iki program da yaklaşık olarak 350 dosyadan (program kodu dosyaları, veri dosyaları, ekran dosyaları, vs.) oluşmaktadır. İki programdaki aynı işlevi gören dosyaların hemen hemen hepsinin isimleri aynıdır ve ayrıca bu dosyaların hemen hemen hepsinin içerikleri de aynıdır. Bu bir tesadüf olamaz. Diğer bir deyişle, bu iki programın birbirlerinden bağımsız olarak geliştirilmiş olması imkansızdır. İki programdan birinin diğerinden, ya da ikisinin birden üçüncü bir kaynaktan kopyalanmış olduğu kesindir.
3. Davacı tarafın programındaki dosyalar içinde release.scx isimli bir dosya mevcuttur. Bu dosya çalıştığında, ekranda "Program Adı 1.00. Yazılım sahibi A yazılım firması 1999. Tüm hakları saklıdır." ifadelerini içeren bir çerçeve görünmektedir. Bu bir künye dosyasıdır ve programın kimin tarafından geliştirildiğini göstermektedir.

   Bununla beraber, bu dosyanın varlığı programın davacı tarafından geliştirildiğini kanıtlamamaktadır. Bunun sebebi, "Yöntemin Tarifi" kısmının 2. paragrafında anlatıldığı üzere, bir programın kaynak koduna sahip olunduğunda, o kod üzerinde istenilen değişikliklerin rahatlıkla yapılabilmesidir. Buna göre, davacı taraf davalı taraf tarafından geliştirilmiş olan programı alıp anılan dosyayı ve yazıyı eklemiş olabilir veya davalı taraf davacı taraf tarafından geliştirilmiş olan programı alıp anılan dosyayı ve yazıyı silmiş olabilir.
4. Her iki programa ait dosyaların bire bir karşılaştırılması sırasında başka farklılıklar da tespit edilmiştir. Bunların bazıları aşağıda listelenmiştir:

a) Programda kullanılan veri dosyaları, *data1.dbc* dosyasında görülmektedir. Davacı tarafın *data1.dbc* dosyasında 19 veri dosyası varken, davalı tarafın *data1.dbc* dosyasında 16 veri dosyası yer almaktadır.

b) Program çalıştırıldığında, kullanıcının önüne menü seçeneklerini getiren *anamenu.scx* dosyası aktif hale geçmektedir. Davacı tarafın *anamenu.scx* dosyasının ‘keypress’ modülünde, 3. paragrafta anılan *release.scx* dosyasını

çalıştıran bir kısım mevcuttur; davalı tarafın *anamenu.scx* dosyasının 'keypress' modülü ise boştur.

c) Davacı tarafın *brj.prg* kod dosyasında *dtur.dbf* isimli veri dosyası kullanılırken, davalı tarafın *brj.prg* kod dosyasında *ramazan.dbf* isimli veri dosyası kullanılmaktadır. Her iki veri dosyası da aynı amaca yöneliktir; sadece isimleri farklıdır.

d) Davacı tarafın programında *fyt* isimli kod ve veri dosyaları varken, bunlar davalı tarafın programında mevcut değildir.

e) Davacı tarafın programında kurye bilgilerini tuttuğu anlaşılan *kurye.dbf* veri dosyasında 63 kayıt bulunurken, davalı tarafın programındaki *kurye.dbf* dosyasının içi boştur.

5. Her iki tarafın programlarındaki dosyaların yaratılma ve güncellenme tarihleri (ve saatleri) de birbirleriyle karşılaştırılmıştır. Aynı isimli dosyaların büyük çoğunluğunda tarihler aynıdır. Bunun anlamı, dosyanın bir taraf tarafından yaratıldığı ve diğer tarafın bu dosyayı kopyaladıktan sonra üzerinde herhangi bir değişiklik yapmadığıdır. Bazı dosyalarda ise davacı tarafın dosyasının tarihi davalı tarafın aynı isimli dosyasının tarihinden ileridir. Bazı dosyalarda ise bunun tersidir.

   Bilgisayar kullanımına hakim bir kişi tarafından, bilgisayar üzerindeki dosyaların tarihini istediği gibi değiştirmek oldukça kolaydır. Buna göre, dosyaların tarihlerini incelemek, genellikle dosyanın ilk olarak kimin tarafından hazırlandığını gösterme konusunda yeterli bir kanıt değildir.

   Ayrıca, programı kopyalayan kişi dosyanın içeriğini sonradan değiştirmiş olabilir (böylece, kopyalayan kişideki dosyanın tarihi programı geliştiren kişideki dosyanın tarihinden ileride olabilir) ya da programı geliştiren kişi dosya kopyalandıktan sonra dosyanın içeriğini değiştirmiş olabilir (böylece, geliştiren kişideki dosyanın tarihi programı kopyalayan kişideki dosyanın tarihinden ileride olabilir).

6. Davalı tarafça sunulan dokümanlar da incelenmiştir. Bu dokümanlar, programda yer alan bütün dosyaların listesini ve dosyaların içeriklerinin ayrıntılı bir

dökümünü vermektedir. Bu dökümlerin, programdaki dosyaların kaynak kodlarını yazıcıdan bastırarak alındığı anlaşılmaktadır.

7. Yerleşik yazılım mühendisliği yöntemleri, gerek bu gibi ihtilafların çözülmesi, gerekse de karmaşık bir mühendislik ürünü olan programların bakımının kolaylığı için, böyle ürünlerin geliştirilmesi sırasında şirket içinde yazılım hakkında spesifikasyon, tasarım ve geliştirme dokümanların üretilmesini öngörür, ancak iki taraf da bu tür bir kanıt öne sürmemiştir.

3.2.2. Örnek Dava 2

Söz konusu dava kaynak kodların başka bir programda kullanıldığına ilişkin şikayet üzerine açılmıştır. Bu davada bilirkişi incelemesi uzun olmakla birlikte sadece konuyla alakalı olan kısımlar verilmiştir. Mahkeme söz konusu davada, kaynak kodların MOSS ile incelenmesini bilirkişilerden talep etmiştir. Bilirkişi ise MOSS veya herhangi başka bir kaynak kod intihal tespit yazılımı kullanmayıp gözle satır satır ve dosya dosya kaynak kod benzerliğini tespit etmeye çalışmıştır.

- **Teknik Bilirkişi İncelemesi**

Mahkeme: TC. İstanbul 4. Fikri ve Sınai Haklar Hukuk Mahkemesi

Dosya No: 2006/33

Davacı: A Bilişim San. ve Tic. AŞ.

Davalı: B Bilgisayar Yazılım ve Tic. Ltd. Şirketi

Davanın konusu:

Davaya konu yazılımlar (‘X Ambar Yönetim Sistemi’ ile ‘Y Ambar Yönetim Sistemi’) arasındaki ilişkinin birbirine benzerlikleri ya da farklı oldukları şekilde tespiti ile buna bağlı olarak (karşılıklı şekilde) tecavüzün men’i ve ref’i, maddî ve manevî tazminat, hükmün ilanı talepleri (dava ve karşı dava olarak)

Mahkemenin Talimatı:

Mahkemenin 27.2.2009 günlü ara kararında önceki teknik rapora karşı itirazlar dikkate alınarak muhasip bilirkişi dışındaki önceki bilirkişilerce ortak ek rapor tanzim etmeleri, bu ek raporda davalının sonradan ibraz ettiği kaynak kodları değil 22.5.2006 tarihli dilekçe ekinde ibraz ettiği kaynak kodları ile davacının önceki kaynak kodlarının

MOSS inceleme sistemine göre ve gerekirse yazılımın satırları bakımından incelenip karşılaştırılarak rapor alınması istemiştir.

Bilirkişi İncelemesi:

Mahkemenin isteği doğrultusunda her iki tarafın kaynak kodlarının karşılaştırması için ve 17.11.2006 tarihli bilirkişi raporunda verilen **MOSS** programı neticeleri değerlendirmek üzere tarafımızca **MOSS** programı kullanılmak istenmiş, ancak kullanım için istediğimiz kullanıcı ve şifre bilgileri ilgili kuruluş tarafından verilmemiştir. **Dolayısıyla MOSS kullanarak değerlendirme yapılamamıştır. CodeSuite** isminde programın limitli demo sürümü internet ortamında indirilerek karşılaştırma işlemi yapılmak istendi. Ancak inceleme kapsamı (dosya sayısı ve boyutu) büyük olduğundan programın lisansı gerektiğinden bu programda da çalışma yapılamadı.

Bu durumda 17.11.2006 tarihli bilirkişi raporunda verilen **MOSS** programı neticelerine bakarak benzerlik oranları yüksek çıkan kaynak dosyaları **satır satır incelenerek** değerlendirilmiştir. Söz konusu kaynak kodlarının **dosya dosya karşılaştırılması** neticesinde elde edilen sonuçlar değerlendirildiğinde, özellikle davacı-karşı davalının *MüşteriA* firmasındaki uygulamasında 11 kaynak dosyasındaki benzerlik oranı yüksek çıkmıştır. Görsel olarak bu dosyaların davalı-karşı davacı kaynak kodlarında bulunan dosyaları incelendiğinde **MOSS** programının bulduğu benzerliğin doğru olduğu görülmektedir. Yani davalı-karşı davacı kaynak dosyalarındaki kodları listelenen davacı-karşı davalı firmanın 11 dosyasındaki kodlarla benzerlikler içermekte, davalı-karşı davacının kaynak dosyaları bunların üzerine uygulamaya dönük olarak (davalı-karşı davacının 11 adet kaynak dosyasında) dosya isim değişiklikleri ve eklemeler olduğu görülmüştür.

4. BULGULAR

Çalışmanın "Malzeme ve Yöntem" bölümünde benzerlik tespiti yapılan *7zip* yazılımında kaynak kod intihal ve klon kod tespit yazılımlarının iki C kaynak kodu arasındaki verdiği benzerlik oranı Tablo 4.1'de verilmiştir. Klon kod tespit yazılımlarının tespit edebildikleri klon kod bloklarının ve klon kod satırlarının sayıları ise Tablo 4.2'de verilmiştir. HoDoKu yazılımının Plaggie ile iki Java kaynak kodu arasında yapılan benzerlik ölçümünün sonucu ise Tablo 4.3'te verilmiştir.

Tablo 4.1: 7zip yazılımında tespit edilen benzerlik oranları

Araç	Benzerlik Oranı % (XzEnc_910.c –XzEnc_922.c)
MOSS	85
JPlag	89
CCFinder	□
Sherlock	52
Simian	88

Tablo 4.2: 7zip yazılımının versiyonlarında tespit edilen klon kod satır sayıları

Araç	Tespit edilen klon blok sayısı (XzEnc_910.c –XzEnc_922.c)	Tespit edilen klon kod satır sayısı (XzEnc_910.c –XzEnc_922.c)
SIM	14 blok	354 satır
CPD	4 blok	249 satır
Duplo	11 blok	201 satır

Tablo 4.3: 7zip yazılımında tespit edilen benzerlik oranları

Araç	Benzerlik Oranı % (Sudoku1.0.java – Sudoku2.0.java)
Plaggie	96.5

"Malzeme ve Yöntem" bölümünde incelenen iki davada ise bilirkişilerin herhangi bir kaynak kod intihal tespit veya klon kod yazılımı kullanmadıkları görülmüştür. Örnek

Dava 1'de kaynak kod dosyalarının benzerliği için hiçbir yazılım kullanılmamıştır. Örnek Dava 2'de mevcut bilirkişinin herhangi bir yazılım kullanmadığı fakat daha önce görevlendirilen bilirkişinin MOSS ile benzerlik ölçümü yaptığı bilirkişi raporundan anlaşılmaktadır.

Normal metinlere göre tespit etmesi hayli zor olan kaynak kod intihalinin hiçbir yazılım kullanmadan tespit edilmesi sağlıklı sonuçlar vermeyecektir. Yüzlerce kaynak kod dosyasını ve bunların binlerce satırlık kodlarını gözle kontrol ederek benzerlik tespitinin yapılması mümkün gözükmemektedir. Bu alanda geliştirilmiş olan yazılımların kullanılması mahkeme heyetinin doğru karar vermesinde etkili rol oynayacak ve adaletin doğru bir şekilde tesisini sağlayacaktır.

Bu alanda geliştirilen araçların bir kısmı iki kaynak kod veya iki klasördeki dosyalar arasında kaynak kod intihal tespiti yaparak dosyaların birbirine benzerlik oranlarını vermektedir. Benzerlik oranlarını vermeyen ve sadece klonlanan kod bloklarını ve satırlarını gösteren yazılımlar da bu alandaki ihtiyaca yönelik olarak kullanılabilir.

Kaynak kod intihali ve klon kod tespiti yapan yazılımlar aynı zamanda akademide bilgisayar programlama derslerinde verilen ödevler arasında kopya ve intihal tespiti yapmak için de kullanılmaktadır (Cosma ve Joy, 2006). Öğrencilere verilen programlama ödevlerinde öğrencilerin ödevleri birbirlerine vererek ödev program üzerinde birkaç değişiklik yaparak aynı yapıyı kullandıkları bilinen bir durumdur. Bu yazılımların kullanılması ile benzetilerek yapılan ödevlerin tespiti çalışan ve kopya çeken öğrencilerin ayırt edilmesini ve sağlıklı not dağıtımını sağlayacaktır.

5. TARTIŞMA VE SONUÇ

Bilgisayar programları ülkemizde fikrî ürün olarak değerlendirilmekte ve FSEK ile korunmaktadır. Fakat bilgisayar programının eser olarak sahiplerine verilen haklar diğer eser sahiplerine verilen haklarla aynıdır. Kanunda bilgisayar programlarının kendine has yapısı göz önüne alındığında kanunda çok fazla üzerinde durulmamış sadece esas noktalarda açıklamalar ve tanımlar getirilerek mesele çözülmeye çalışılmıştır. Oysa gelişmiş ülkelerde bilgisayar programları için ayrı bir düzenleme yoluna gidilerek bilgisayar programları üzerindeki haklar genel fikrî mülkiyet çerçevesinde fakat bilgisayar programlarının gerektirdiği düzenlemeler ve tanımlamaları içerecek şekilde ele alınmaktadır. Ülkemizde bilgisayar programlarının fikrî mülkiyet durumu ve bilişimle ilgili suçlar ayrı kanunlarda tarif edilmekte ve düzenlenmektedir. Bu sorunun çözümü için en akılcı yol bilgisayar programlarının ve diğer bilişimle ilgili tanımlamaların fikrî mülkiyet kanunları esas alınarak tanım, suç ve cezaların bir kanun altında düzenlenmesi olacaktır.

Yazılımın çeşitli hukukî problemleri arasında kaynak kodların intihal edilmesi de yer almaktadır. Kaynak kodlar FSEK ile koruma altına alınan bilgisayar programı unsurları arasındadır. Kaynak kod intihali bilgisayar programını geliştiren kişiler açısından maddî ve manevî zararlara yol açmaktadır. FSEK eser sahiplerinin haklarına tecavüz halinde zararın tazmini ve tecavüzün kaldırılması için düzenlemeler getirmiştir.

Hukukî açıdan koruma sağlanan kaynak kodların intihali durumunda intihalin tespiti ise bir diğer sorun olarak belirmektedir. Kaynak kod intihalinin tespiti diğer intihal türlerinden yapı olarak da farklılık göstermektedir. Kaynak kodlar normal metinlere göre daha kurallı bir yapı arz ettiklerinden kaynak kod intihalinin tespiti diğer metin türlerine göre daha zordur.

Kaynak kodlarda intihal yapısal ve metinsel intihal olarak iki kısma ayrılmaktadır. Kodda değişken ve fonksiyon isimlerinin değişimi, basit ekleme çıkarma gibi kaynak

kod üzerinde basit değişiklikler metinsel değişiklikler olarak değerlendirilmekte ve bu tarz kaynak kod intihalleri kolaylıkla tespit edilmektedir. Yapısal intihaller ise programın mantığının farklı bir şekilde ifade edilmesidir. Kaynak kod komut ve dizilim açısından farklı gözükse de işlevi aynı kalmaktadır. Bu tarz intihallerin tespiti ise daha zordur.

Kaynak kod intihalini tespit için çeşitli teknikler ve bu tekniklere dayalı çeşitli araçlar geliştirilmiştir. Bu araçlardan MOSS gibi kimisi kaynak kodların birbirine benzerlik oranını vermekte CPD gibi kimi araçlar ise sadece klonlanmış kod satırlarını tespit edebilmektedir. Tezde bu alanda geliştirilmiş dokuz yazılımın kullanımı incelenmiştir.

Kaynak kod intihali sadece yazılım sektöründe değil akademide de önemli bir sorundur. Öğrencilere verilen programlama ödevlerinin kopyalanarak ve üzerinde az bir değişiklik yapılarak sunulması bu alanda geliştirilen yazılımların önemini daha da artırmaktadır.

Kaynak kod benzerliği yazılım sektöründe de önemli bir sorundur. Zaman problemi yüzünden kodların kaynak kod dosyaları arasında kopyalanması kısa vadede programcının istediğini yerine getirse de uzun vadede yazılımda karmaşıklık yaratacak, aynı fonksiyonun bir çok kaynak kod dosyasında yer alması ve bu kodda bir düzeltme ihtiyacı hasıl olduğunda kodun kopyalandığı tüm dosyalarda düzeltme yapılması gerekecektir. Bu ise yazılımın bakım maliyetini artıracak ve yazılım üretiminin süresinin uzamasına sebep olacaktır. Kaynak kod benzerlik tespit araçları bu durumlarda çok önemli işlev yerine getirmektedir. Kopyalanan kod bloklarının tespiti ile bunlar tek bir dosyada veya tek bir fonksiyon altında birleştirilerek yazılım daha stabil olacak ve bakım maliyetleri ve zamanı da düşecektir.

Tezde incelenen örnek davalarda görülmüştür ki, kaynak kod intihali herhangi bir araç kullanılmadan gözle kontrol edilerek tespit edilmeye çalışılmıştır. Binlerce satırlık kodların gözle benzerlik ve intihal tespitini yapmak sağlıklı sonuçlar vermeyecektir. Bilirkişilerin bu konuda geliştirilmiş yazılımları kullanımı, mahkeme heyetinin doğru ve adil karar vermesinde etkili olacaktır. Tez, kaynak kod intihalini tespit eden yazılımların kullanımını içermesi bakımından hem bilirkişiler, hem akademisyenler hem de yazılım sektörü çalışanları için önem taşımaktadır.

KAYNAKLAR

ADALET BAKANLIĞI, 2005, "Fikrî Ve Sınaî Hak Korsanlığı Ve Taklitçiliğinde Devletin Vergi Kaybı Hakkında Genelge (Avrupa Birliği Genel Müdürlüğü)", http://www.mevzuat.adalet.gov.tr/html/26890.html, [Ziyaret Tarihi: 29.06.2011].

AHO, A.V. ve diğ., 2006, *Compilers: Principles, Techniques, and Tools*, 2. ed, Pearson/Addison Wesley, ISBN 978-0321486813.

AHTIAINEN, A. ve diğ., 2006, *Plaggie: GNU-licensed source code plagiarism detection engine for Java exercises*, *Içinde*: *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, New York, ACM, (Baltic Sea '06), 141–142, doi: 10.1145/1315803.1315831.

AIKEN, A., 2011, "Plagiarism Detection", http://theory.stanford.edu/~aiken/moss/, [Ziyaret Tarihi: 05.12.2011].

AIST, 2010, "The archive of CCFinder Official Site", http://www.ccfinder.net/ccfinderxos.html, [Ziyaret Tarihi: 06.12.2011].

AKSOY, P., DENARDIS, L., 2007, *Information Technology in Theory*, 1. ed, Boston Mass., Thomson Course Technology, ISBN 9781423901402.

AKSU, M., 2006, *Bilgisayar programlarının fikrî mülkiyet hukukunda korunması*, 1. ed, İstanbul, Beta, ISBN 9789752955127.

ARORA, A., BANSAL, S., 2005, *Comprehensive Computer and Languages*, Delhi, Laxmi Publications, ISBN 8170083559.

ARWIN, C., TAHAGHOGHI, S.M.M., 2006, *Plagiarism detection across programming languages*, *Içinde*: *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, Darlinghurst, Australia, Australian Computer Society, Inc., (ACSC '06), 277–286, ISBN 1-920682-30-9.

ATEŞ, M., 2003, *Fikir ve Sanat Eserleri Üzerindeki Hakların Kapsamı ve Sınırlandırılması*, 1. ed, Ankara, Seçkin Yayıncılık, ISBN 9753475802.

BASIT, H.A., 2006, "Analysis and semi-automated detection of design-level similarity patterns in software", Ph.D. Thesis, National Universıty Of Singapore.

BAYAMLIOĞLU, İ.E., 2007, "Fikir Ve Sanat Eserleri Hukukunda Teknolojik Koruma", Doktora Tezi, Marmara Üniversitesi Özel Hukuk Anabilim Dalı.

BEŞİROĞLU, A., 2004, *Düşünce Ürünleri Üzerinde Haklar*, 3. ed, İstanbul, Beta Basım Yayım, ISBN 9752953476.

BİRLEŞMİŞ MİLLETLER, 2011, "İnsan hakları evrensel beyannamesi", http://unic.un.org/aroundworld/unics/common/documents/publications/udhr/ankara_udhr_turkish.pdf, [Ziyaret Tarihi: 30.06.2011].

BLOSS, A., 2003, *Language processors*, *Içinde*: Ralston, A. ve diğ. (ed.) *Encyclopedia of Computer Science*, Chichester-UK, John Wiley and Sons Ltd., 955–958, ISBN 0-470-86412-5.

BLUNDELL, B.G., 2007, *Computer Hardware*, London, Thomson, ISBN 9781844807512.

BRITISH AND IRISH LEGAL INFORMATION INSTITUTE, 1998, "Flyde Microsystems Limited v. Key Radio Systems Limited [1998] EWHC Patents 340 (11th February, 1998)", http://www.bailii.org/ew/cases/EWHC/Patents/1998/340.html, [Ziyaret Tarihi: 27.09.2011].

BRONSON, G.J., ROSENTHAL, D.A., 2005, *Introduction to Programming with Visual Basic .NET*, 1. ed, Sudbury-MA, Jones & Bartlett Learning, ISBN 9780763724788.

BSA, 2011, "Eighth Annual Bsa Global Software 2010 Piracy Study", http://portal.bsa.org/globalpiracy2010/downloads/study_pdf/2010_BSA_Piracy_Study-Standard.pdf, [Ziyaret Tarihi: 16.06.2011].

BT DÜNYASI, 2011, "Türkiye Bilişim Sektörü 27,3 milyar doları buldu - BTdünyası", http://www.btdunyasi.net/index.php?module=news&news_id=8272&cat_id=5, [Ziyaret Tarihi: 15.06.2011].

BURD, E., BAILEY, J., 2002, *Evaluating Clone Detection Tools for Use during Preventative Maintenance*, *Içinde*: *Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation*, Washington, IEEE Computer Society, (SCAM '02), 36-43, doi: 10.1109/SCAM.2002.1134103, ISBN 0-7695-1793-5.

BURROWS, S. ve diğ., 2007, "Efficient plagiarism detection for large code repositories", *Içinde*: *Software: Practice and Experience*, 37 (2), 151-175, doi: 10.1002/spe.750.

CBR STAFF WRITER, 2010, "Global IT spending to reach $1.6 trillion in 2011, says IDC - Computer Business Review", http://itservices.cbronline.com/news/global-it-spending-to-reach-16-trillion-in-2011-says-idc_031210, [Ziyaret Tarihi: 15.06.2011].

CERİTOĞLU, F., 2008, "Fikir Ve Sanat Eserleri Hukukunda İntihal Ve Esinlenme", Yüksek Lisans Tezi, Marmara Üniversitesi Sosyal Bilimler Enstitüsü.

CHILOWICZ, M. ve diğ., 2009, "Finding Similarities in Source Code Through Factorization", *Içinde*: *Electronic Notes in Theoretical Computer Science*, 238 (5), 47-62, doi: 10.1016/j.entcs.2009.09.040.

CİMİLLİ, C. ve diğ., 2004, *Yazılımlar Üzerindeki Hakların FSEK Çerçevesinde İncelenmesi*, *Içinde*: *Akademik Bilişim 2004*, KTÜ, Trabzon.

CİNOĞLU, F., 2010, "Eser Üzerinde İşverenin Hak Sahipliği", Yüksek Lisans Tezi, Kadir Has Üniversitesi Sosyal Bilimler Enstitüsü.

COSMA, G., 2008, "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis", Ph.D. Thesis, University of Warwick.

COSMA, G., JOY, M., 2006, "Source-code plagiarism: A UK academic perspective", *Içinde*: *Proc. 7th Annu. Conf. Higher Education Academy Network for Information and Computer Sciences*, 116-120.

COSMA, G., JOY, M., 2008, "Towards a Definition of Source-Code Plagiarism", *Içinde*: *IEEE Transactions on Education*, 51 (2), 195-200, doi: 10.1109/TE.2007.906776.

COŞGUN, D., 2006, "FİKRİ VE SINAİ HAKLAR", http://www.fkmd.org/index.php?option=com_content&view=article&id=171:fykry-ve-sinay-haklar&catid=19&Itemid=79, [Ziyaret Tarihi: 16.11.2011].

COVERITY.COM, 2004, "Coverity Incorporated: Press Release: Story 3 | Coverity", http://www.coverity.com/html/press_story03_12_14_04.html, [Ziyaret Tarihi: 21.09.2011].

CROFT, D.W., 2003, "Open Source License Comparison \ Tutorials \ Library \ CroftSoft", http://croftsoft.com/library/tutorials/opensource/, [Ziyaret Tarihi: 11.11.2011].

ÇÖLKESEN, R., 2002, *Bilgisayar programlama ve yazılım mühendisliğinde veri yapıları ve algoritmalar*, İstanbul, Papatya, ISBN 9789756797235.

DALYAN, Ş., 2008, "Bilgisayar Programlarının Fikrî Hukukta Korunması", Yayımlanmış Doktora Tezi, Ankara Üniversitesi Sosyal Bilimler Enstitüsü.

DARNELL, P.A., MARGOLIS, P.E., 1996, *C, A Software Engineering Approach*, 3. ed, New York, Springer-Verlag New York Inc., ISBN 9780387946757.

DAVEY ve diğ., 1995, "The development of a software clone detector.", *Içinde*: *International Journal of Applied Software Technology*, 1 (3-4), 219-236.

DAVIDSON, C.H., 2003, *Source program*, *Içinde*: Ralston, A. ve diğ. (ed.) *Encyclopedia of Computer Science*, 4. ed, Chichester-UK, John Wiley and Sons Ltd., 1664, ISBN 0-470-86412-5.

DAVIS, R. ve diğ., 1996, "A new view of intellectual property and software", *Içinde*: *Communications of the ACM*, 39 , 21-30, doi: 10.1145/227234.227237.

EĞİTEK, 2011, "Mülkiyet", http://egitek.meb.gov.tr/aok/aok_kitaplar/AolKitaplar/hukuk_1/3.pdf, [Ziyaret Tarihi: 26.06.2011].

ELO, T., HASU, T., 2003, *Detecting Co-Derivative Source Code—An Overview*, *Içinde*: *Teknis-juridinen selvitys tekijänoikeudesta tietokoneohjelman lähdekoodiin Suomessa ja Euroopassa*, Helsinki Chamber of Commerce, 1-48.

ENCYCLOPÆDIA BRITANNICA ONLINE, 2011, "Plagiarism", http://www.britannica.com/EBchecked/topic/462640/plagiarism, [Ziyaret Tarihi: 14.11.2011].

ERALP, Ö., 2011, "Bilişim Suçları Türk Ceza Kanunu Madde 243 - Bilişim Sistemine Girme", http://www.ozgureralp.av.tr/makaleler/bilisimsuclarinagirme243.html, [Ziyaret Tarihi: 27.12.2011].

ERDOĞAN, B., 2005, "Programlama Başarısı İle Akademik Başarı, Genel Yetenek, Bilgisayara Karşı Tutum, Cinsiyet ve Lise Türü Arasındaki İlişkilerin İncelenmesi", Yüksek Lisans Tezi, Marmara Üniversitesi Eğitim Bilimleri Enstitüsü.

EREL, Ş.N., 1994, "Fikri Hukukta Bilgisayar Programlarının Korunması", *Içinde*: *Ankara Üniversitesi Siyasal Bilgiler Fakültesi Dergisi*, 49 (1-2), 141-164.

EROĞLU, S., 2000, *Rekabet hukukunda bilgisayar programlarının korunması*, 1. ed, İstanbul, Beta, ISBN 9789754868739.

ESL, I., 2001, *Introduction to Information Technology*, 1. ed, India, Pearson, ISBN 9788177581188.

EU, 2011, "EUR-Lex - 31991L0250 - EN", http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:NOT, [Ziyaret Tarihi: 04.08.2011].

FAIDHI, J.A.W., ROBINSON, S.K., 1987, "An empirical approach for detecting program similarity and plagiarism within a university programming environment", *Içinde*: *Comput. Educ.*, 11 (1), 11–19, doi: 10.1016/0360-1315(87)90042-X.

FEARNLEY, C., 2010, "The Nature and Importance of Source Code and Learning Programming with Python", http://blog.remoteresponder.net/category/systems-management/programming/, [Ziyaret Tarihi: 24.05.2011].

FELLER, J., FITZGERALD, B., 2002, *Understanding Open Source Software development*, Addison-Wesley, ISBN 9780201734966.

FISHER, T., 2011, "Version Number Definition", http://pcsupport.about.com/od/termsv/g/version-number.htm, [Ziyaret Tarihi: 21.11.2011].

FREEDMAN, A., 2001, *Computer desktop encyclopedia*, *Içinde*: 9. ed, New York, Osborne/McGraw-Hill, ISBN 9780072193060.

FREIRE, M. ve diğ., 2007, "AC: An Integrated Source Code Plagiarism Detection Environment", *Içinde*: *arXiv:cs/0703136*.

GARTNER, 2011, "Forecast Alert: IT Spending, Worldwide, 2008-2015, 2Q11 Update", http://www.gartner.com/DisplayDocument?doc_cd=214540, [Ziyaret Tarihi: 25.07.2011].

GALBRAITH, P., 2009, *Developing Web Applications with Apache, MySQL, Memcached, and Perl*, Indianapolis, John Wiley & Sons, ISBN 9780470538326.

GODSE, A.P., GODSE, D.A., 2008, *Fundmentals Of Programming*, 1. ed, India, Technical Publications, ISBN 9788184313048.

GOEL, A., 2010, *Computer Fundamentals*, New Delhi, Pearson Education India, ISBN 9788131733097.

GOEL, S., RAO, D., 2005, *Plagiarism and its Detection in Programming Languages*, JIITU, Department of Computer Science and Information Technology.

GOLDMAN, R., GABRIEL, R.P., 2005, *Innovation Happens Elsewhere: Open Source as Business Strategy*, Amsterdam, Morgan Kaufmann, ISBN 9781558608894.

GÖKYAYLA, E., 2009, *Fikir ve Sanat Eserleri Hukukunda İntihal Kavramı*, *Içinde*: *Fikri Mülkiyet Hukuku Yıllığı*, 1. ed, İstanbul, On İki Levha Yayıncılık, 293-308, ISBN 9771309361000.

GRUNE, D., 2008, ftp://ftp.cs.vu.nl/pub/dick/similarity_tester/sim.pdf, [Ziyaret Tarihi: 13.12.2011].

GÜNDEM, O., 2006, "Fikir Ve Sanat Eserleri Kanununda Eser Sahibinin Haklarına Bağlantılı Haklar, Bu Hakların Sınırlandırılması Ve Korunması", Yüksek Lisans Tezi, Kırıkkale Üniversitesi Sosyal Bilimler Enstitüsü.

HAGE, J. ve diğ., 2010, *A comparison of plagiarism detection tools*, Technical Report, Utrecht University Department of Information and Computing Sciences.

HAMILTON, J., 2008, "Static Source Code Analysis Tools and their Application to the Detection of Plagiarism in Java Programs", Dissertation, University of London Department of Computing at Goldsmiths.

HARRIS, S., 2011, "Simian - Similarity Analyser | Duplicate Code Detection for the Enterprise | Features", http://www.harukizaemon.com/simian/features.html, [Ziyaret Tarihi: 05.12.2011].

HENDERSON, H., 2009, *Encyclopedia of Computer Science and Technology*, REV. ED., New York, Facts On File, ISBN 9780816063826.

HIGO, Y., 2006, "Code Clone Analysis Methods for Efficient Software Maintenance", Ph.D. Thesis, Osaka University.

HIRSCH, E., 1950, "Bern Sözleşmesi", *Içinde*: *Ankara Üniversitesi Hukuk Fakültesi Dergisi*, 7 (1-2), 130-145.

HIRSCH, E.E., 1943, *Hukuki Bakımdan Fikri Say*, İkinci Cilt, İstanbul.

HUKUKTURK, 2004, "Danıştay 10. Dairesi - Esas No: 1992/4550 - Karar No :1994/1856 - Tarih: 27/04/1994 | HukukTürk | Mevzuat-İçtihat Bilgi Bankası | Danıştay Kararları", http://www.hukukturk.com/fractal/hukukTurk/pages/findDanistayDetail_n.jsp?&pKararId=1334&pMevzuatId=3592, [Ziyaret Tarihi: 04.10.2011].

ICS, H.E.A., 2011, "Source Code Similarity Detection Tools - Plagiarism", http://www.ics.heacademy.ac.uk/resources/assessment/plagiarism/detectiontools_sourcecode.html, [Ziyaret Tarihi: 15.11.2011].

INFOETHER, 2011, "PMD - Finding copied and pasted code", http://pmd.sourceforge.net/cpd.html, [Ziyaret Tarihi: 08.12.2011].

JAFAR, Y.M., 2007, "Clone Detection Using Pictorial Similarity in Slice Traces", Masters Thesis, Kings College London.

JI, J.-H. ve diğ., 2007, *A source code linearization technique for detecting plagiarized programs*, *Içinde*: *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, New York, ACM, (ITiCSE '07), 73–77, doi: 10.1145/1268784.1268807, ISBN 978-1-59593-610-3.

JIA, Y., 2007, "Clone Detection Using Dependence Analysis and Lexical Analysis", Masters Thesis, Department of Computer Science King's College London.

JIANG, L., 2009, "Scalable Detection of Similar Code: Techniques and Applications", Ph.D. Thesis, University Of California Davis.

JIANG, Z.M., 2006, "Visualizing and Understanding Code Duplication in Large Software Systems", Masters Thesis, University of Waterloo.

JONES, E.L., 2001, *Metrics based plagarism monitoring*, *Içinde*: *IN 'Proceedıngs Of The Sixth Annual CCSC Northeastern Conference*, Middlebury, Consortium for Computing Sciences in Colleges, (CCSC '01), 253–261.

JONES, T.C., 1984, "Reusability in Programming: A Survey of the State of the Art", *Içinde*: *IEEE Transactions on Software Engineering*, SE-10 (5), 488-494, doi: 10.1109/TSE.1984.5010271.

JOY, M., LUCK, M., 1999, "Plagiarism in programming assignments", *Içinde*: *IEEE Transactions on Education*, 42 (2), 129-133, doi: 10.1109/13.762946.

KAMIYA, T., 2008, "Tutorial of GUI front-end GemX", http://www.ccfinder.net/doc/10.2/en/tutorial-gemx.html, [Ziyaret Tarihi: 06.12.2011].

KAMIYA, T. ve diğ., 2002, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code", *Içinde*: *IEEE Transactions on Software Engineering*, 28 (7), 654- 670, doi: 10.1109/TSE.2002.1019480.

KAPSER, C., GODFREY, M.W., 2006, *«Cloning Considered Harmful» Considered Harmful*, *Içinde*: *Proceedings of the 13th Working Conference on Reverse Engineering*, Washington, IEEE Computer Society, 19–28, doi: 10.1109/WCRE.2006.1, ISBN 0-7695-2719-1.

KAPSER, C., GODFREY, M.W., 2003, *Toward a taxonomy of clones in source code: A case study*, *Içinde*: *Proc. Int'l Workshop on Evolution of Large Scale Industrial Software Architectures (ELISA)*, 67–78.

KARAHAN, S. ve diğ., 2009, *Fikri Mülkiyet Hukukunun Esasları*, 2. ed, Ankara, Seçkin Yayıncılık, ISBN 9789750210624.

KHASON, T., 2009, "Open Source licenses comparison table", http://khason.net/blog/open-source-licenses-comparison-table/, [Ziyaret Tarihi: 11.11.2011].

KIM, M. ve diğ., 2004, *An Ethnographic Study of Copy and Paste Programming Practices in OOPL*, *Içinde*: *Proceedings of the 2004 International Symposium on Empirical Software Engineering*, Washington, IEEE Computer Society, 83–92, doi: 10.1109/ISESE.2004.10, ISBN 0-7695-2165-7.

KOMONDOOR, R.V., 2003, "Automated duplicated code detection and procedure extraction", Ph.D. Thesis, The University of Wisconsin - Madison.

KONECKI, M. ve diğ., 2009, *Detecting computer code plagiarism in higher education*, *Içinde*: *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces, 2009. ITI '09*, Cavtat/Dubrovnik-Croatia, IEEE, 409-414, doi: 10.1109/ITI.2009.5196118, ISBN 978-953-7138-15-8.

KONI-N'SAPU, G.G., 2001, "A Scenario Based Approach for Refactoring Duplicated Code in Object Oriented Systems", Masters Thesis, University of Bern.

KOSCHKE, R. ve diğ., 2006, *Clone Detection Using Abstract Syntax Suffix Trees*, *Içinde*: *Proceedings of the 13th Working Conference on Reverse Engineering*, Washington, IEEE Computer Society, 253–262, doi: 10.1109/WCRE.2006.18, ISBN 0-7695-2719-1.

KUDYBA, S., DIWAN, R.K., 2002, *Information Technology, Corporate Productivity, and the New Economy*, Westport Conn., Quorum Books, ISBN 9781567204209.

KULAKLI, E., 2008, "Fikir Ve Sanat Eserleri Kanunu'na Göre Eser, Bilgisayar Programlarının Eser Niteliği Ve Eser Sahipliğinin Belirlenmesi, Bilgisayar Programları Açısından Eser Sahibi", http://www.odakhukuk.net/akademik/bilgisayar%20programlarinin%20eser%20niteligi%20ve%20eser%20sahibi.pdf, [Ziyaret Tarihi: 05.06.2011].

KUSTANTO, C., LIEM, I., 2009, *Automatic Source Code Plagiarism Detection*, *Içinde*: *Proceedings of the 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, Washington, IEEE Computer Society, (SNPD '09), 481–486, doi: 10.1109/SNPD.2009.62, ISBN 978-0-7695-3642-2.

LECKY-THOMPSON, G.W., 2005, *Corporate software project management*, Hingham, Mass., Charles River Media, ISBN 1584503858 9781584503859.

LEE, K.D., 2008, *Programming Languages: An Active Learning Approach*, New York, Springer, ISBN 9780387794211 0387794212.

LEEMHUIS, T., 2011, "What's new in Linux 2.6.39 - The H Open Source: News and Features", http://www.h-online.com/open/features/What-s-new-in-Linux-2-6-39-1242910.html?page=4, [Ziyaret Tarihi: 08.06.2011].

LINFO, T.L.I.P., 2006, "Source Code Definition", http://www.linfo.org/source_code.html, [Ziyaret Tarihi: 14.05.2011].

LOZANO, A. ve diğ., 2007, *Evaluating the Harmfulness of Cloning: A Change Based Experiment*, *Içinde*: *Proceedings of the Fourth International Workshop on Mining Software Repositories*, Washington, IEEE Computer Society, (MSR '07), doi: 10.1109/MSR.2007.8, ISBN 0-7695-2950-X.

MA, Y.-S., WOO, D.-K., 2007, *Applying a Code Clone Detection Method to Domain Analysis of Device Drivers*, *Içinde*: *Proceedings of the 14th Asia-Pacific Software Engineering Conference*, Washington, IEEE Computer Society, (APSEC '07), 254–261, doi: 10.1109/APSEC.2007.25, ISBN 0-7695-3057-5.

MA, Y.-S., WOO, D.-K., 2008, "Domain Analysis of Device Drivers Using Code Clone Detection Method", *Içinde*: *ETRI Journal*, 30 (3), 394-402, doi: 10.4218/etrij.08.0107.0204.

MAHMOUD, H.A., 2008, *Detecting Disguised Plagiarism*, Technical Report, University of Waterloo.

MCCONNELL, S., 2004, *Code complete*, 2. ed, Redmond Wash., Microsoft Press, ISBN 9780735619678.

MEMİŞ, T., 2009, *Fikri Hukuk Bakımından Kaynak Kodlarının Korunması*, *Içinde*: *Fikri Mülkiyet Hukuku Yıllığı*, 1. ed, İstanbul, On İki Levha Yayıncılık, 293-308, ISBN 9771309361000.

MESSINA, S., 1954, "Doktrinde, Mukayeseli Mevzuatta, Milletlerarası İçtihatta Edebiyat ve Sanat İntihali", *Içinde*: *Ankara Üniversitesi Hukuk Fakültesi Dergisi*, 11 (3), 266-285, doi: 10.1501/Hukfak_0000001223.

MISHNE, G., 2003, "Source Code Retrieval Using Conceptual Graphs", Masters Thesis, University of Amsterdam.

MISHNE, G., DE RIJKE, M., 2004, "Source Code Retrieval using Conceptual Similarity", *Içinde*: *PROC. 2004 CONF. COMPUTER ASSISTED INFORMATION RETRIEVAL (RIAO '04*, (1984), 539--554.

MOZGOVOY, M., 2006, "Desktop Tools for Offline Plagiarism Detection in Computer Programs", *Içinde*: *Informatics in education*, 5 (1), 97-112.

MOZGOVOY, M., 2007, "Enhancing Computer-Aided Plagiarism Detection", Dissertation, University Of Joensuu.

NEW MEDIA RIGHTS, 2008, "Open Source Licensing Guide", http://www.newmediarights.org/open%5Fsource/new%5Fmedia%5Frights%5Fopen%5Fsource%5Flicensing%5Fguide, [Ziyaret Tarihi: 11.11.2011].

NOERGAARD, T., 2010, *Demystifying Embedded Systems Middleware*, 1. ed, Terma, Denmark, Newnes, ISBN 9780750684552.

OXFORD UNIVERSITY PRESS, 2011, "Definition of plagiarism from Oxford Dictionaries Online", http://oxforddictionaries.com/definition/plagiarism, [Ziyaret Tarihi: 12.11.2011].

PARKER, A., HAMBLEN, J.O., 1989, "Computer algorithms for plagiarism detection", *Içinde*: *IEEE Transactions on Education*, 32 (2), 94-99, doi: 10.1109/13.28038.

PARR, T., 2009, "The difference between compilers and interpreters - ANTLR 3 - ANTLR Project", http://www.antlr.org/wiki/display/ANTLR3/The+difference+between+compilers+and+interpreters, [Ziyaret Tarihi: 24.12.2011].

PATE, J.R. ve diğ., 2011, *Clone evolution: a systematic review*, Technical Report, The University of Alabama Department of Computer Science.

PRECHELT, L. ve diğ., 2000, *JPlag: Finding plagiarisms among a set of programs*, Technical Report, University of Karlsruhe Department of Informatics.

RAYMOND, E., 2010, "How do computer languages work?", http://www.linuxdoc.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/languages.html, [Ziyaret Tarihi: 24.12.2011].

RIEGER, M., 2005, "Effective Clone Detection Without Language Barriers", Ph.D. Thesis, University of Berne.

ROSEN, S., 2003, *Software*, *Içinde*: Ralston, A. ve diğ. (ed.) *Encyclopedia of Computer Science*, 4. ed, Chichester-UK, John Wiley and Sons Ltd., 1599–1601, ISBN 0-470-86412-5.

ROY, C.K., CORDY, J.R., 2009, *A Mutation/Injection-Based Automatic Framework for Evaluating Code Clone Detection Tools*, *Içinde*: *International Conference on Software Testing, Verification and Validation Workshops, 2009. ICSTW '09*, Washington, IEEE Computer Society, 157–166, doi: 10.1109/ICSTW.2009.18, ISBN 978-0-7695-3671-2.

ROY, C.K., CORDY, J.R., 2007, "A Survey on Software Clone Detection Research", *Içinde*: *School Of Computing Tr 2007-541, Queen's Unıversıty*, 115.

ROY, C.K. ve diğ., 2009, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach", *Içinde*: *Science of Computer Programming*, 74 (7), 470-495, doi: 10.1016/j.scico.2009.02.007.

SARIDOĞAN, E., 2008, *Yazılım mühendisliği*, İstanbul, Papatya Yayıncılık Eğitim, ISBN 9789756797570.

SCHNIEDER, E., 2011, *Forms/Format 2010: Formal Methods for Automation and Safety in Railway and Automotive Systems*, Springer, ISBN 9783642142604.

SINGH, R., 2009, *Design and Implementation of Compiler*, Delhi-India, New Age International, ISBN 9788122428650.

SOFTIC, 2006, "Microsoft Corp. v. Shuuwa System Trading K.K., Tokyo District Court, opinion of Jan. 30, 1987", http://www.softic.or.jp/en/cases/Microsoft_v_Shuwa.html, [Ziyaret Tarihi: 19.11.2011].

SPINELLIS, D., 2003, *Code reading : the open source perspective*, Boston MA, Addison-Wesley, ISBN 9780201799408.

SRAKA, D., KAUCIC, B., 2009, *Source code plagiarism*, *Içinde*: Luzar-Stiffler, V. ve diğ. (ed.) *Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces*, Cavtat/Dubrovnik-Croatia, IEEE, 461-466, doi: http://dx.doi.org/10.1109/ITI.2009.5196127, ISBN 978-953-7138-15-8.

ST.AMANT, K., 2007, *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives*, 1. ed, IGI Global, ISBN 1591409993.

STACK OVERFLOW, 2009, "Size - How many developers are there in the world? - Stack Overflow", http://stackoverflow.com/questions/453880/how-many-developers-are-there-in-the-world, [Ziyaret Tarihi: 06.10.2011].

SULUK, C., 2004, *Korsanlıkla Mücadele Ve Matbaa Sektörü*, *Içinde*: *2. Uluslararası Katılımlı Kağıt-Karton, Mürekkep, Matbaa Sempozyumu Ve Sergisi*, İzmir, T.M.M.O.B. Kimya Mühendisleri Odası Yayını, 159-178.

SULUK, C., ORHAN, A., 2005, *Uygulamalı fikri mülkiyet hukuku*, İstanbul, Arıkan, ISBN 9789756145258.

TAIRAS, R.A., 2010, "Representation, Analysis, And Refactoring Techniques to Support Code Clone Maintenance", Ph.D. Thesis, The University of Alabama at Birmingham.

TAŞÇI, K. ve diğ., 2007, *Yazılım Endüstrisinin Gelişiminde Fikri Mülkiyet Hakları Koruması*, *Içinde*: *III.Ulusal Yazılım Mühendisliği Sempozyumu*, Ankara, Elektrik Mühendisleri Odası, ISBN 9789944893374.

TBMM, 2007, "Wipo Telif Hakları Andlaşması", http://www.tbmm.gov.tr/kanunlar/k5647.html, [Ziyaret Tarihi: 04.10.2011].

TDK, 2011, "Türk Dil Kurumu - Büyük Türkçe Sözlük", http://tdkterim.gov.tr/bts/, [Ziyaret Tarihi: 27.06.2011].

TECH TERMS DICTIONARY, 2007, "Source Code Definition", http://www.techterms.com/definition/sourcecode, [Ziyaret Tarihi: 08.06.2011].

TEKİN, U. ve diğ., 2009, *Yazılım Klonları ve Klon Belirleme Yöntemleri*, *Içinde*: *IV. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS'09)*, İstanbul, 111-115.

TEKİNALP, Ü., 2005, *Fikri Mülkiyet Hukuku*, Güncelleştirilmiş ve Gözden Geçirilmiş 3. Baskı, İstanbul, Arıkan, ISBN 9789756145494.

TELİF HAKLARI VE SİNEMA GENEL MÜDÜRLÜĞÜ, 2007, "Hakka Tecavüz Halinde Hukuki ve Cezai Prosedür", http://www.kultur.gov.tr/TR/belge/1-26492/eski2yeni.html, [Ziyaret Tarihi: 18.11.2011].

TODD, D.L., 2007, "Free and Open Source License Comparison", http://blogs.oracle.com/davidleetodd/entry/free_and_open_source_license, [Ziyaret Tarihi: 11.11.2011].

TRANSFORM GROUP, 2011, "Program Transformation Wiki / Why Decompilation", http://www.program-transformation.org/Transform/WhyDecompilation, [Ziyaret Tarihi: 30.10.2011].

TUCKER, A.B., NOONAN, R., 2002, *Programming languages: principles and paradigms*, McGraw-Hill, ISBN 9780072381115.

TUNCER, İ., 2009, "Fikri mülkiyet Hakkına Tecavüzün Mevcut Olmadığının Tespiti Davası", *Dokuz Eylül Üniversitesi Hukuk Fakültesi Dergisi*, 11 (Özel Sayı 2009), 617-650.

TÜRK HUKUK KURUMU, 1991, *Türk Hukuk Lûgatı*, 3. ed, Ankara, Başbakanlık Basımevi.

UPB, 2011, "Hukuk ve Ceza Davaları", http://www.upb.org.tr/index.php?option=com_content&view=article&id=19&Itemid=17, [Ziyaret Tarihi: 30.12.2011].

USLU, R., 2003, *Türk Fikir ve Sanat Hukuku'nda Eser Kavramı*, 1. ed, Ankara, Seçkin Yayıncılık, ISBN 9783476493.

UYGUR, A.B., 2004, "Eser Sahibine Tanınan Haklara Getirilen Kısıtlamalar", Yüksek Lisans Tezi, Ankara Üniversitesi Sosyal Bilimler Enstitüsü.

VENKATARAM, S.K., 2005, *Advanced Microprocessor and Microcontroller*, 3. ed, New Delhi, Laxmi Publications, ISBN 8170083109.

VESELOSKY, V., 2011, "Compiling Linux Software from Source Code - Control-Escape", http://www.control-escape.com/linux/lx-swinstall-tar.html, [Ziyaret Tarihi: 08.06.2011].

WANG, W., 2010, *Reverse engineering : technology of reinvention*, 1. ed, Boca Raton FL, CRC Press, ISBN 9781439806302.

WEISSGERBER, P., DIEHL, S., 2006, *Identifying Refactorings from Source-Code Changes*, *Içinde*: *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, Washington, IEEE Computer Society, 231–240, doi: 10.1109/ASE.2006.41, ISBN 0-7695-2579-2.

WILSON, J.A.J., 2011, "Benefits of open source code", http://www.osswatch.ac.uk/resources/whoneedssource.xml, [Ziyaret Tarihi: 22.12.2011].

WIPO, 2011, "Convention Establishing the World Intellectual Property Organization", http://www.wipo.int/treaties/en/convention/trtdocs_wo029.html#P50_1504, [Ziyaret Tarihi: 29.06.2011].

WIPO, 1996, "WIPO Copyright Treaty", http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html, [Ziyaret Tarihi: 04.10.2011].

WTO, 2011a, "Agreement on Trade-Related Aspects of Intellectual Property Rights", http://www.wto.org/english/docs_e/legal_e/27-trips.pdf, [Ziyaret Tarihi: 04.08.2011].

WTO, 2011b, "WTO | intellectual property (TRIPS) - frequently-asked questions", http://www.wto.org/english/tratop_e/trips_e/tripfq_e.htm, [Ziyaret Tarihi: 30.06.2011].

YADAV, A., 2008, *Microprocessor 8085, 8086*, New Delhi-India, University Science Press/Laxmi Publications Pvt. Ltd., ISBN 9788131803561.

YASAMAN, H., 2006, *Fikri ve sınai mülkiyet hukuku : fikir ve sanat eserleri endüstriyel tasarımlar, patentler ile ilgili makaleler, mütalâalar, bilirkişi raporları*, 1. ed, İstanbul, Vedat Kitapçılık Basım Yayım Dağıtım, ISBN 9789758875597.

YILMAZ, Z., 2007, "Eser Ve Eserle İlgili Kavramlar", http://www.muzafferozcan.av.tr/2010/09/27/eser-ve-eserle-ilgili-kavramlar/, [Ziyaret Tarihi: 04.10.2011].

ZEIDMAN, B., 2004, "Detecting Source-Code Plagiarism", *Içinde*: *Dr. Dobb's Journal*, 55-60.

ZEIDMAN, R., 2008, *Multidimensional Correlation of Software Source Code*, *Içinde*: *Third International Workshop on Systematic Approaches to Digital Forensic Engineering, 2008. SADFE '08*, Washington, IEEE Computer Society, 144-156, doi: 10.1109/SADFE.2008.9, ISBN 978-0-7695-3171-7.

ZEIDMAN, B., 2011, *The Software IP Detective's Handbook: Measurement, Comparison, and Infringement Detection*, Prentice Hall Professional, ISBN 9780137035335.

ÖZGEÇMİŞ

Doğum Tarihi ve Yeri: 9 Ocak 1983 - İstanbul

Unvanı: Araştırma Görevlisi

Kurumu: Enformatik Bölümü (2010- …)

Öğrenim Bilgileri:

Derece	Alan	Üniversite	Fakülte/Enstitü	Yıl
Ön Lisans	Bilgisayar Programlama	Sakarya Üniversitesi	Geyve M.Y.O	2002
Lisans	Fizik	İstanbul Üniversitesi	Fen Fakültesi	2008
Yüksek Lisans	Enformatik	İstanbul Üniversitesi	Fen Bilimleri Enstitüsü	2012

Yayınlar:

Özcan, M., **Özen, Z.,** Erol, Ç., Ayvaz Reis, Z., 2011, *Akademisyenler için Kişisel Web Sitesi Hazırlama Sistemi*, Akademik Bilişim'11 - XIII. Akademik Bilişim Konferansı Bildirileri, 1-5 Şubat 2011 İnönü Üniversitesi, Malatya, Baskıda.

Özen, Z., 2011, *The Advantages of the Use of Faceted Search Techniques in the Academy and Educational Software*, 2nd World Conference On Informatıon Technology, 23-27 Kasım 2011, Antalya, Baskıda.

Gülseçen, S., Ayvaz Reis, Z., Erol, Ç., Kartal Karataş, E., Gezer, M., Şimşek, İ., Asaadi, Y., Derelioğlu, Y., Yıldırım, K., Gürsul, F., Mutlu, D., Şişman, B., **Özen, Z.**, 2011, *Teaching computer programming online or face to face: a quantitative study,* 2nd World Conference On Informatıon Technology, 23-27 Kasım 2011, Antalya, Baskıda.